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

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

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

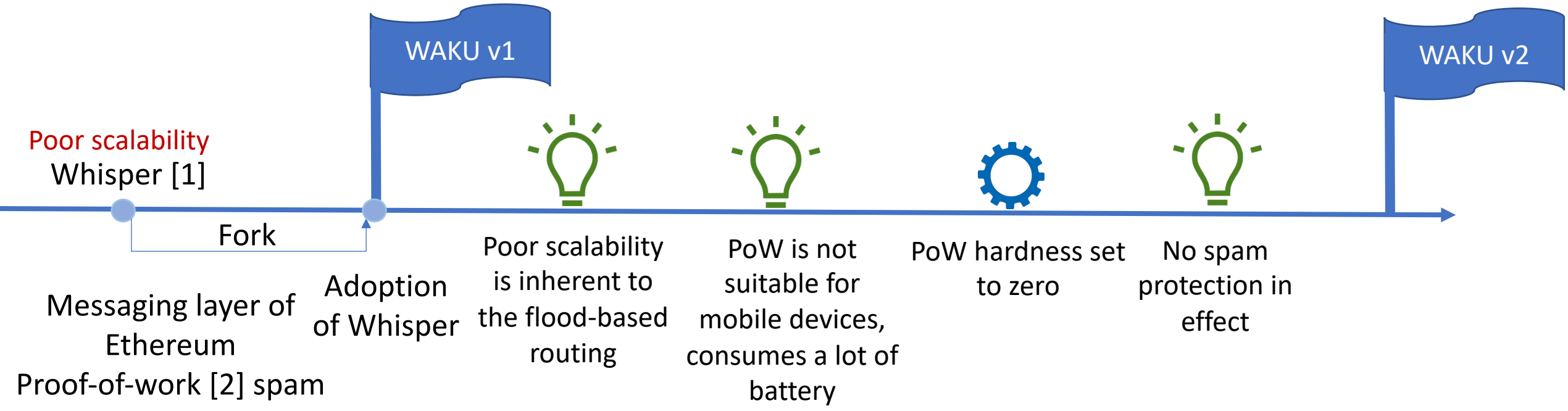
Barry Whitehat (Unaffiliated)

Wei Jie Koh (Independent)

Onur Kilic (Unaffiliated)

Kobi Gurkan (cLabs)

WAKU: Web3 Messaging Layer



[1] https://eips.ethereum.org/eips/eip-627.

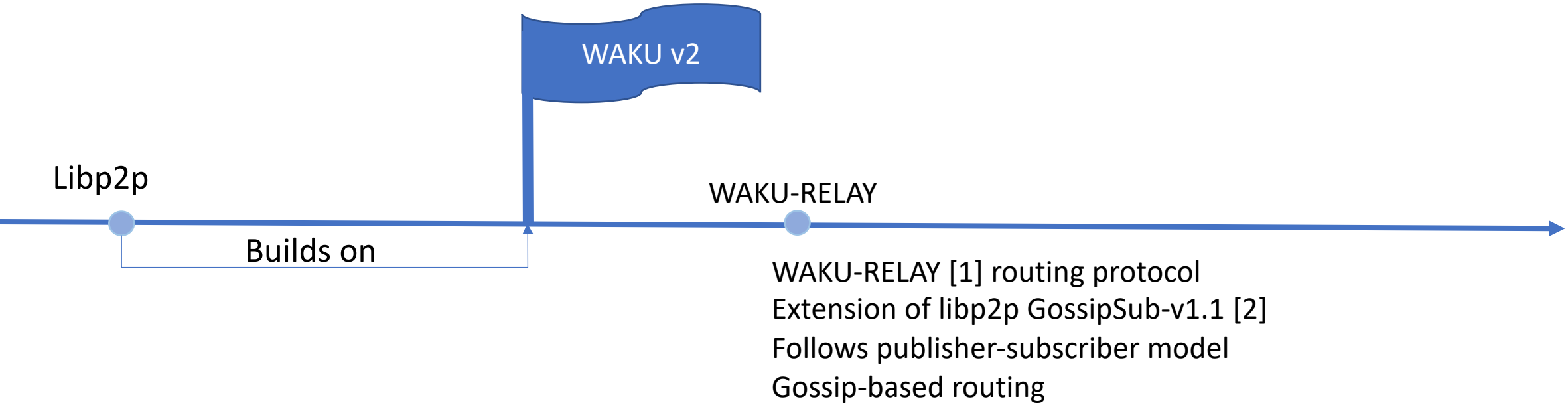
[2] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Annual 456 international cryptology conference. Springer, 1992.

WAKU: Web3 Messaging Layer

- WAKU v2 [1] builds on top of libp2p
- Suite of privacy-preserving, modular, p2p protocols aimed at resource-limited devices

