

WAKU-RLN-RELAY: Privacy-Preserving Peer-to-Peer Economic Spam Protection

Sanaz Taheri (Vac Research and Development, Status Research and Development)

Oskar Thoren (Vac Research and Development, Status Research and Development)

Barry Whitehat (Unaffiliated)

Wei Jie Koh (Independent)

Onur Kilic (Unaffiliated)

Kobi Gurkan (cLabs)

Outline

- WAKU
- WAKU-RELAY: Privacy-Preserving Gossip-Based Routing
- Spam issue in WAKU-RELAY
- Anonymity and Spam Protection
- State-of-the-art P2P Spam Protection Methods
- WAKU-RLN-RELAY: Global Spam Protection and Anonymity Made Possible

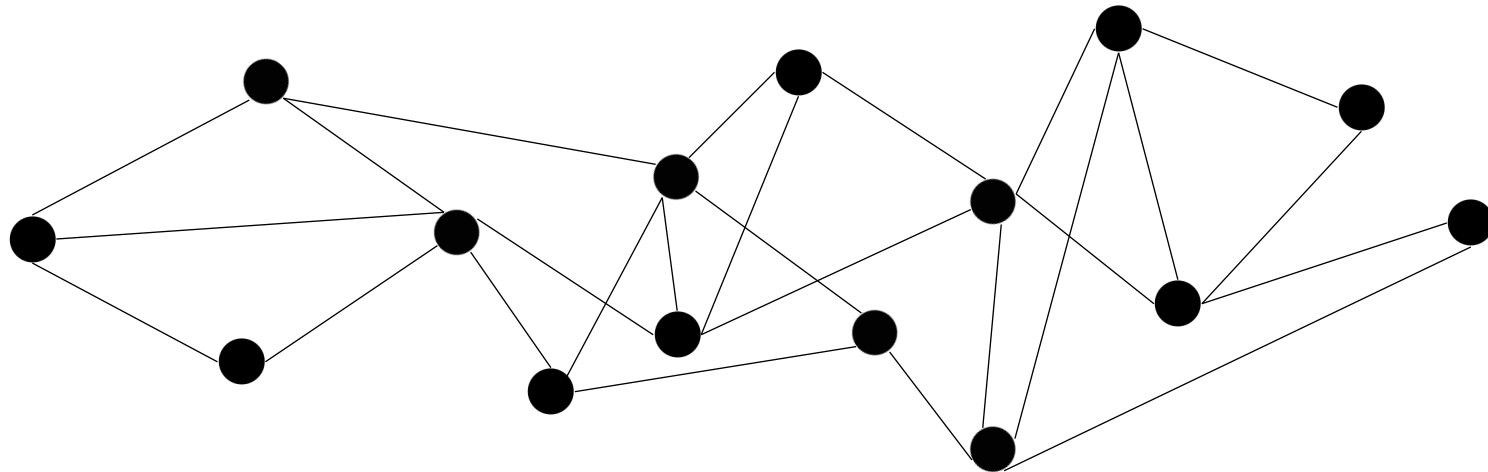
WAKU

- WAKU [1] aims to be Web3 messaging layer.
- Features 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.
- WAKU protocols include:
 - **WAKU-RELAY: privacy-preserving transport**
 - **WAKU-RLN-RELAY: spam-protected version of WAKU-RELAY**
 - WAKU-STORE: historical message storage
 - WAKU-FILTER: light version of WAKU-RELAY for bandwidth limited devices
 - And many more ...
- The full list of RFCs is available in rfc.vac.dev.

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

WAKU-RELAY

- WAKU-RELAY [1] is a Peer-to-Peer transport protocol

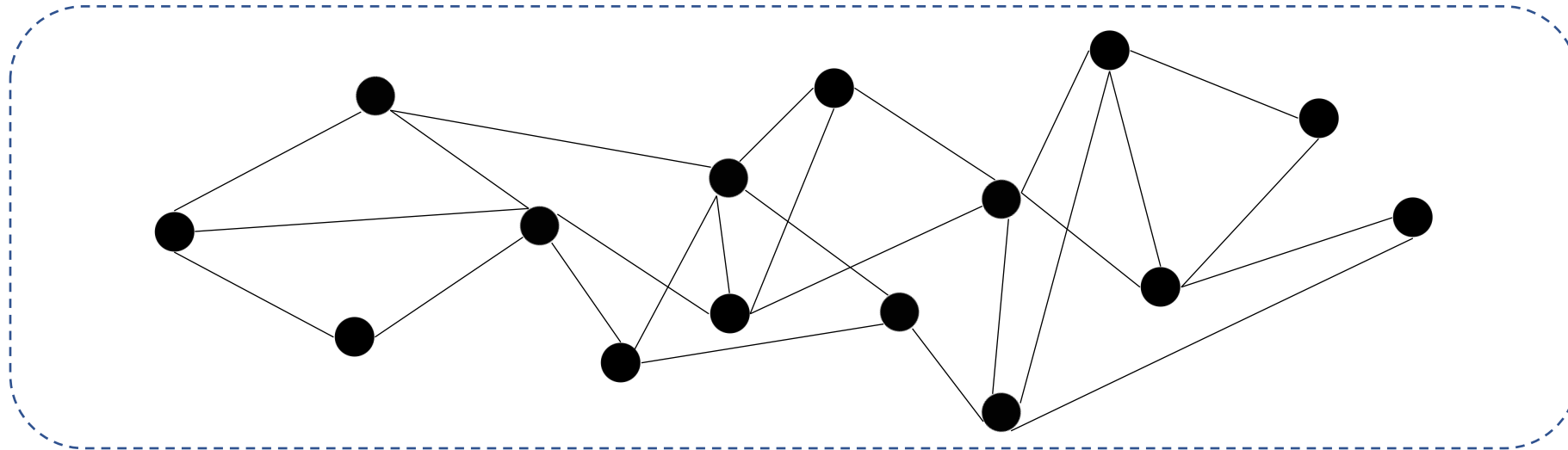


[1] <https://rfc.vac.dev/spec/11/>

WAKU-RELAY [1]

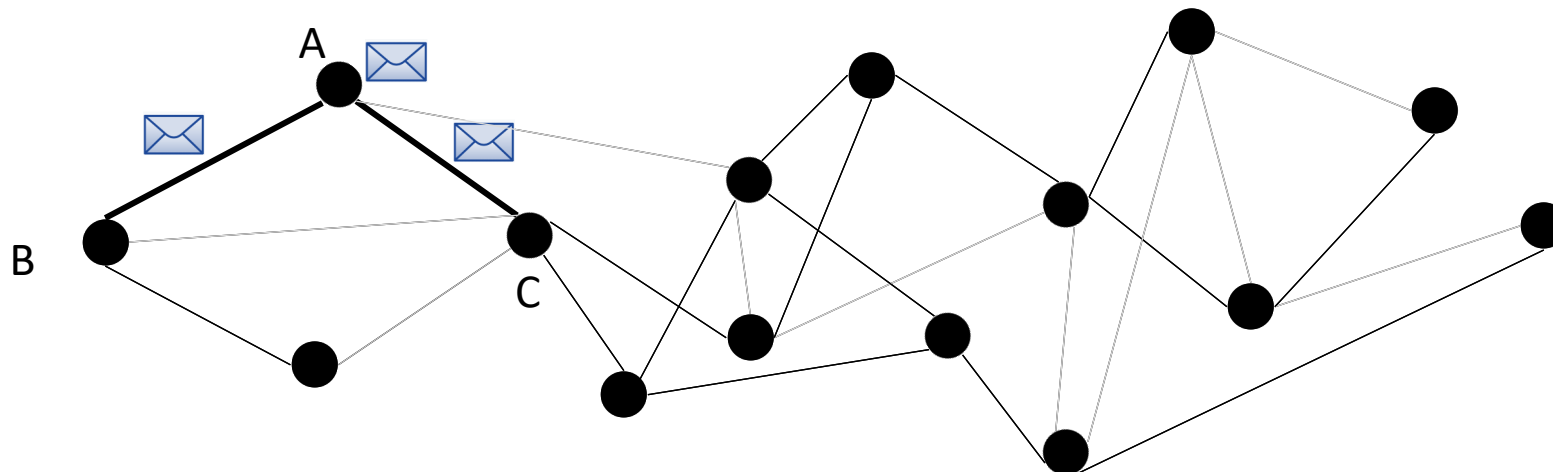
- WAKU-RELAY is a Peer-to-Peer transport protocol
- Follows publisher-subscriber Model
 - Peers subscribed to the same topic form a mesh

Mesh of peers subscribed to the same topic



WAKU-RELAY

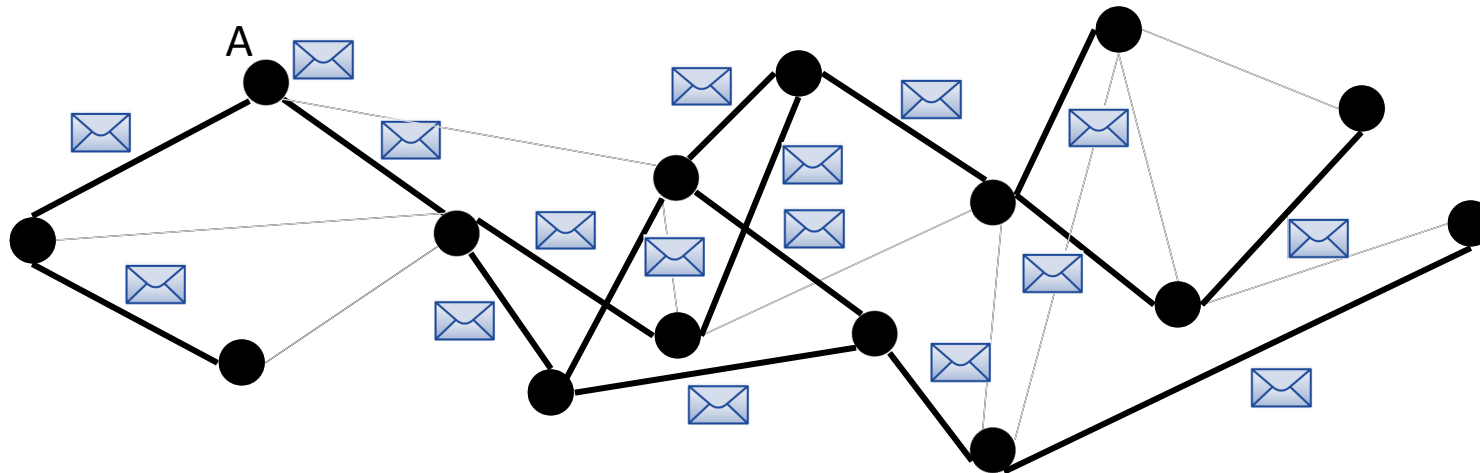
- WAKU-RELAY is a Peer-to-Peer transport protocol
- Follows publisher-subscriber Model
- Gossip-based routing (extension of libp2p GossipSub-v1.1 [1])