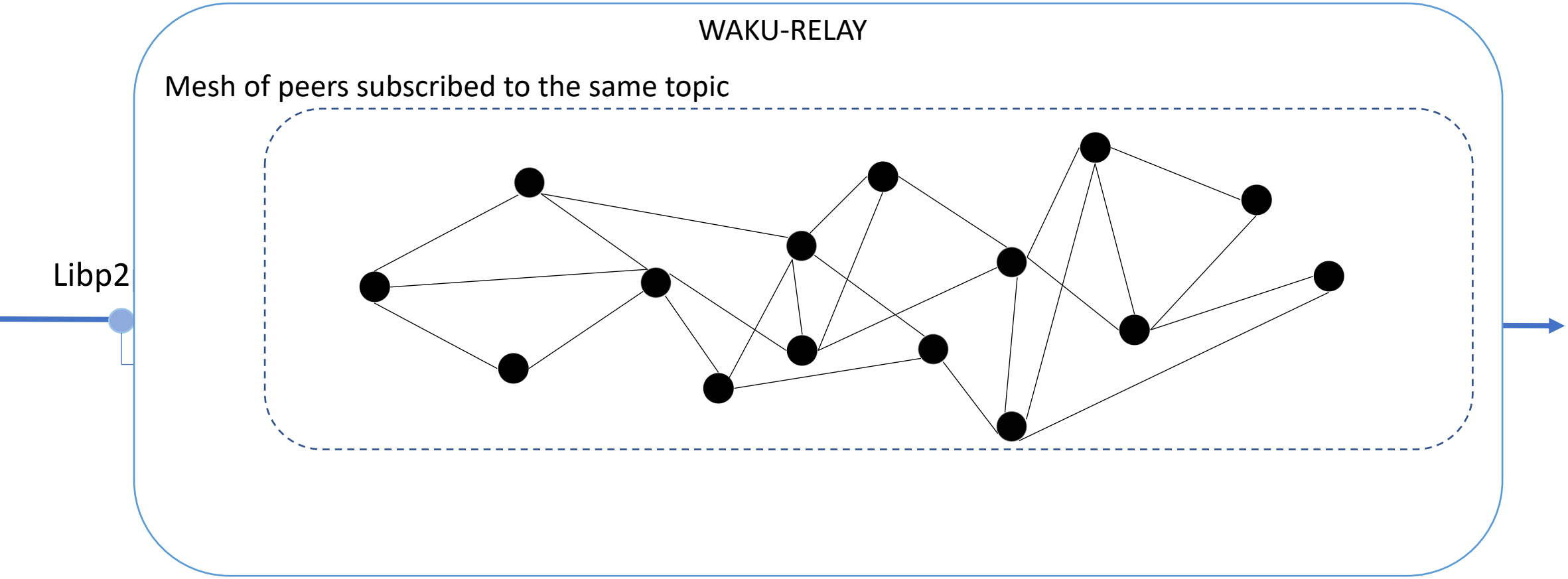
[1] https://rfc.vac.dev/spec/10/

WAKU: Web3 Messaging Layer

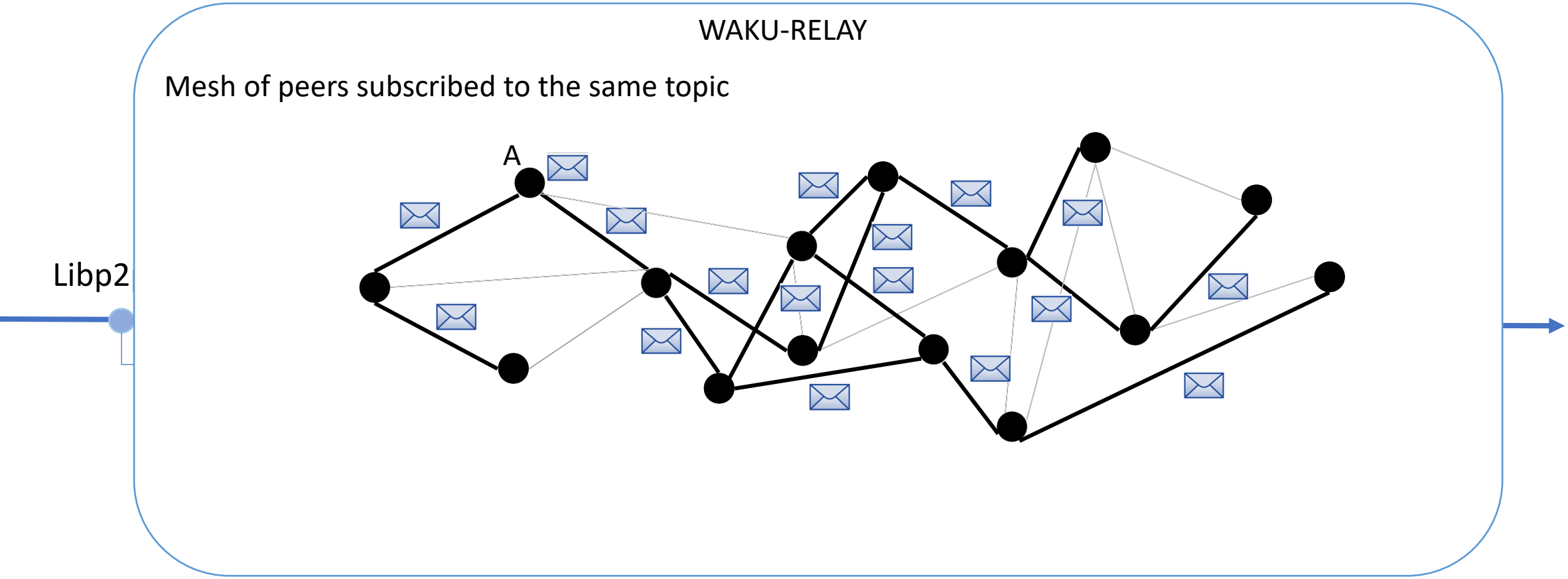


[1] https://rfc.vac.dev/spec/10/
[2] https://github.com/libp2p/specs/tree/master/pubsub/gossipsub

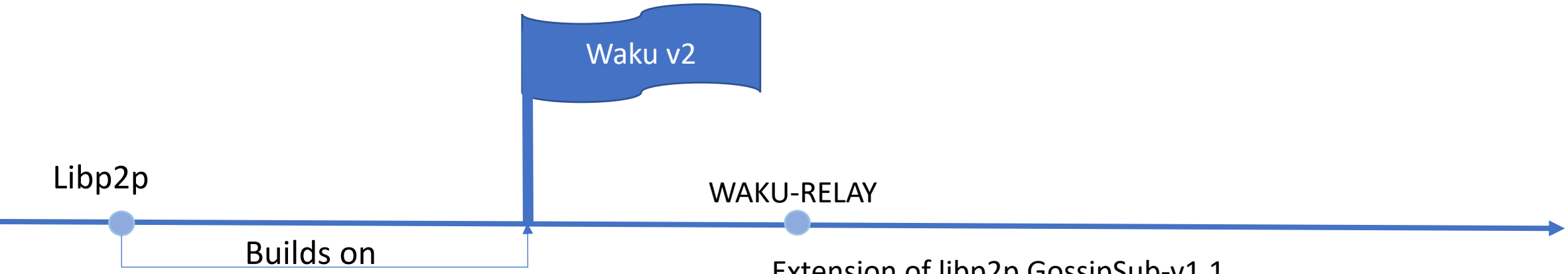
WAKU: Web3 Messaging Layer



WAKU: Web3 Messaging Layer



WAKU: Web3 Messaging Layer



Extension of libp2p GossipSub-v1.1
Follows publisher-subscriber model
Gossip-based routing