[1] <https://github.com/libp2p/specs/tree/master/pubsub/gossipsub>

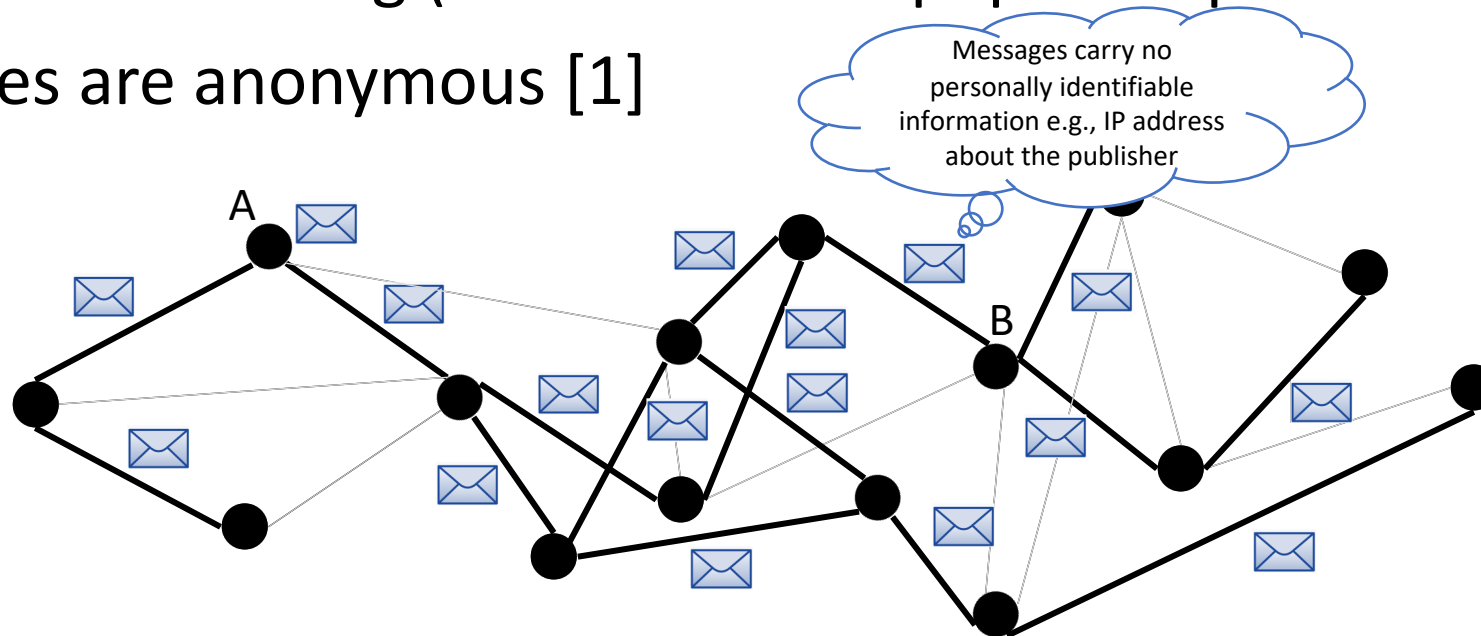
WAKU-RELAY

- WAKU-RELAY is a Peer-to-Peer transport protocol
- Follows publisher-subscriber Model
- Gossip-based routing (extension of libp2p GossipSub-v1.1)



WAKU-RELAY

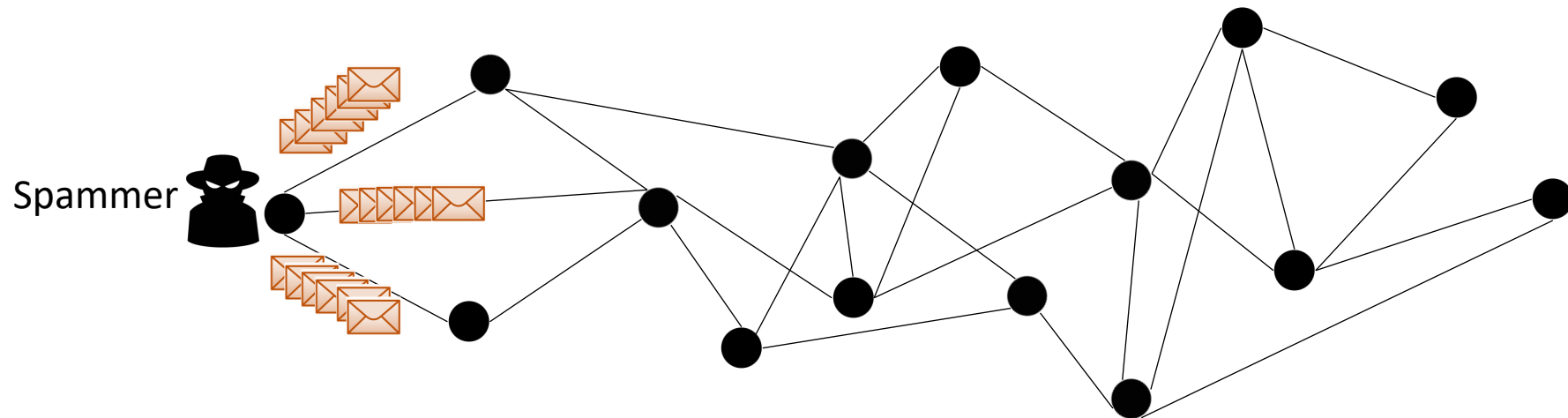
- WAKU-RELAY is a Peer-to-Peer transport protocol
- Follows publisher-subscriber Model
- Gossip-based routing (extension of libp2p GossipSub-v1.1)
- Messages are anonymous [1]



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

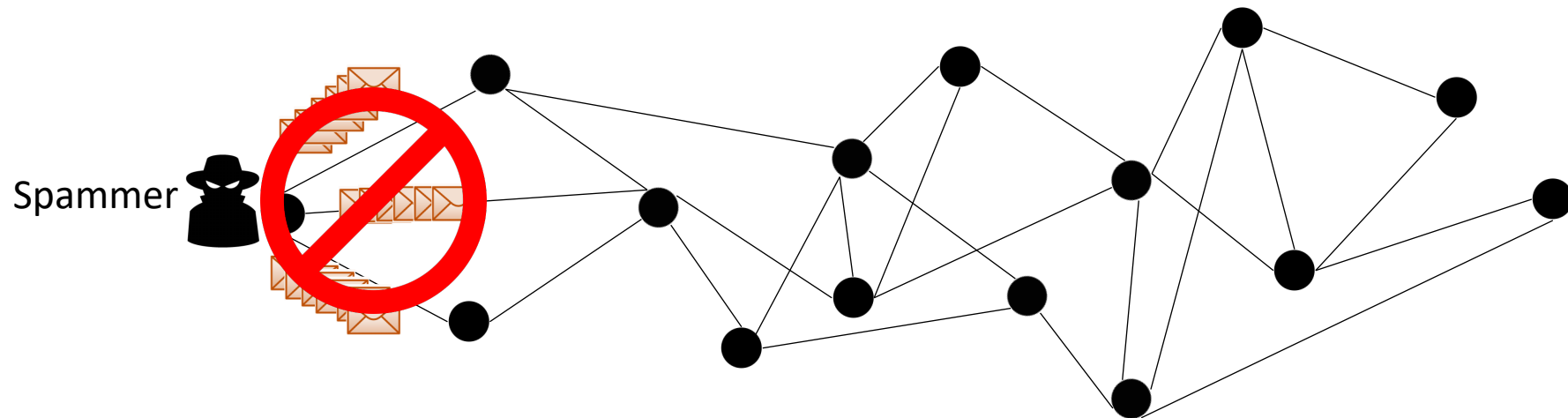
Spam Issue In WAKU-RELAY

- Spammers: entities that publish a large number of messages in a short amount of time, and cause **Denial-of-Service**.



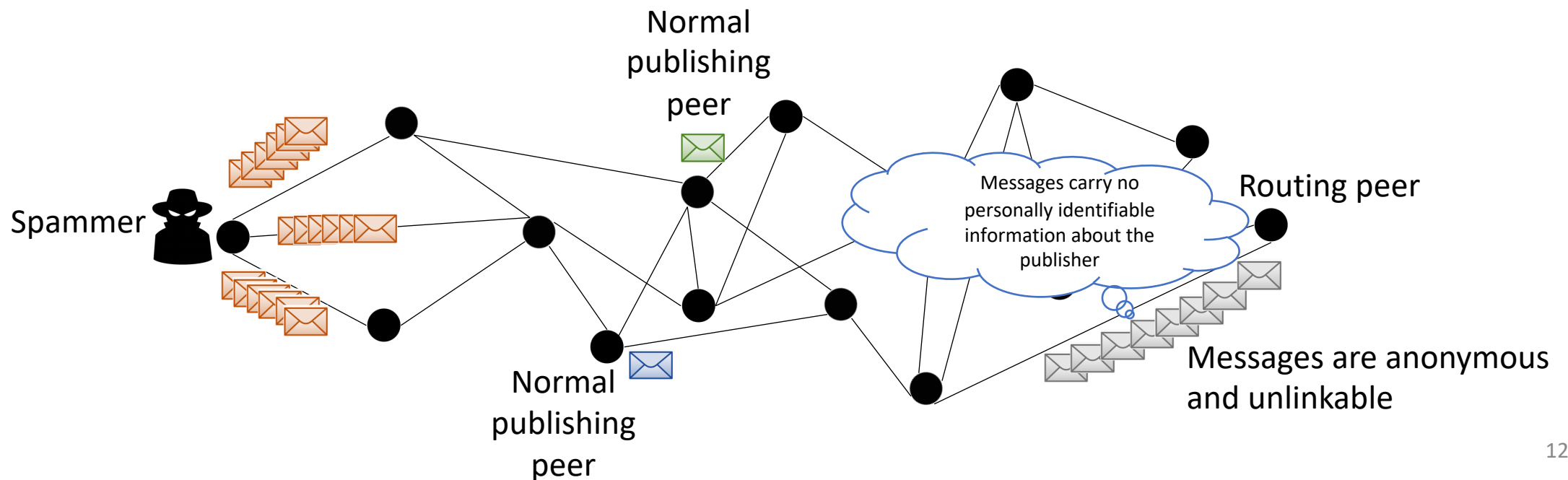
Spam Issue In WAKU-RELAY

- Spammers: entities that publish a large number of messages in a short amount of time, and cause [Denial-of-Service](#).
- Spam Protection = Controlled Messaging Rate



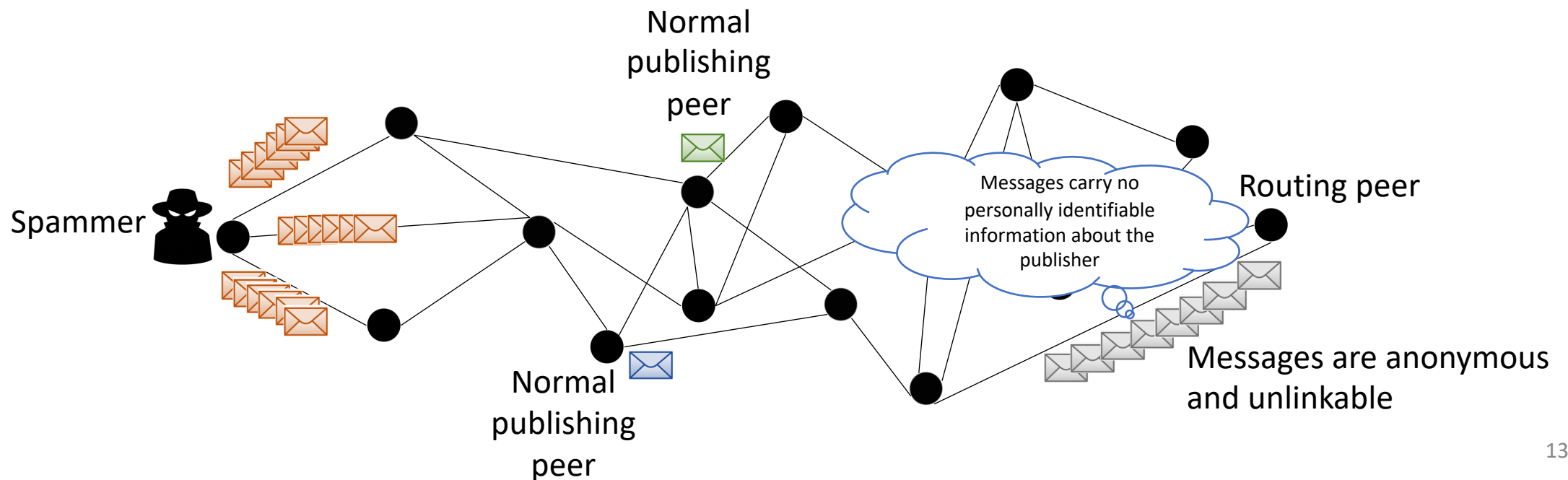
Anonymity and Spam Protection

- Messages are anonymous and not linkable to their origin.