Better scalability

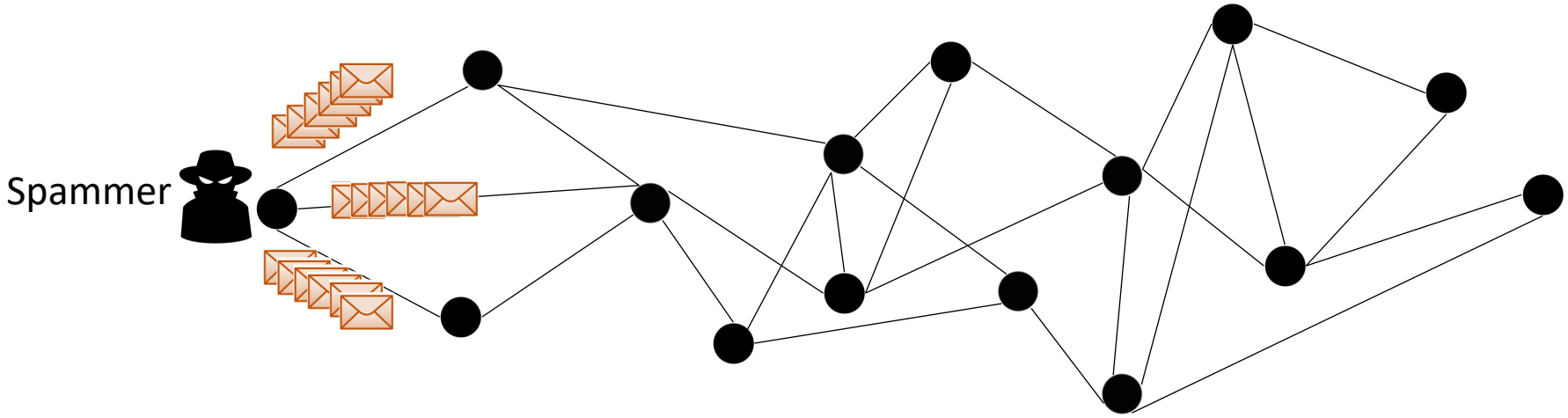
Spam issue

WAKU: Web3 Messaging Layer

Libp2

Spam issue

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

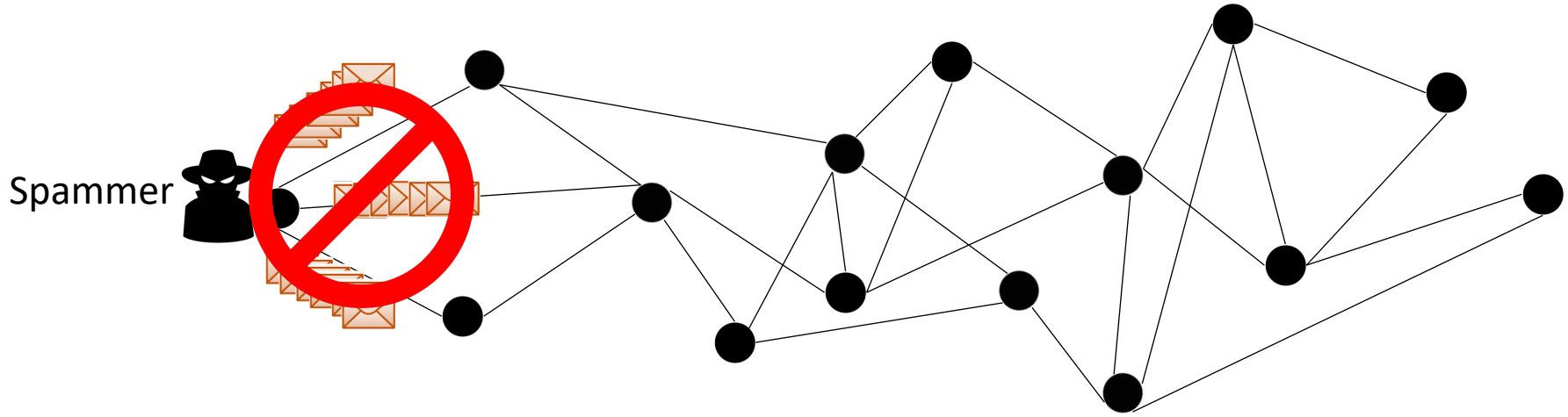


WAKU: Web3 Messaging Layer

Spam issue

Spam Protection = Controlled Messaging Rate

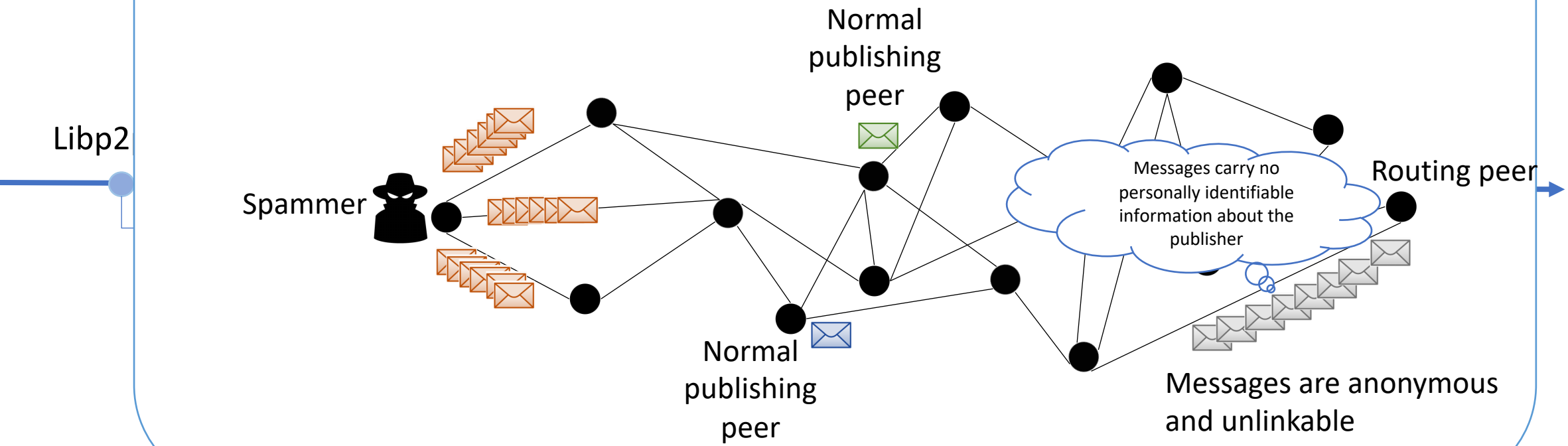
Libp2



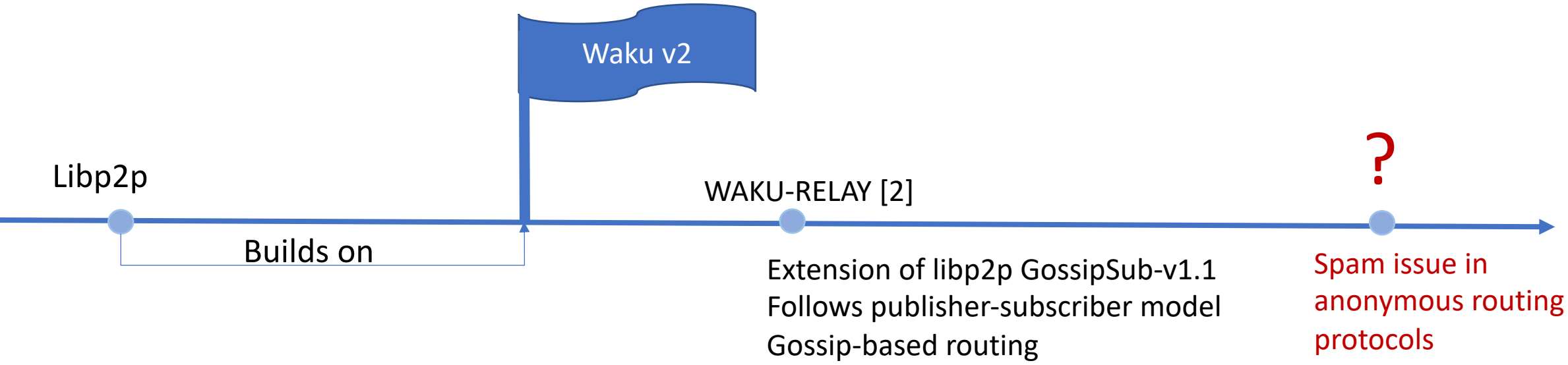
WAKU: Web3 Messaging Layer

Spam issue in anonymous routing protocols

- Messages are anonymous and not linkable to their origin.



WAKU: Web3 Messaging Layer



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: users are financially punished if they send more messages than an application-defined messaging rate.