Anonymity and Spam Protection

- Messages are anonymous and not linkable to their origin.
- Solutions like IP blocking are not effective.



State-of-the-art P2P Spam Protections

Proof-of-work [1] deployed by Whisper [2]

- Messaging rate is proportional to the peer's computational power.
- Computationally expensive.
- Not suitable for network of heterogeneous peers with limited resources.

Peer Scoring [3] adopted by libp2p

- Local to each peer.
- No clear measure of spamming behavior in an anonymous messaging network.
- No global identification of spammer.
- Subject to inexpensive attacks using bots.

[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.md#peerscoring>.



WAKU-RLN-RELAY: Global Spam Protection and Anonymity Made Possible

WAKU-RLN-RELAY

- WAKU-RLN-RELAY [1] = WAKU-RELAY + Rate Limiting Nullifiers (RLN)
- It is a proof of stake rate limited messaging layer: No one can message more than an application-defined messaging rate without being financially punished.

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

RLN Primitive

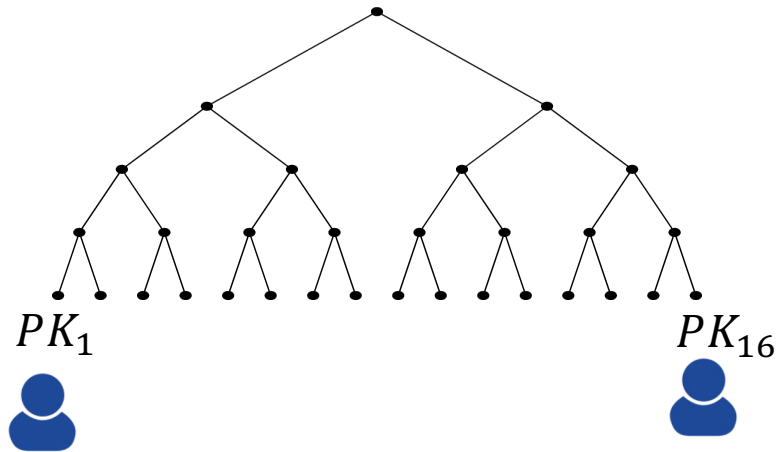
RLN Primitive

- RLN [1] 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>

RLN Primitive: Membership Tree

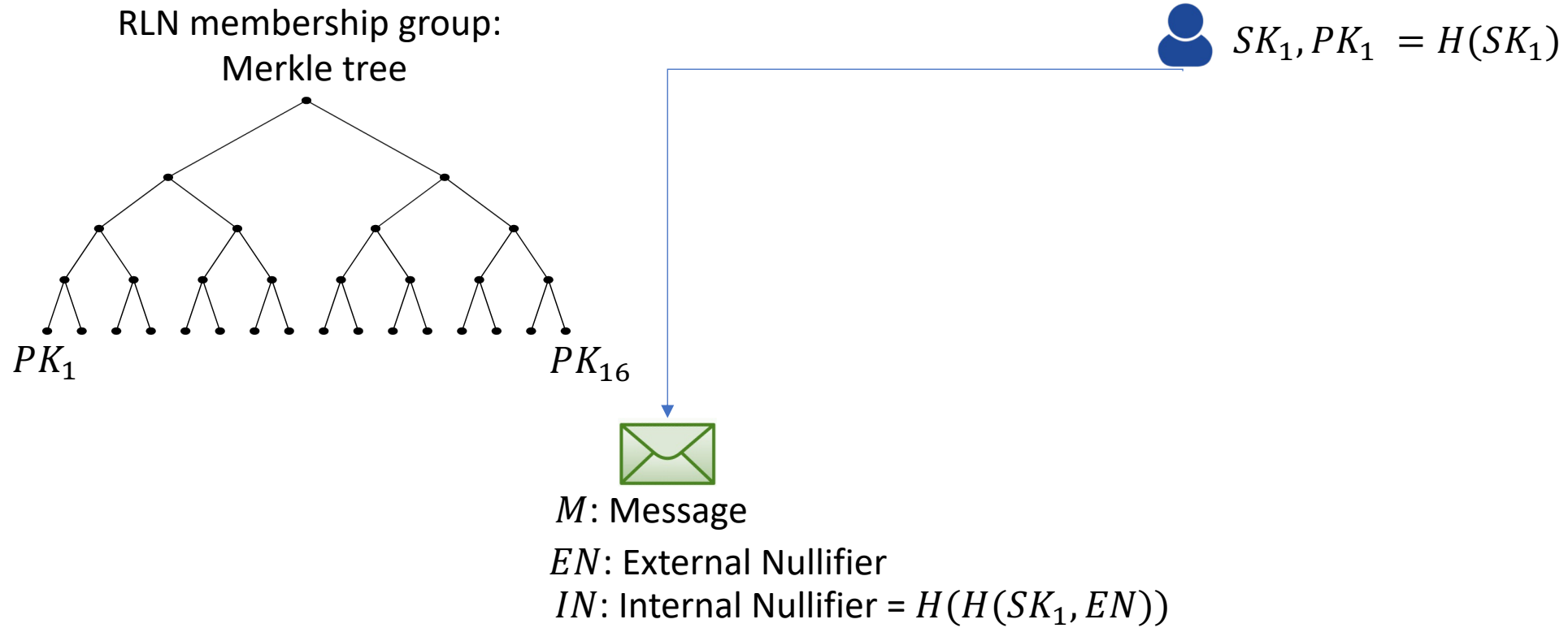
RLN membership group:
Merkle tree



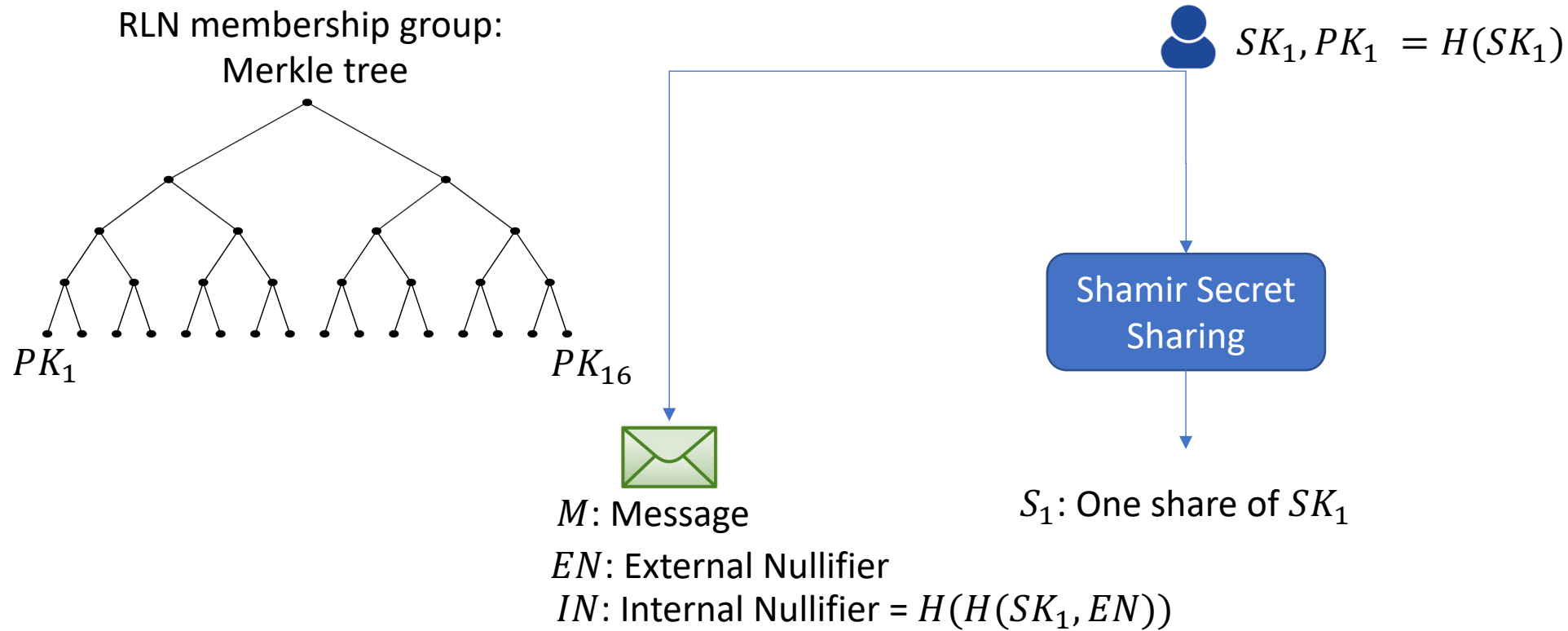
$$SK_1, PK_1 = H(SK_1)$$

$$SK_{16}, PK_{16} = H(SK_{16})$$

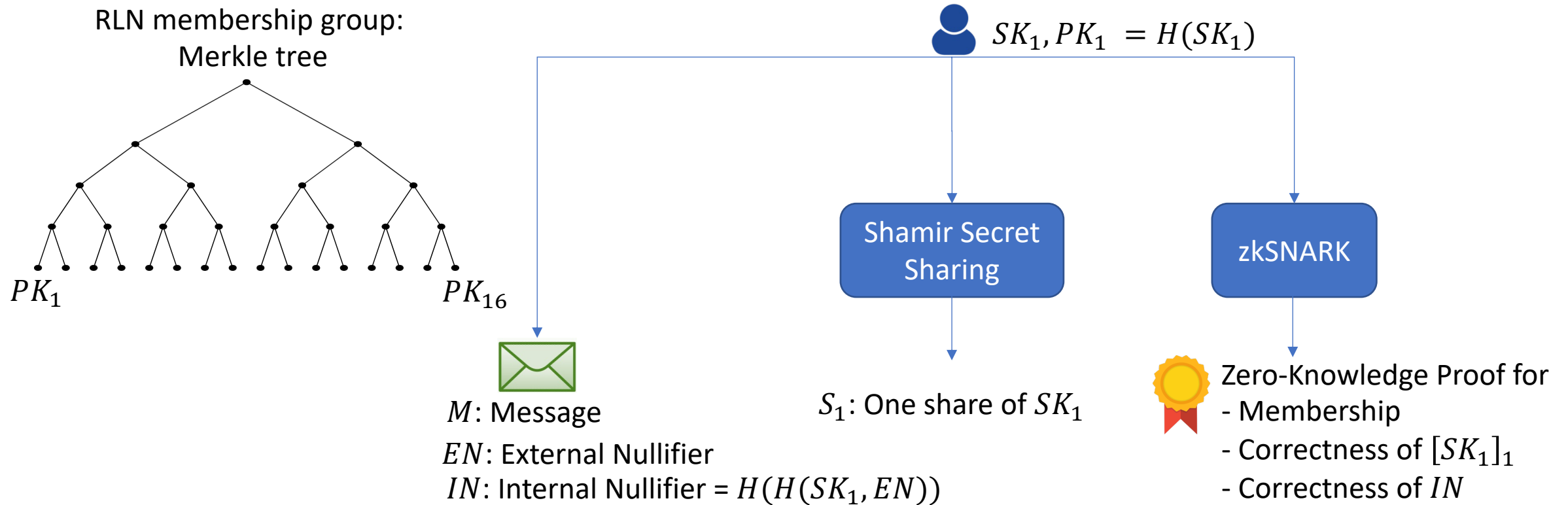
RLN Primitive: Signaling



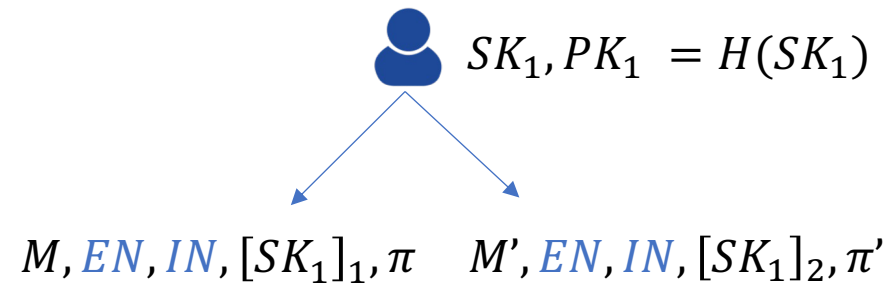
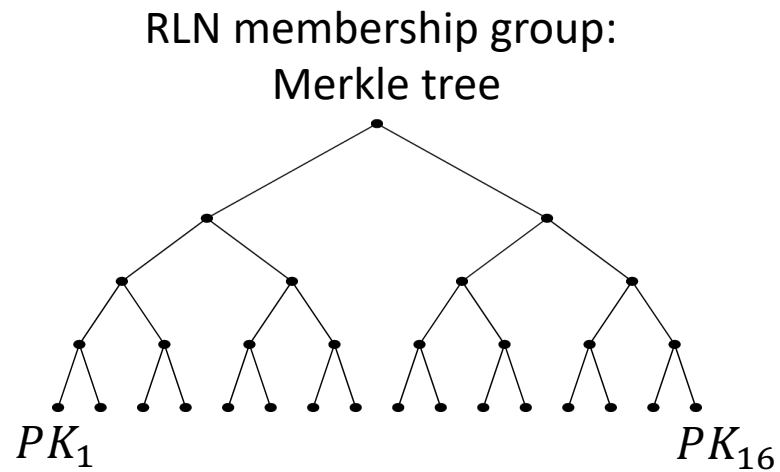
RLN Primitive: Signaling



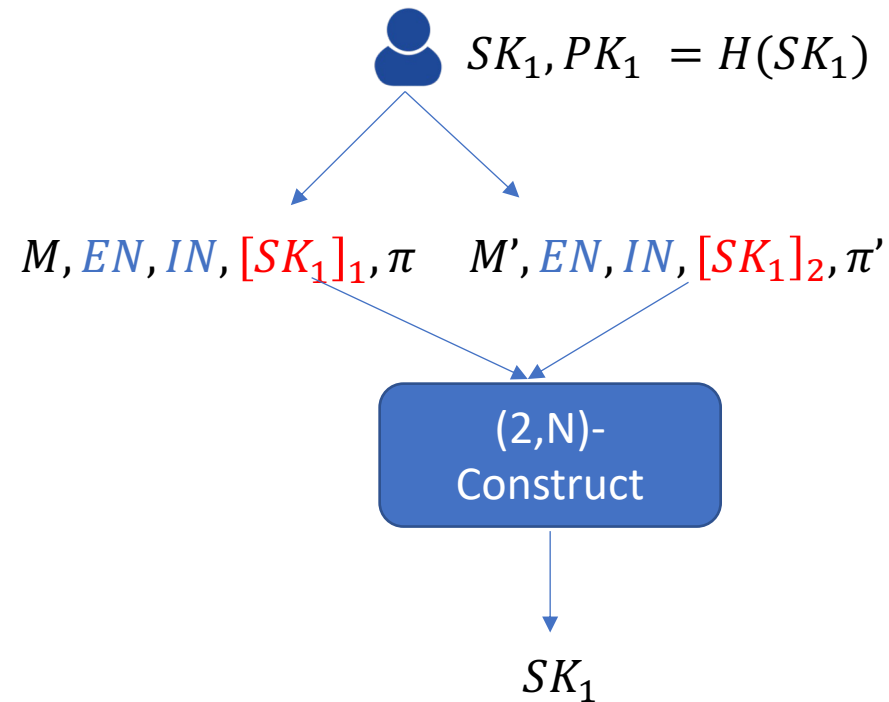
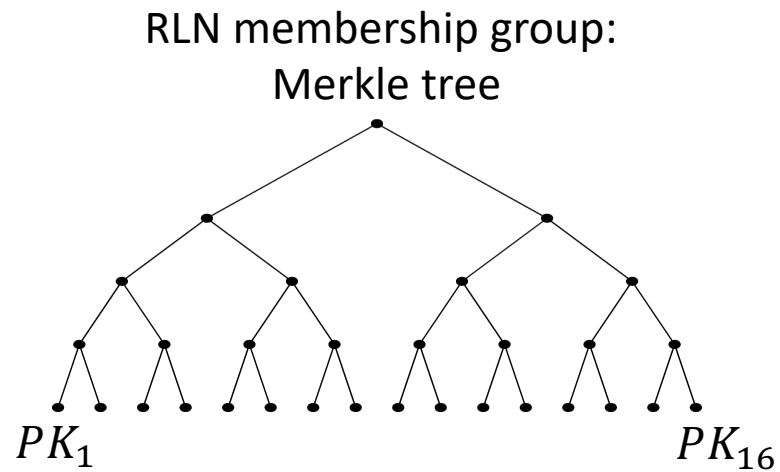
RLN Primitive: Signaling



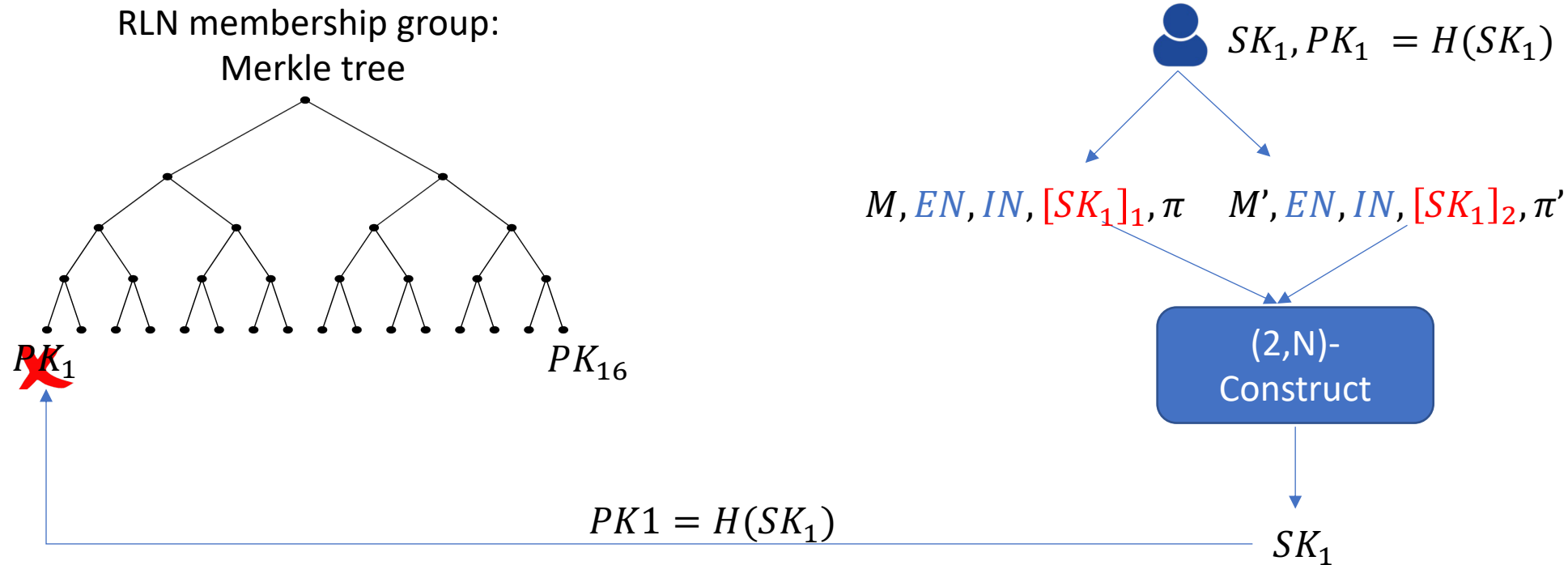
RLN Primitive: Detecting Double Signaling



RLN Primitive: Slashing



RLN Primitive: Slashing

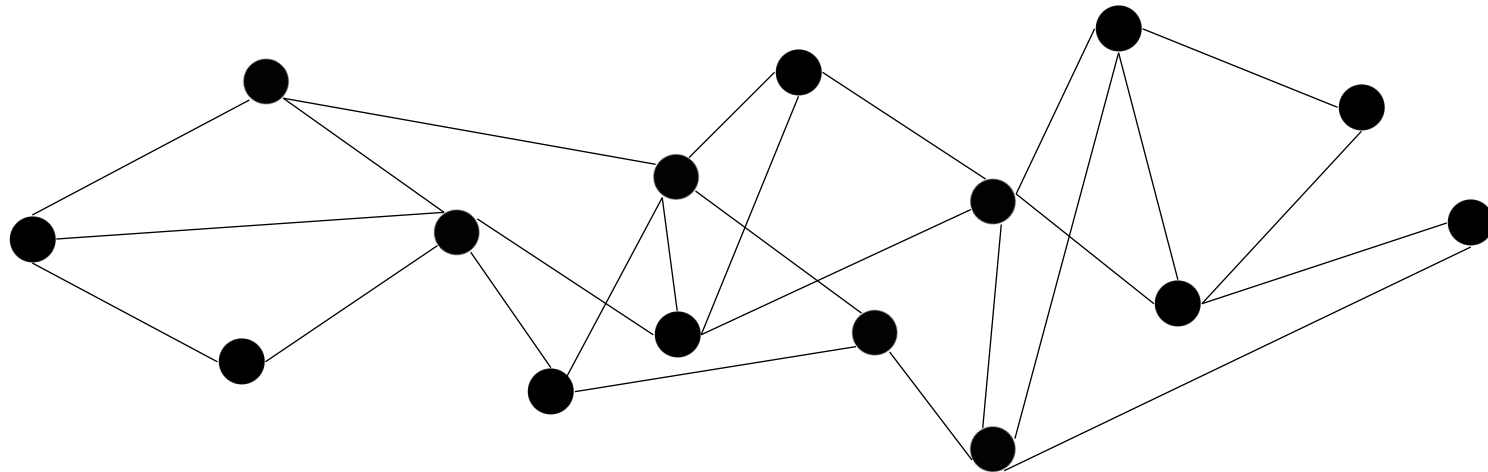




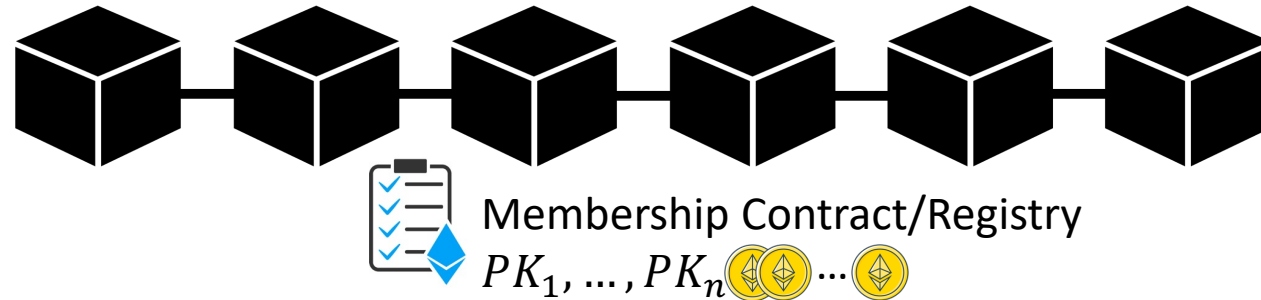
WAKU-RLN-RELAY Construction

WAKU-RLN-RELAY: RLN Group

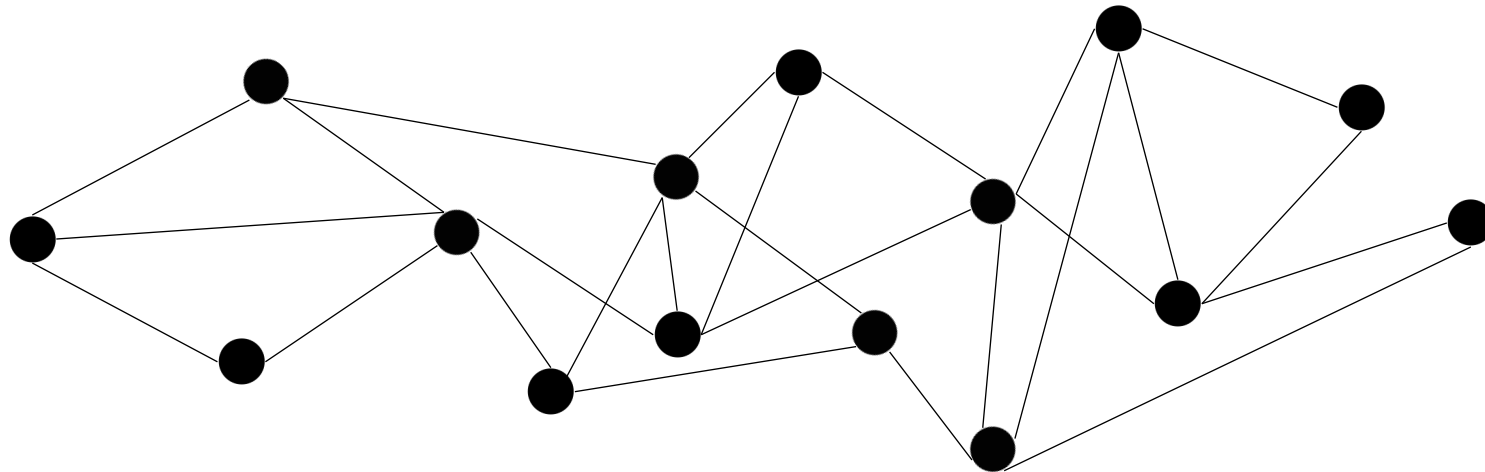
RLN group = Peers
subscribed to the
same topic e.g.,
WAKU-RLN-RELAY



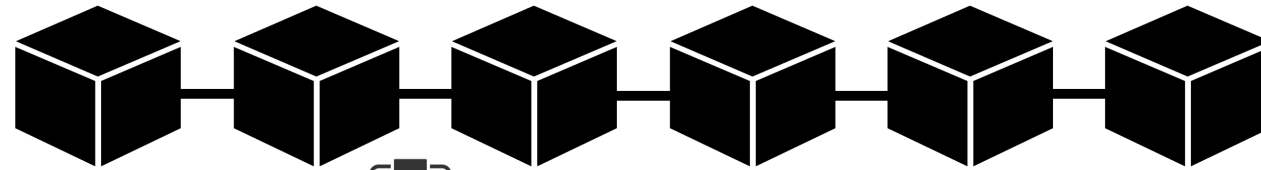
WAKU-RLN-RELAY: Registration






RLN group = Peers
subscribed to the
same topic e.g.,
WAKU-RLN-RELAY



WAKU-RLN-RELAY: Registration

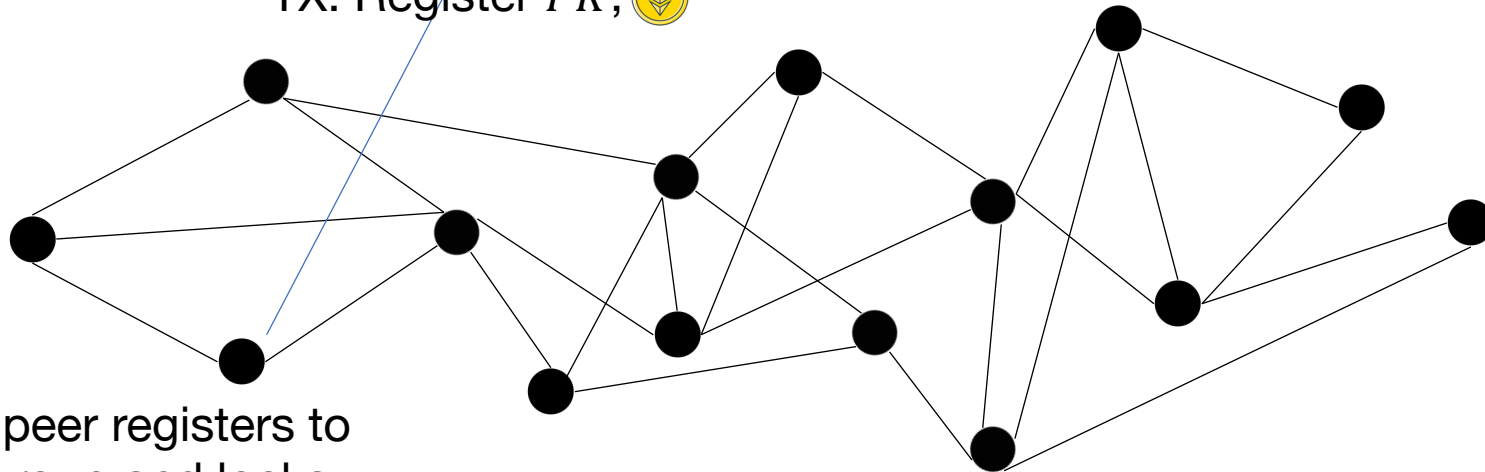


Membership Contract/Registry

PK_1, \dots, PK_n   ... 

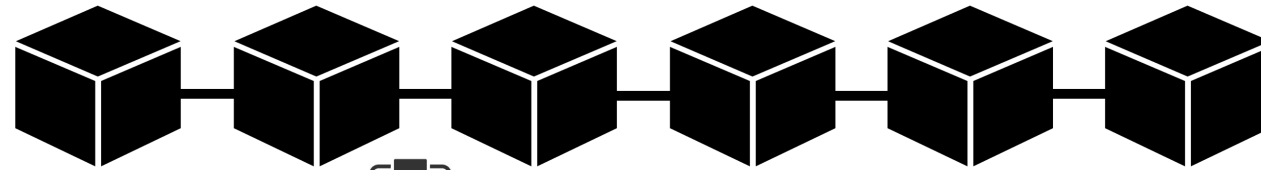
TX: Register PK , 

RLN group = Peers
subscribed to the
same topic e.g.,
WAKU-RLN-RELAY




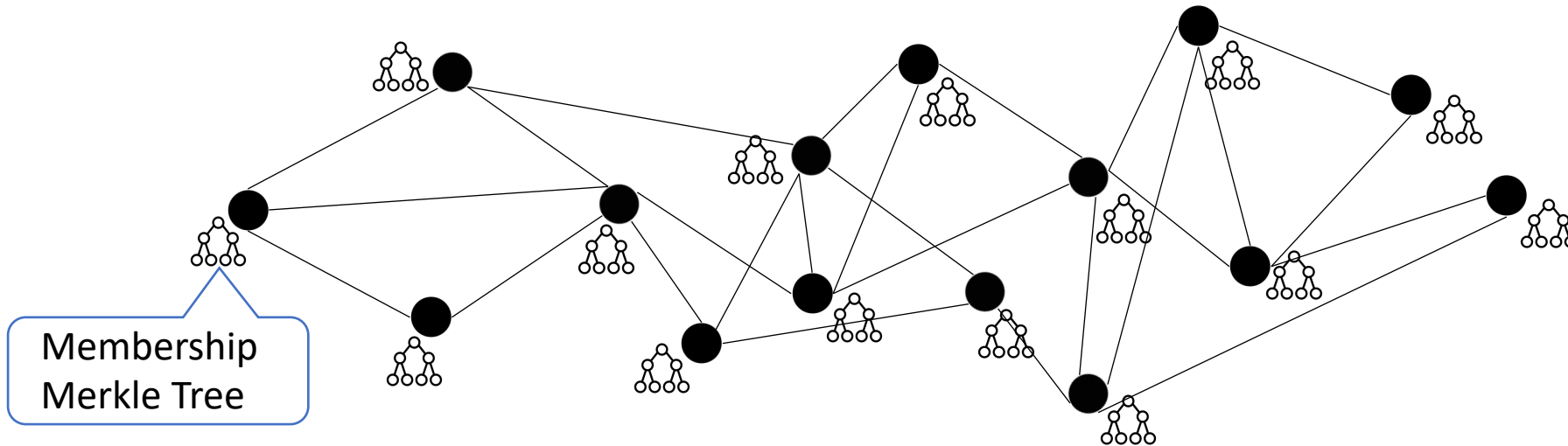
Each peer registers to
the group and locks
some funds