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

RLN Primitive

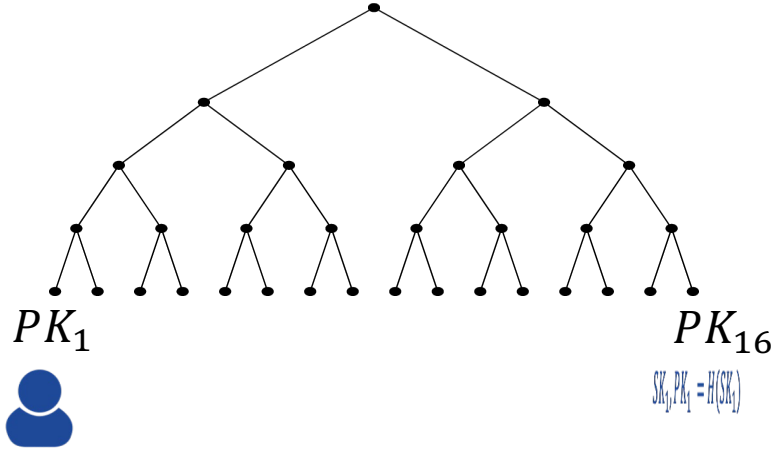
RLN Primitive

- 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

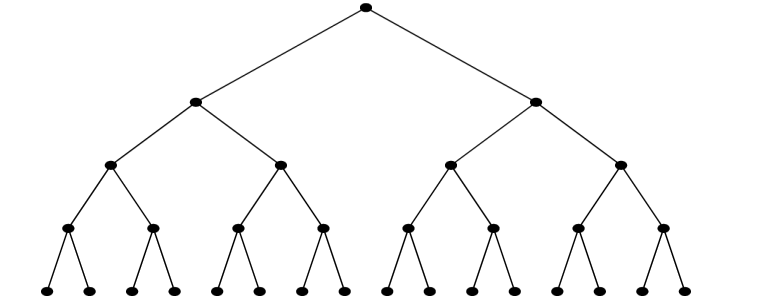


$SK_1, PK_1 = H(SK_1)$

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

RLN Primitive: Signaling

RLN membership group:
Merkle tree



PK_1 PK_{16}

$SK_1, PK_1 = H(SK_1)$

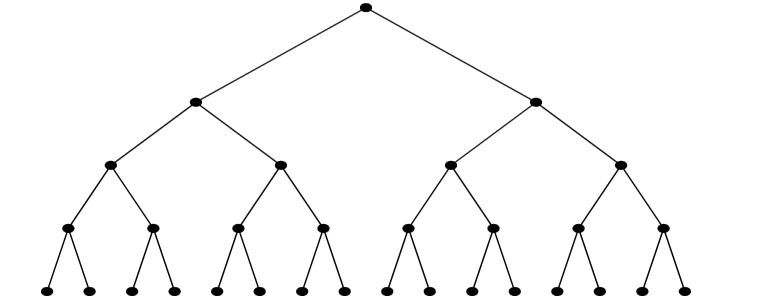
M: Message

EN: External Nullifier

IN: Internal Nullifier = $H(H(SK_1, EN))$

RLN Primitive: Signaling

RLN membership group:
Merkle tree



PK_1 PK_{16}

$SK_1, PK_1 = H(SK_1)$

Shamir Secret Sharing

M: Message

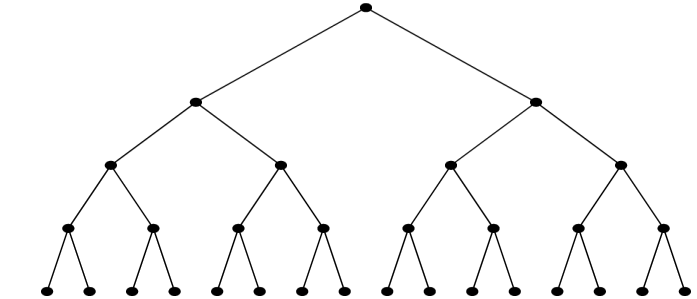
EN: External Nullifier

IN: Internal Nullifier = $H(H(SK_1, EN))$

$[SK_1]_1$: One share of SK_1

RLN Primitive: Signaling

RLN membership group:
Merkle tree



PK_1 PK_{16}

$SK_1, PK_1 = H(SK_1)$

Shamir Secret Sharing

zkSNARK

M: Message

EN: External Nullifier

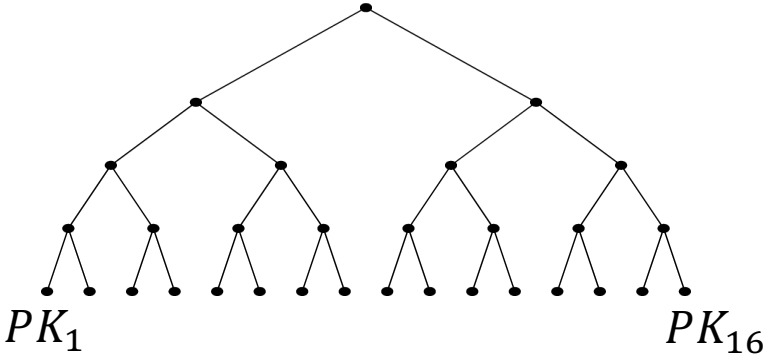
IN: Internal Nullifier = $H(H(SK_1, EN))$

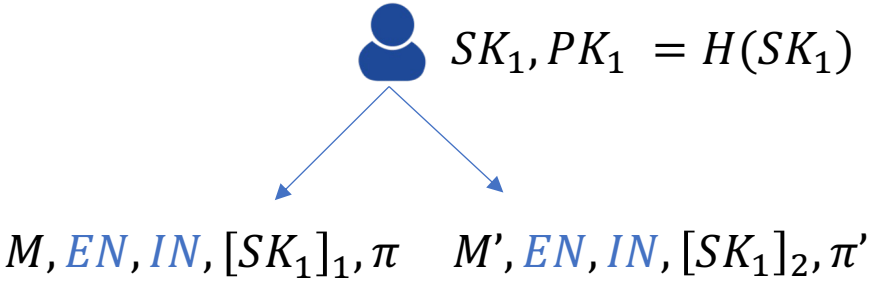
$[SK_1]_1$: One share of SK_1

Zero-Knowledge Proof for
- Membership
- Correctness of $[SK_1]_1$
- Correctness of IN