WAKU-RLN-RELAY: Registration

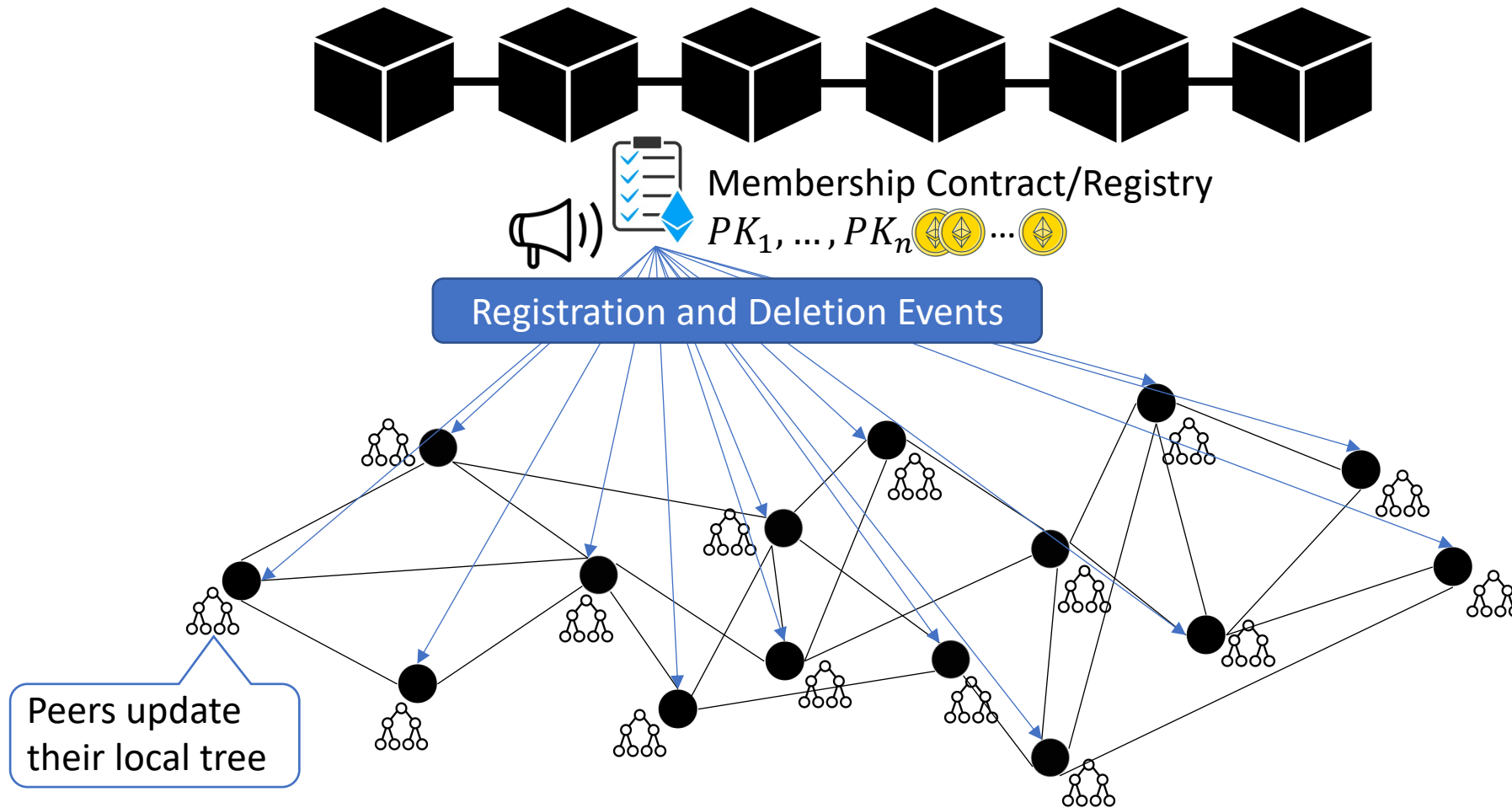


Membership Contract/Registry

PK_1, \dots, PK_n 

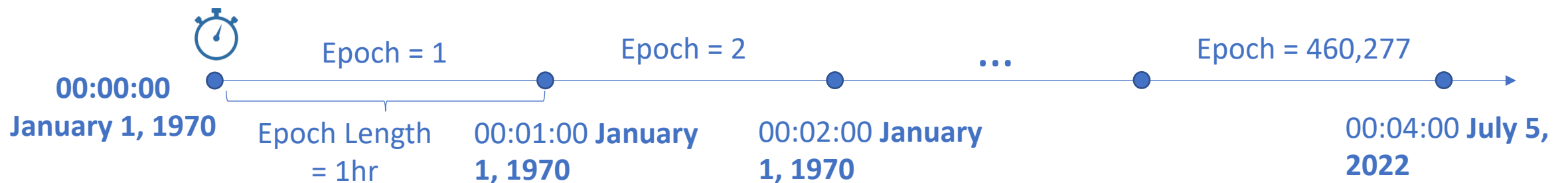


WAKU-RLN-RELAY: Registration



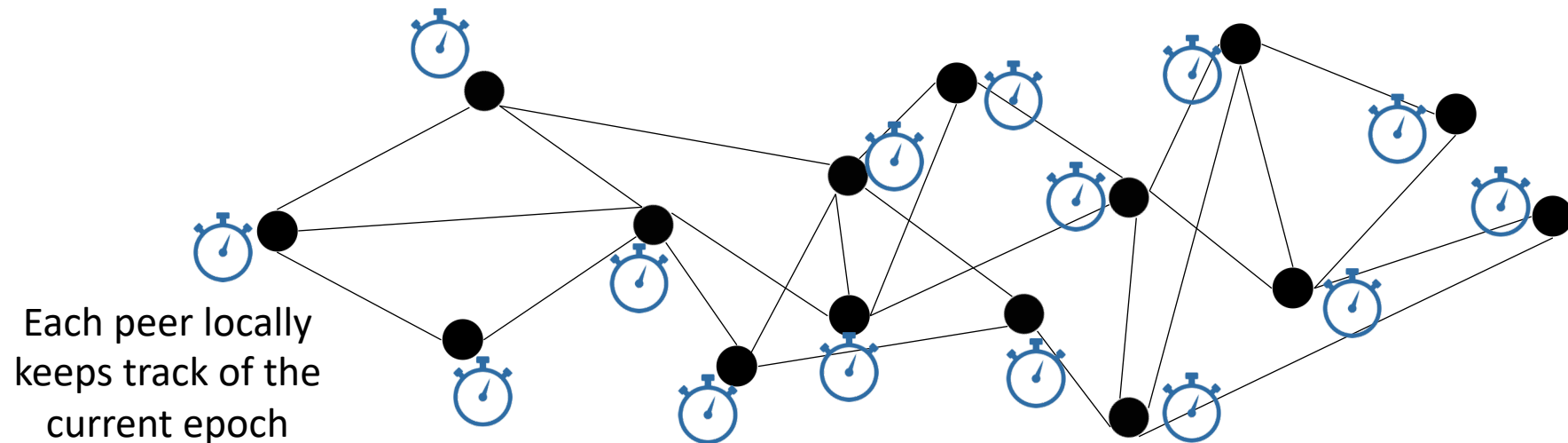
WAKU-RLN-RELAY: Messaging Rate

- Messaging rate is limited to 1 per Epoch (Epoch is the same as External Nullifier).
- The time is divided into some intervals, and each interval corresponds to one Epoch.
- The epoch length affects the messaging rate and is application-dependent.



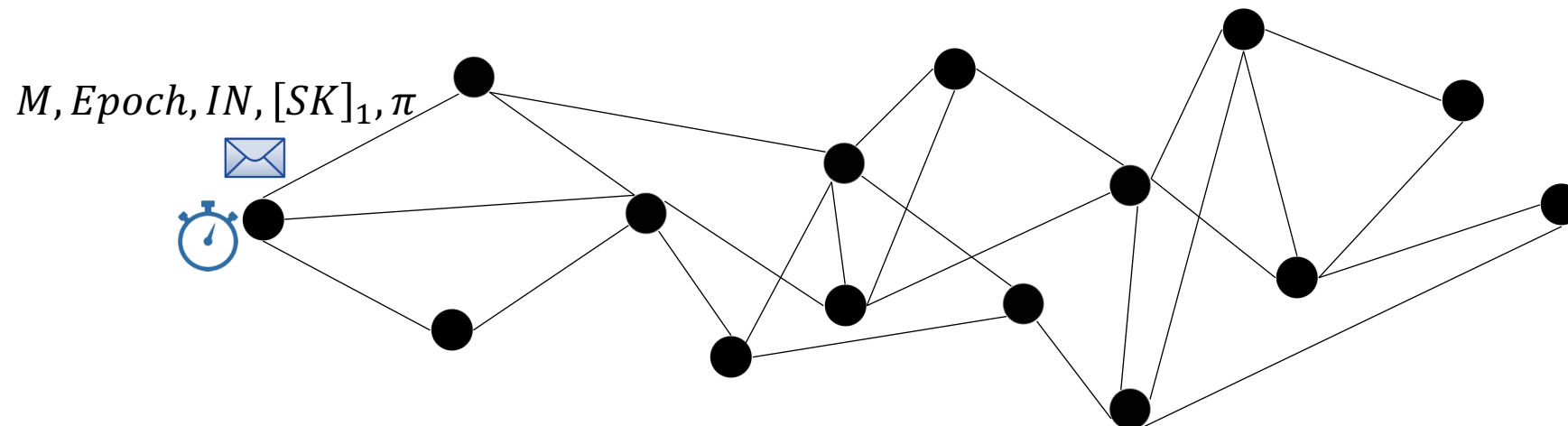
WAKU-RLN-RELAY: Messaging Rate

- Peers keep track of the current epoch count.



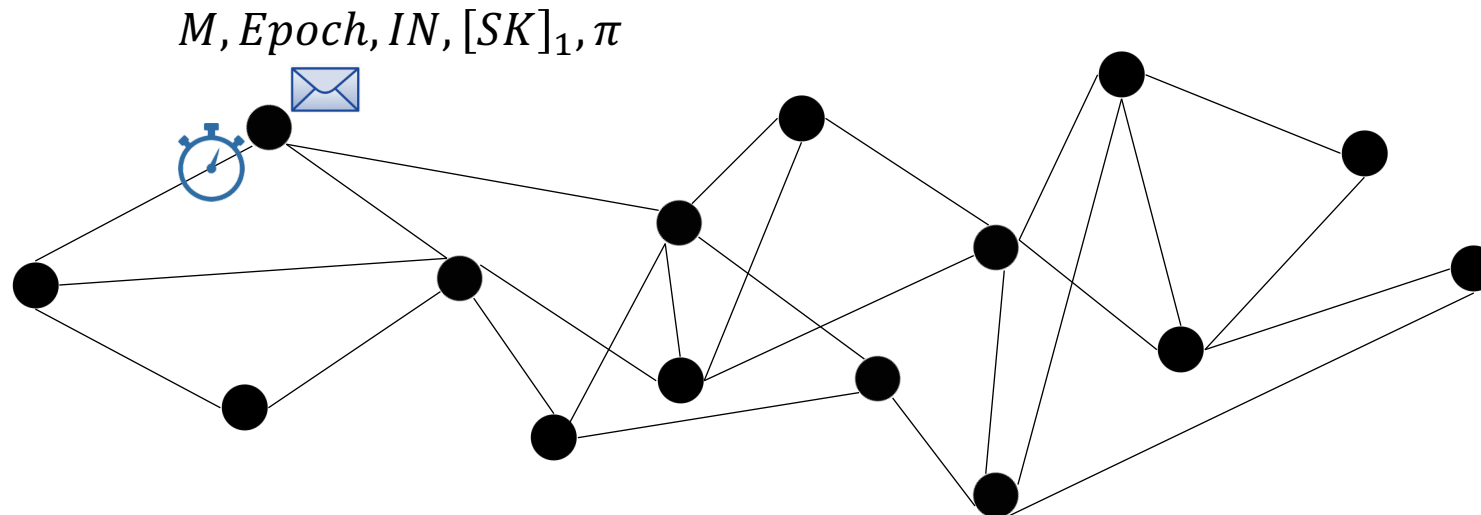
WAKU-RLN-RELAY: Publishing

- Each message is an RLN signal.
- The message owner, attaches the nullifiers, together with a share of its RLN secret key, and the zero-knowledge proof part to the message and sends it to its local mesh.

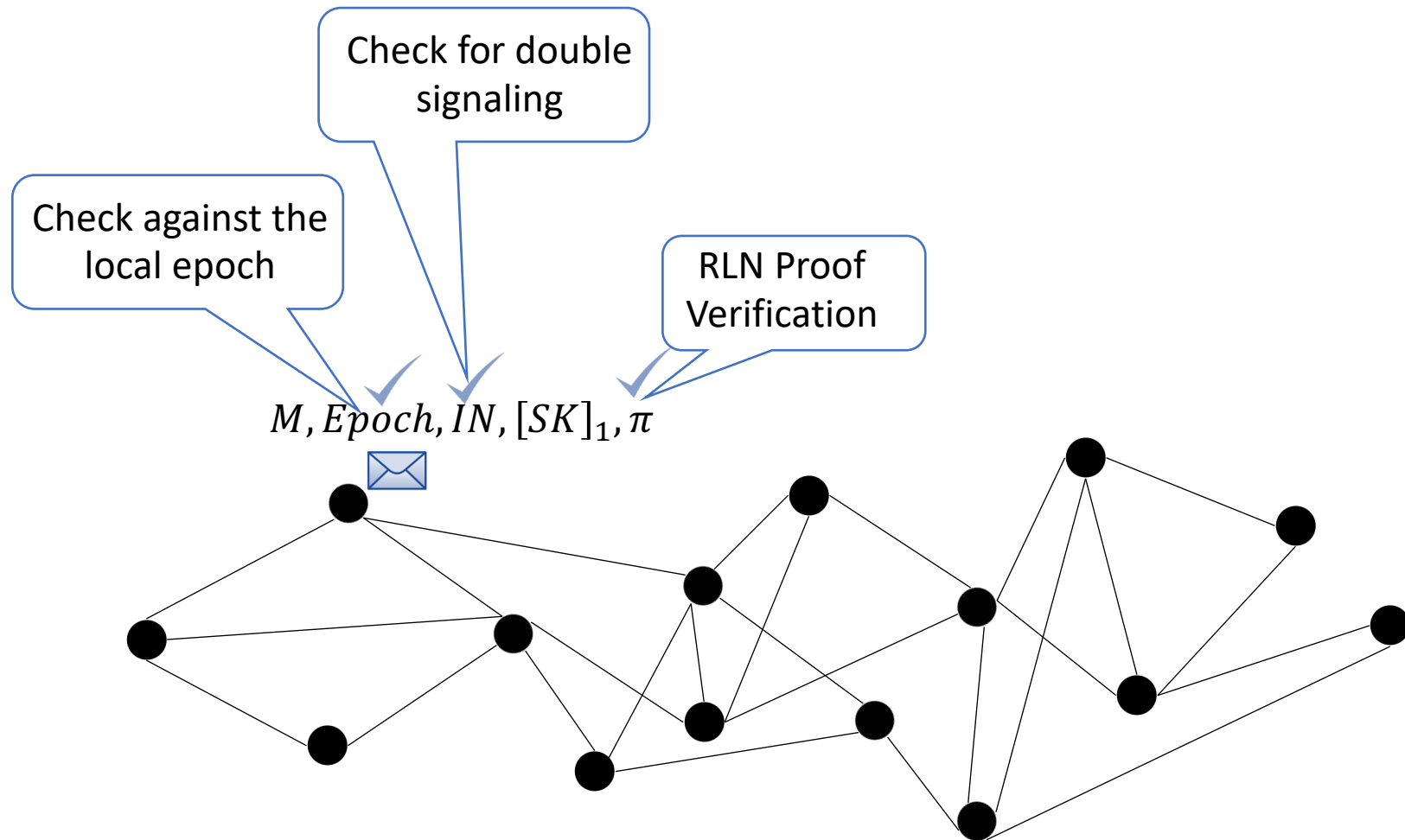


WAKU-RLN-RELAY: Routing

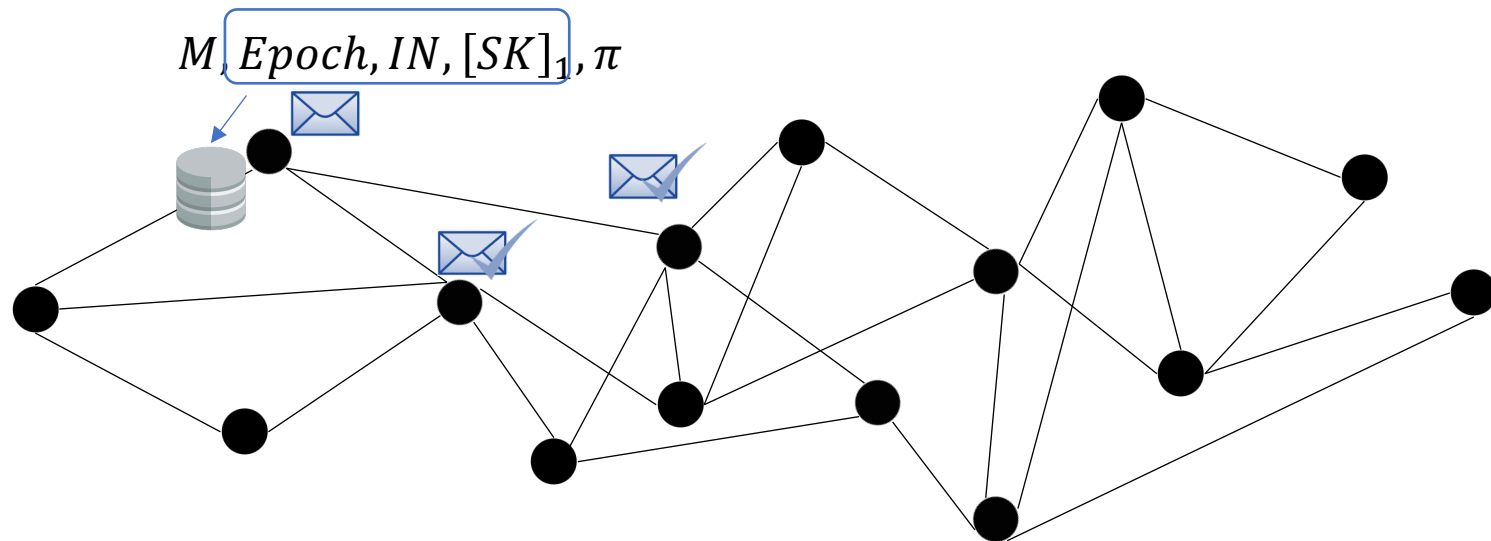
- Regular Gossip-based routing protocol + RLN signal verification (installed as a libp2p GossipSub topic validator).