RLN Primitive: Detecting Double Signaling

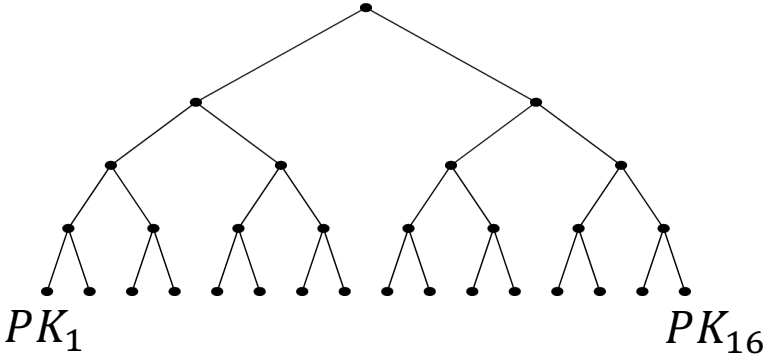
RLN membership group:
Merkle tree

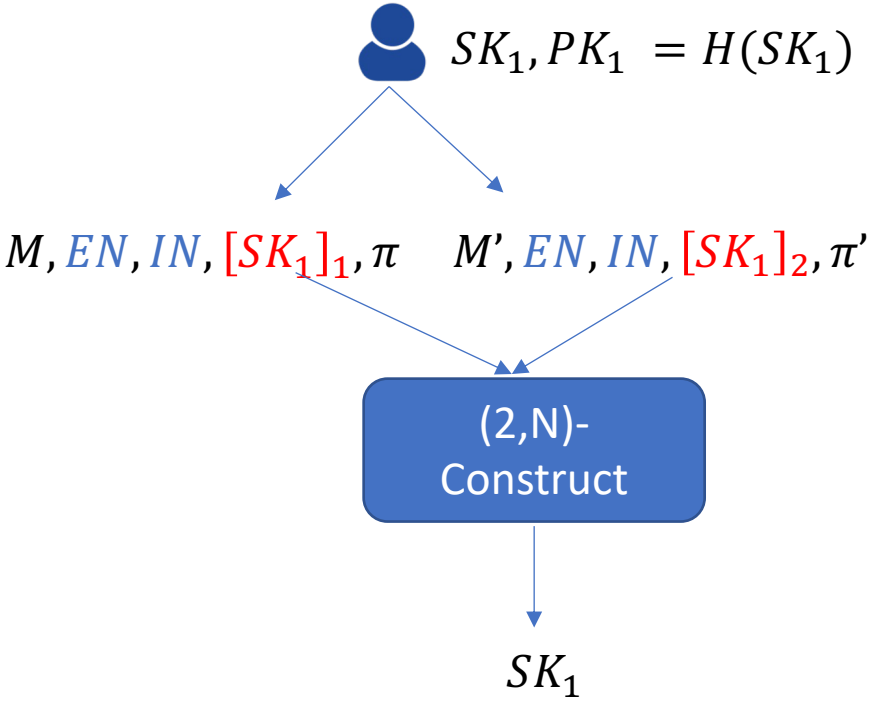




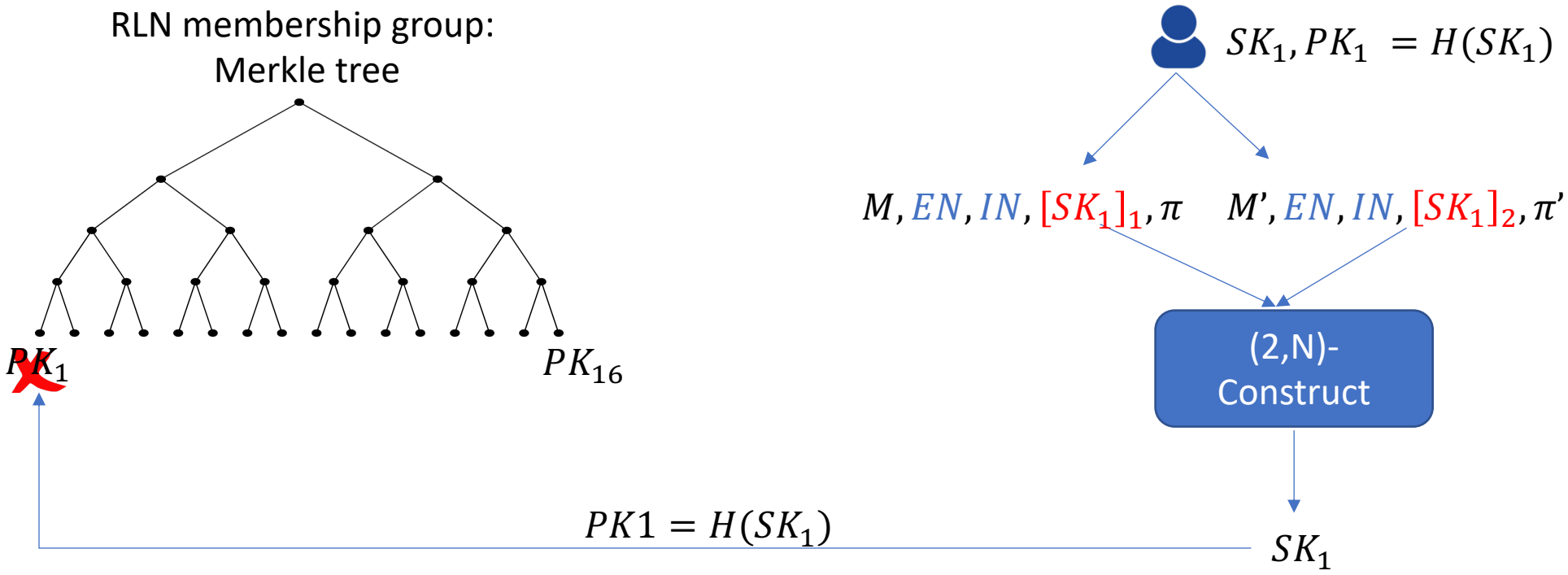
RLN Primitive: Slashing

RLN membership group:
Merkle tree





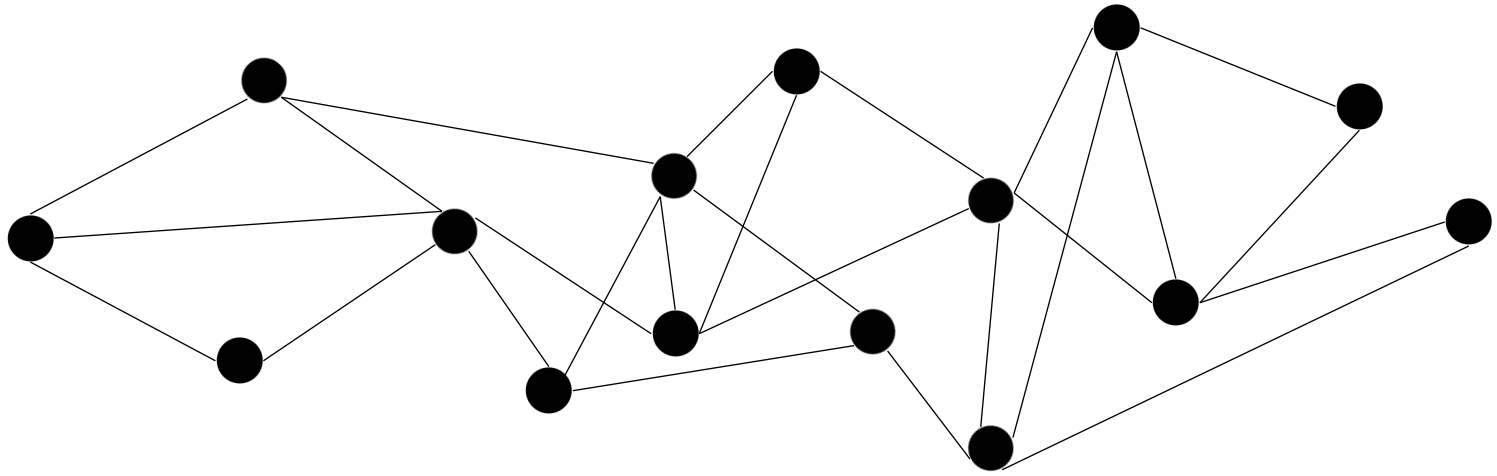
RLN Primitive: Slashing



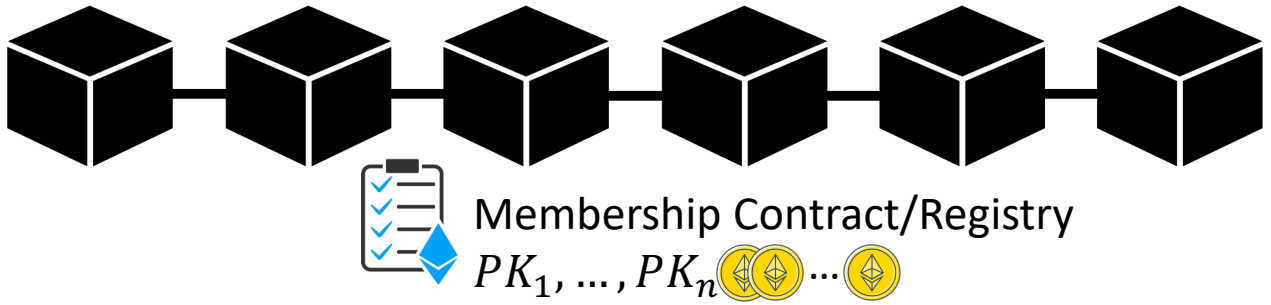
WAKU-RLN-RELAY Construction

WAKU-RLN-RELAY: RLN Group

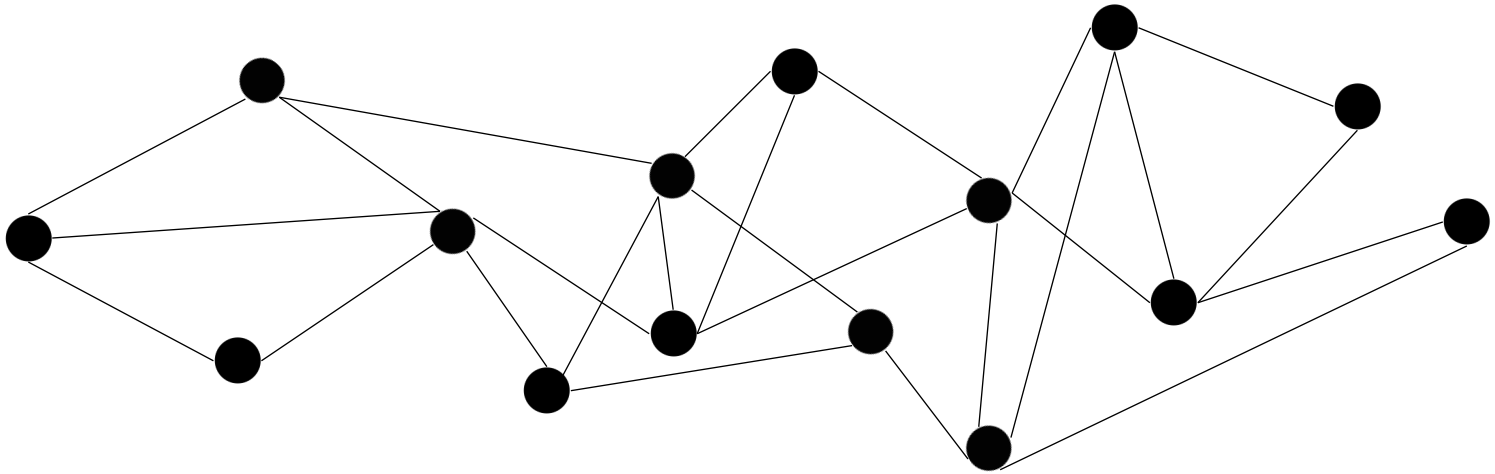
RLN group = Peers subscribed to the same topic e.g., WAKU-RLN-RELAY



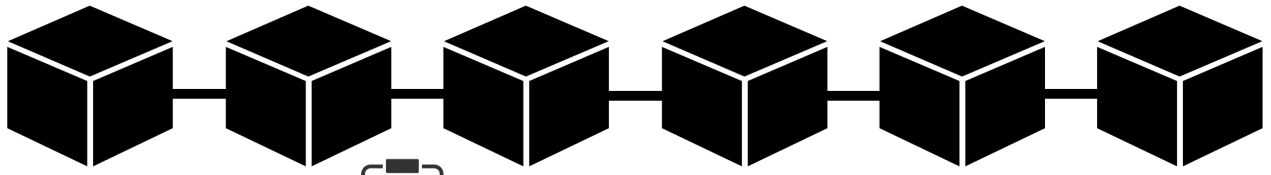
WAKU-RLN-RELAY: Registration



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



WAKU-RLN-RELAY: Registration

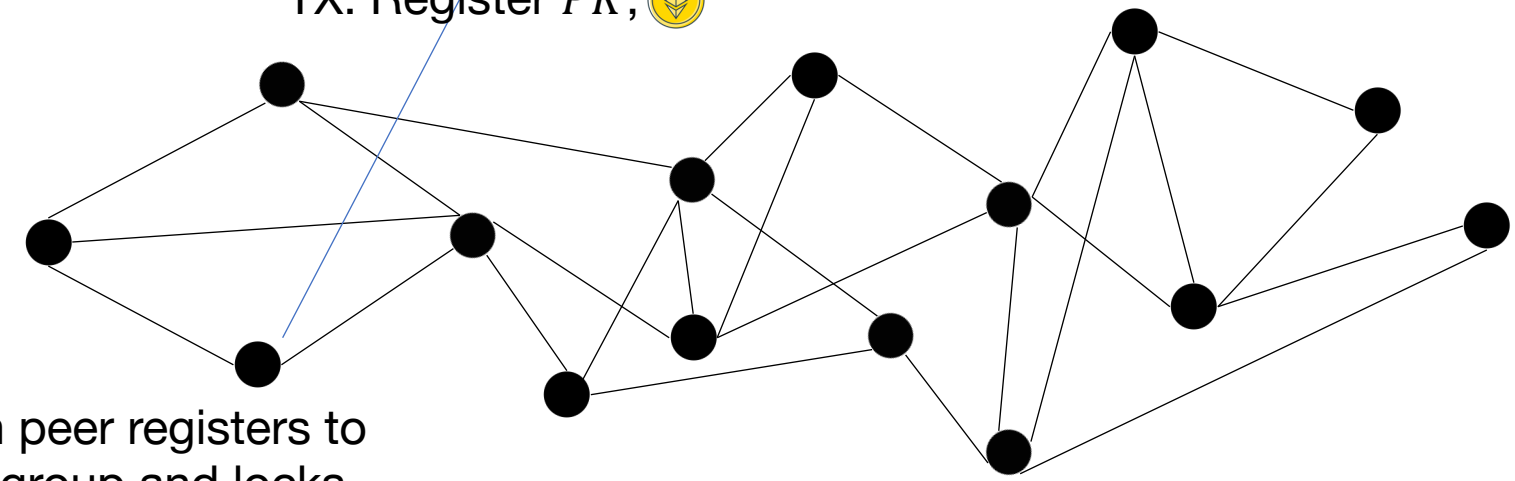


Membership Contract/Registry

$PK_1, \ldots, PK_n$ ...

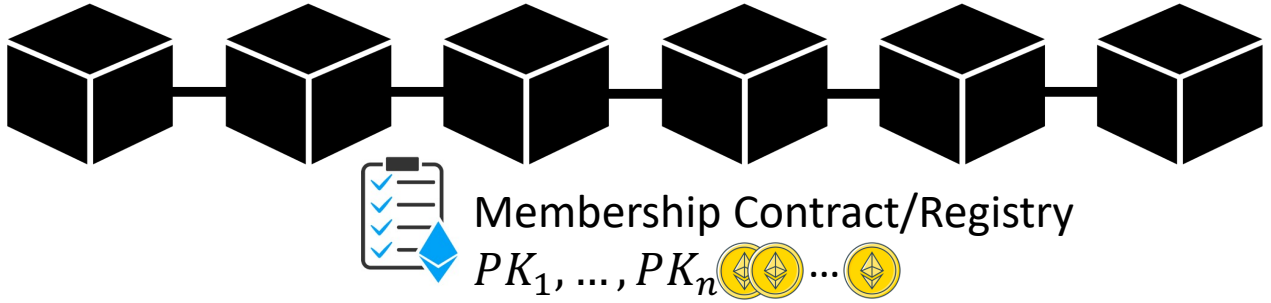
TX: Register PK,

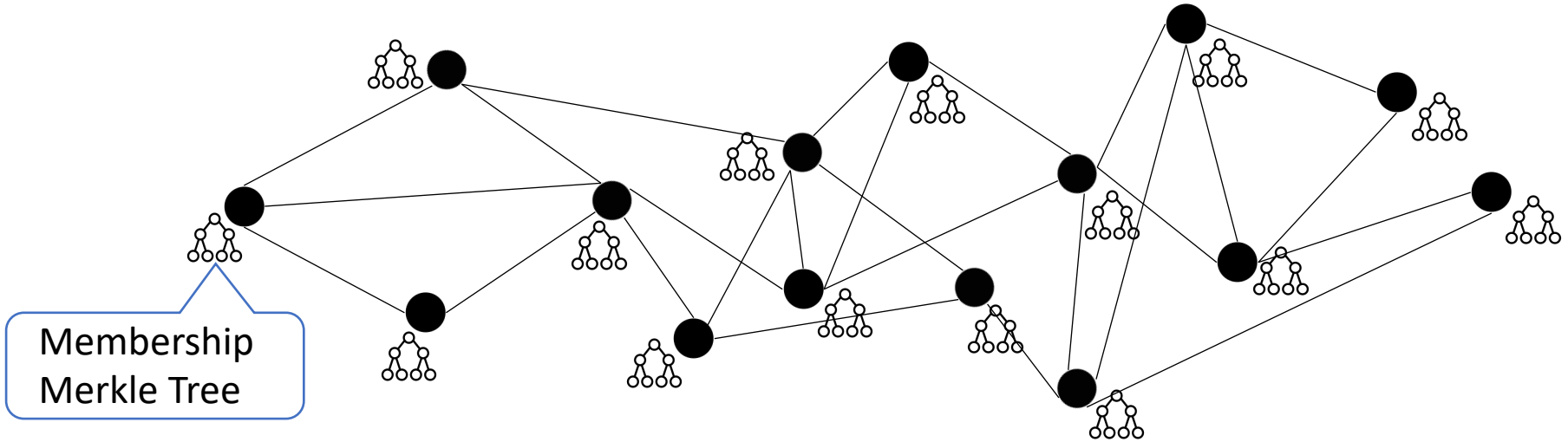
RLN group = Peers subscribed to the same topic e.g., WAKU-RLN-RELAY