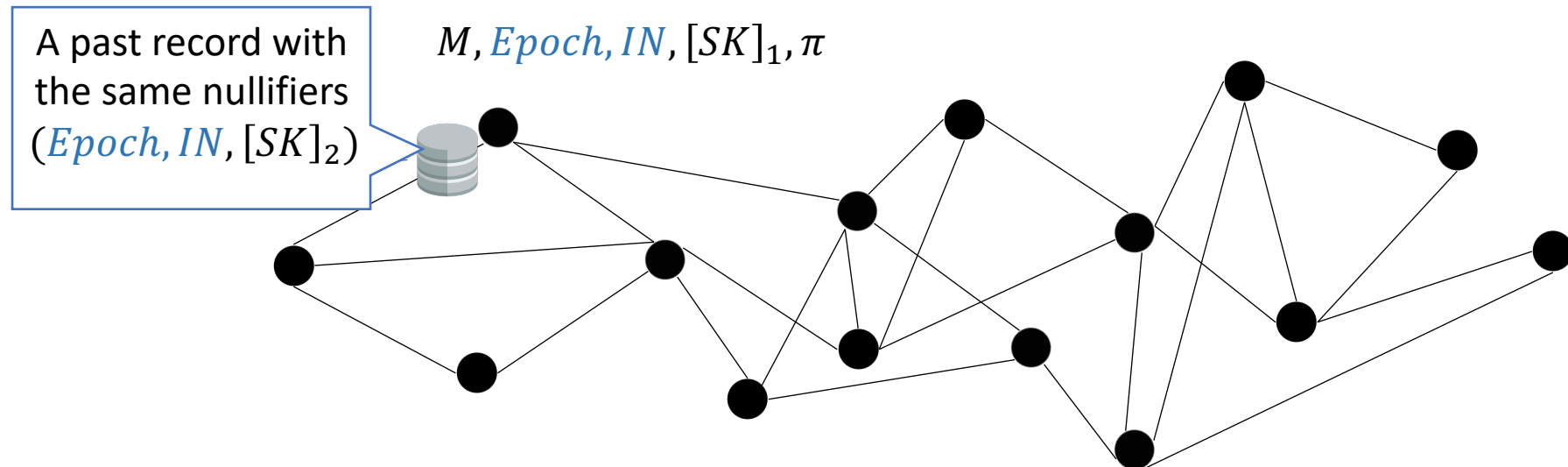
WAKU-RLN-RELAY: Routing



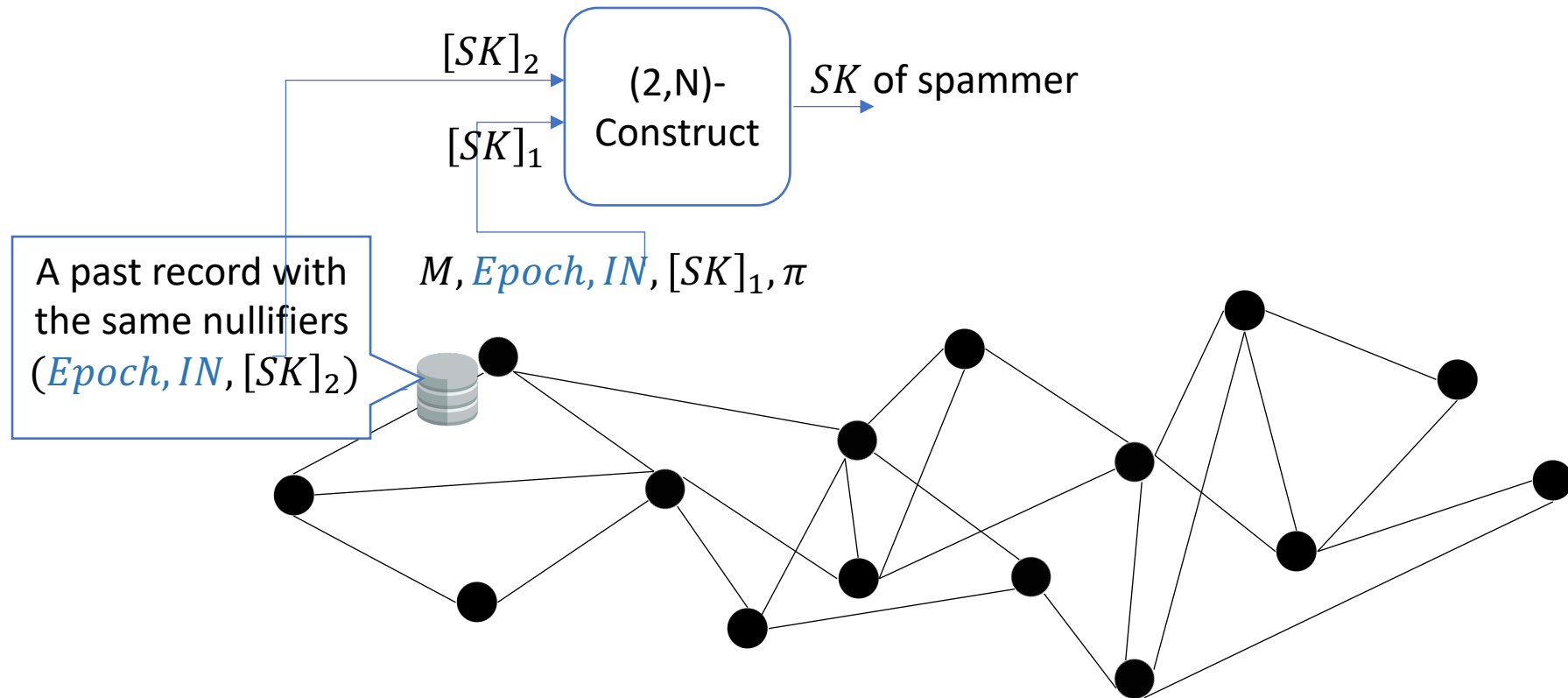
WAKU-RLN-RELAY: Routing



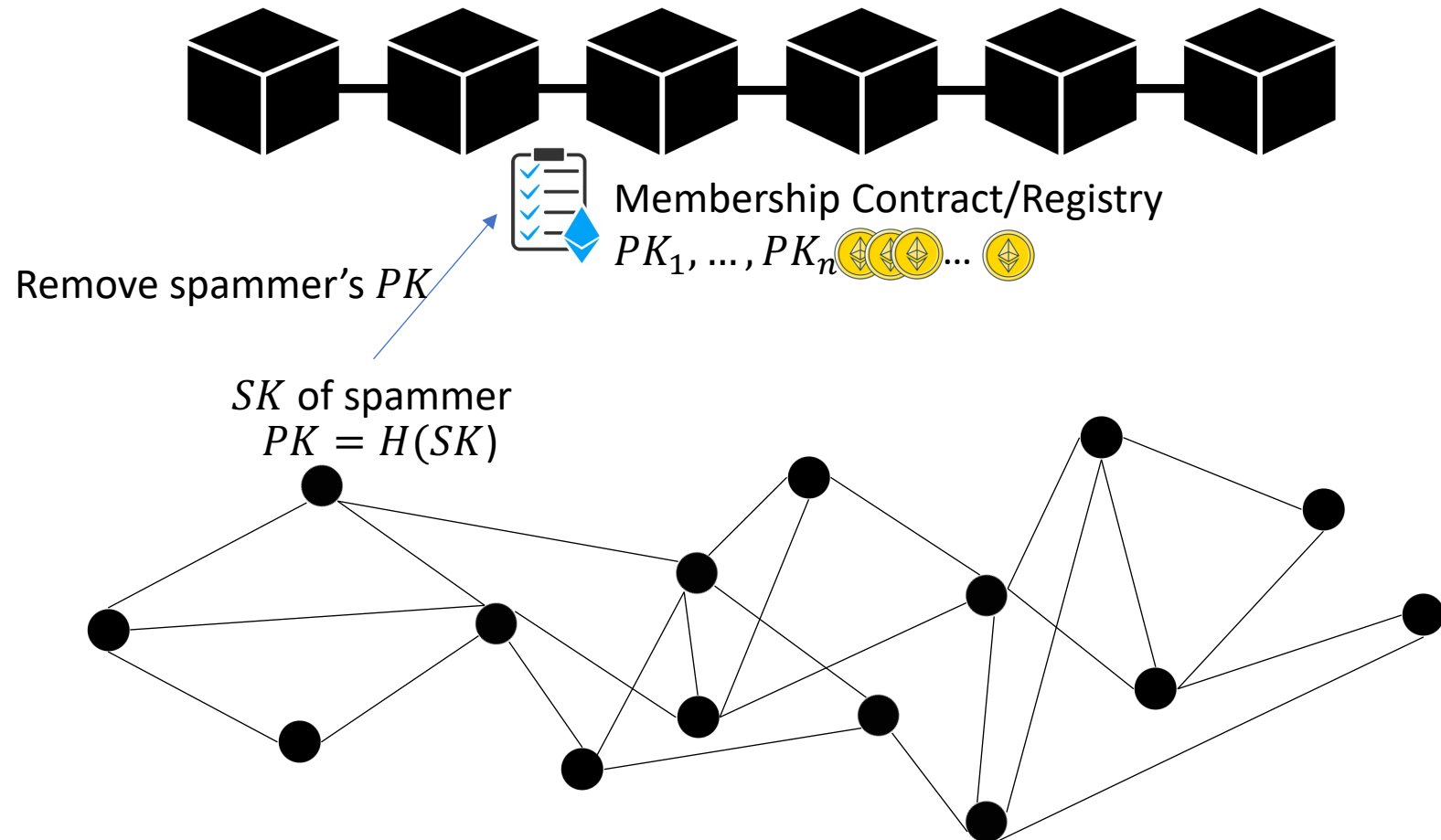
WAKU-RLN-RELAY: Slashing



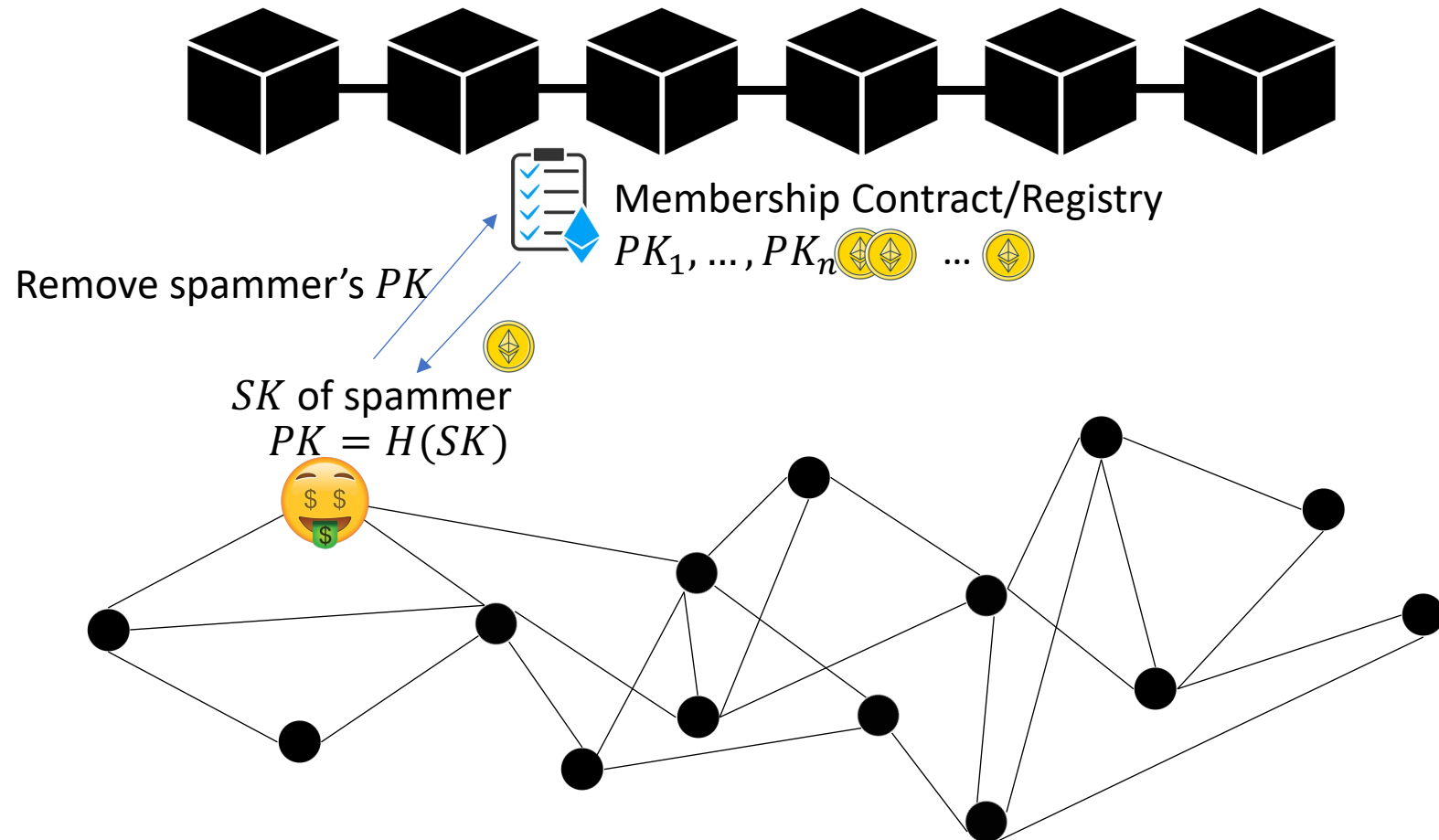
WAKU-RLN-RELAY: Slashing



WAKU-RLN-RELAY: Slashing



WAKU-RLN-RELAY: Slashing



WAKU-RLN-RELAY

WAKU-RLN-RELAY [1] = WAKU-RELAY + Rate Limiting Nullifiers (RLN)

- P2P global spam protection
- Privacy-preserving
- Economic incentives
 - Financial punishment for the spammers and a financial reward for those who catch spammers

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

WAKU-RLN-RELAY Proof-Of-Concept

- The PoC of WAKU-RLN-RELAY is implemented in Nim and is available in the Waku repository <https://github.com/status-im/nwaku>
- A PoC rate-limited command-line chat application using WAKU-RLN-RELAY is implemented in Nim <https://github.com/status-im/nwaku>
 - Instructions on how to run it can be found under the tutorial section i.e. <https://github.com/status-im/nwaku/tree/master/docs/tutorial>

References

- WAKU-RLN-RELAY specs: <https://rfc.vac.dev/spec/17/>
- RLN specifications: <https://rfc.vac.dev/spec/32/>
- The Nim implementation of WAKU-RLN-RELAY: <https://github.com/status-im/nim-waku>
- The JavaScript implementation of Waku: <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>



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

System Parameters

Parameter	Description
Epoch length	The length of epoch in seconds. Application dependent, should be set based on desired throughput.
Maximum Epoch Gap	The maximum allowed gap between the epoch of a routing peer and the incoming message. Should be set based on measures the maximum number of epochs that can elapse since a message gets routed from its origin to all the other peers in the network. Can be calculated as: $\left\lceil \frac{\text{message propagation delay} + \text{clock asynchrony}}{\text{epoch length}} \right\rceil$
Staked fund	The amount of ether to be staked by peers at the registration
Reward portion	The percentage of staked fund to be rewarded to the slashers

Message propagation delay: the maximum time that it takes for a message to be fully disseminated in the GossipSub network.

Clock asynchrony: The maximum difference between the Unix epoch clocks perceived by network peers which can be due to clock drifts.

System throughput

- Message propagation delay
 - Graph diameter * message verification time (30 ms)
 - Potential increased network delay for transportation of RLN related data
- System throughput i.e., messaging rate
 - Affected by the message propagation delay and the epoch length



Implementation setup

- WAKU-RLN-RELAY 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**.
- The instantiated hash function is **Poseidon** with the security level of **128 bits**.
- Proof system is **Groth16** [2].



[1] <https://github.com/kilic/rln>

[2] 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.

Computation overhead

- Proof generation: According to the benchmarking report of [1] for a Merkle tree **depth of 24**, the **proof generation** on an **iPhone 8** takes approximately **~0.5 seconds**.
- **Proof Verification**: is constant and takes approximately **~ 30 milliseconds**.
- 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>

Gas Cost

- **PK Registration:** The estimated gas cost is **40k**.
- **PK Slashing/Deletion:** The estimated gas cost is **40k**.
- **Batch registration/slashing:** The estimated gas cost is **20k**. A Batch consists of $B=128$ keys



Storage overhead

- The **Merkle tree** with **depth 20** takes up **~67MB** storage. With some optimizations it can be reduced to the order of tens of KBs.
- **Identity keys and identity commitment** keys are each of size **32 Bytes**.
- **Prover key** size is approximately **~3.24 MB**.
- **Nullifier map** consists of the internal nullifier and the secret shares of the messages published in the last valid epochs (i.e., not older than the maximum epoch gap). Metadata for each message is of size 3×32 Bytes.



Bandwidth

- Bandwidth overhead = 416 bytes \sim 0.4 KB
 - Merkle tree root: 32 bytes
 - Nullifier: 32 bytes
 - ZKSNARK proof: 256 bytes
 - Epoch: 32 bytes
 - Secret shares: $2 * 32$ bytes



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