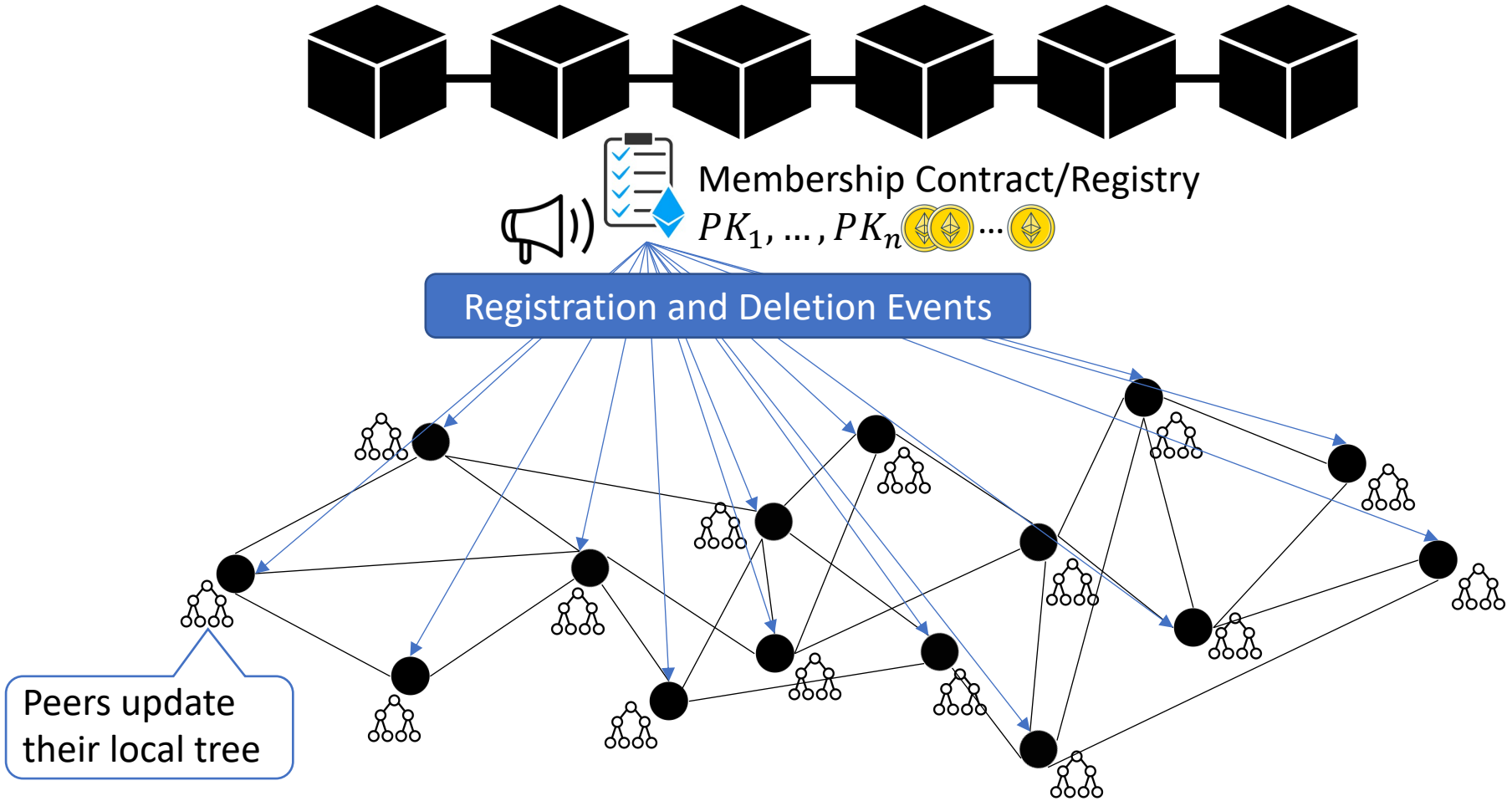
Each peer registers to the group and locks some funds

WAKU-RLN-RELAY: Registration



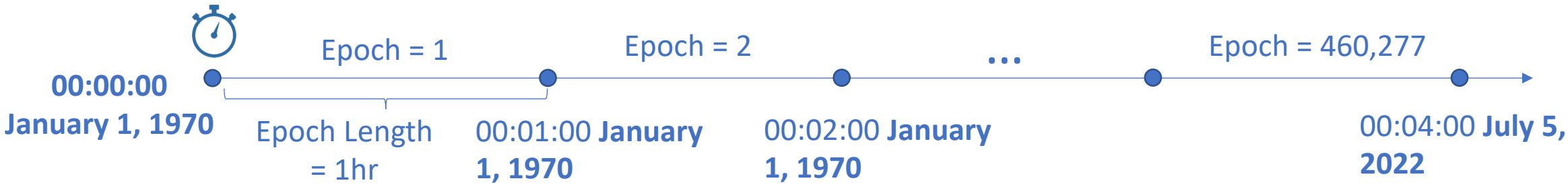


WAKU-RLN-RELAY: Registration



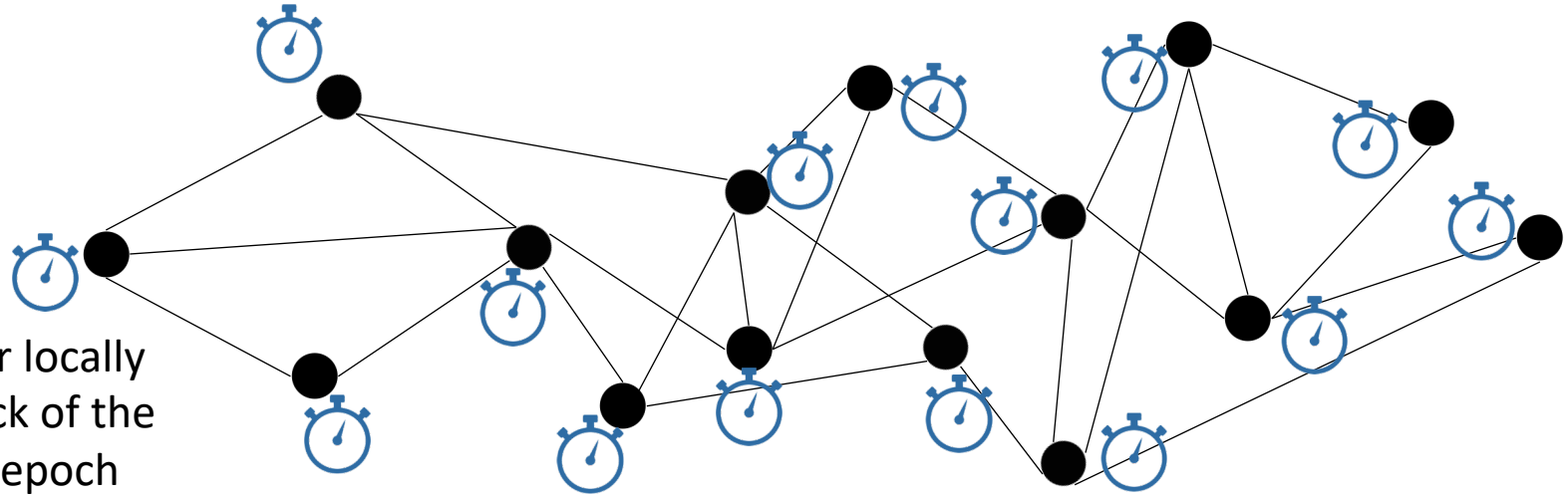
WAKU-RLN-RELAY: Messaging Rate

- 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

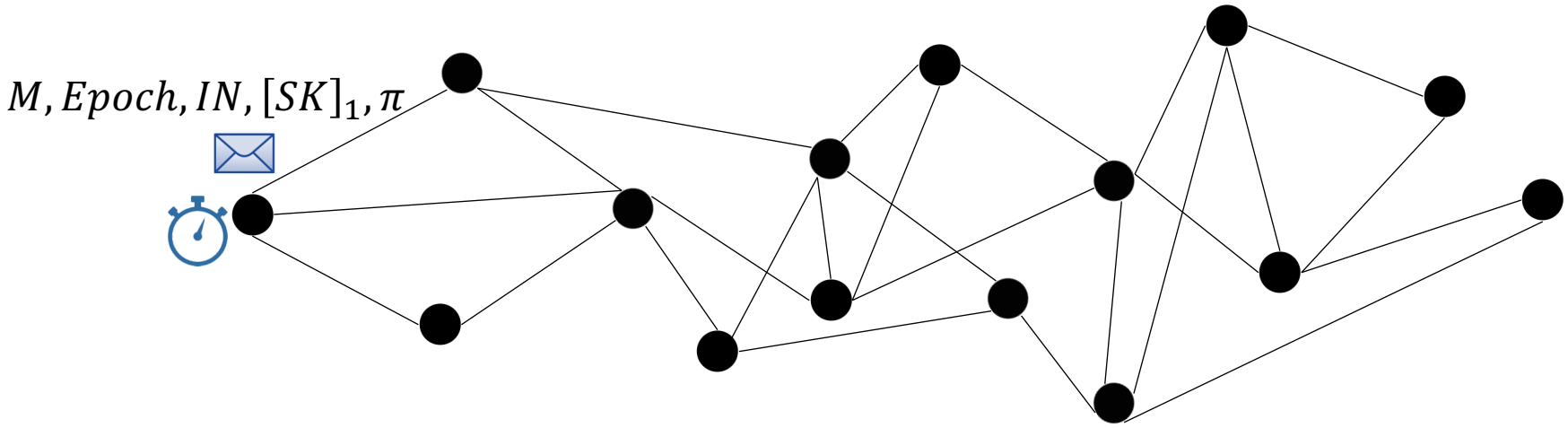
- Peers keep track of the current epoch count.



Each peer locally keeps track of the current epoch

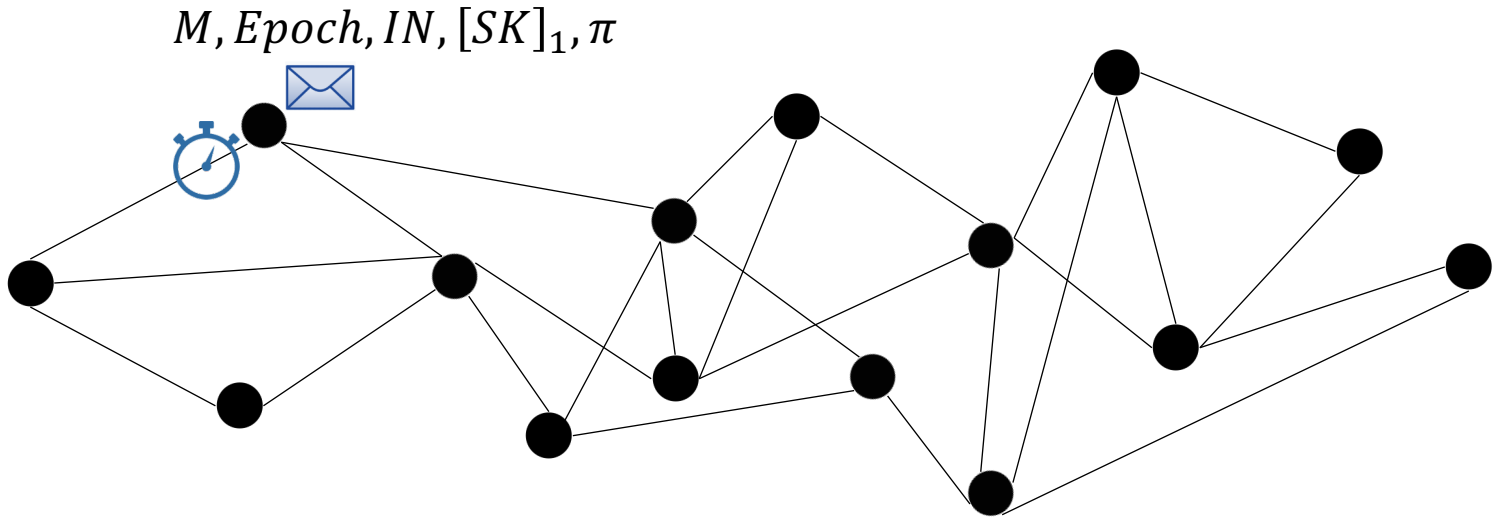
WAKU-RLN-RELAY: Publishing

- 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.

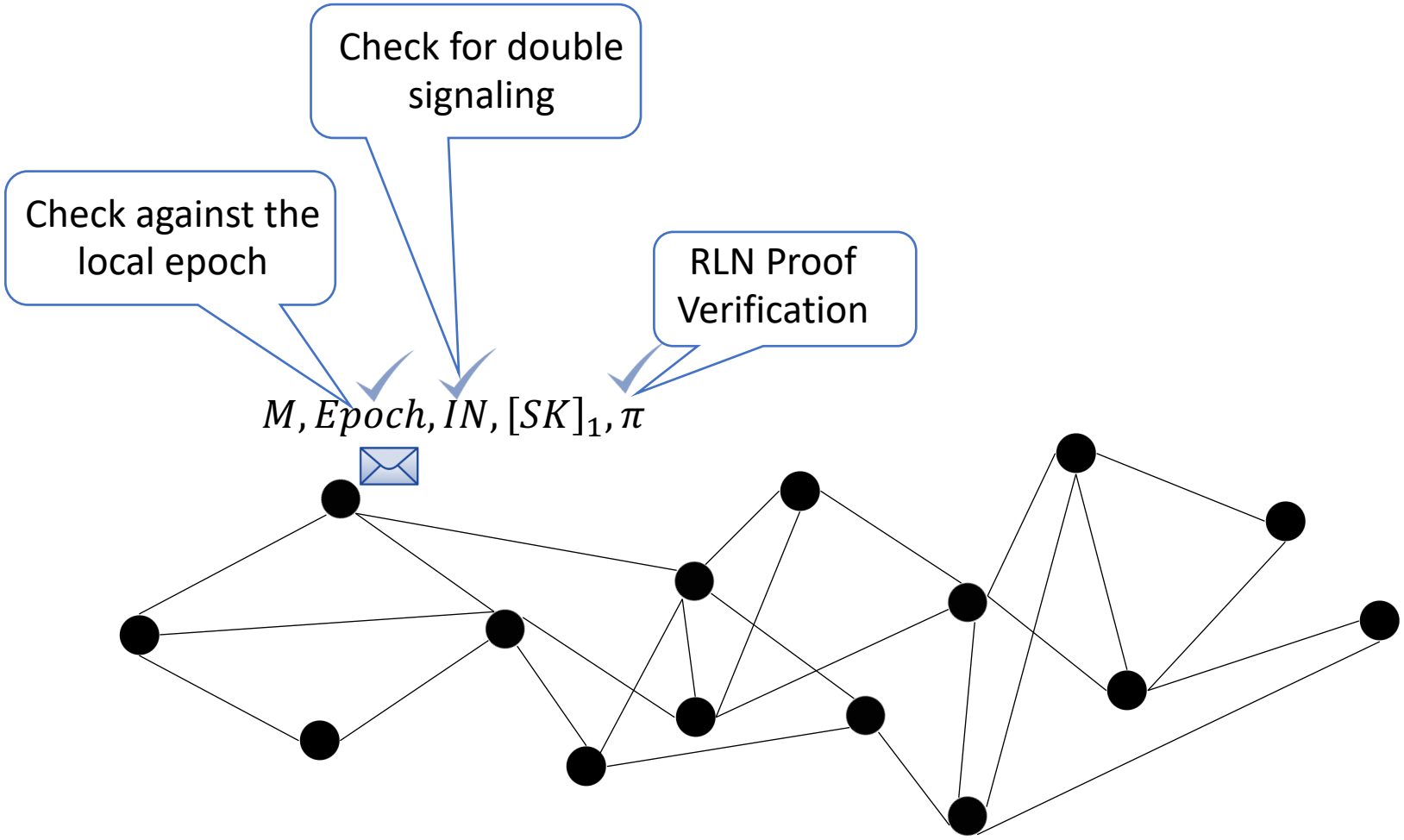


WAKU-RLN-RELAY: Routing

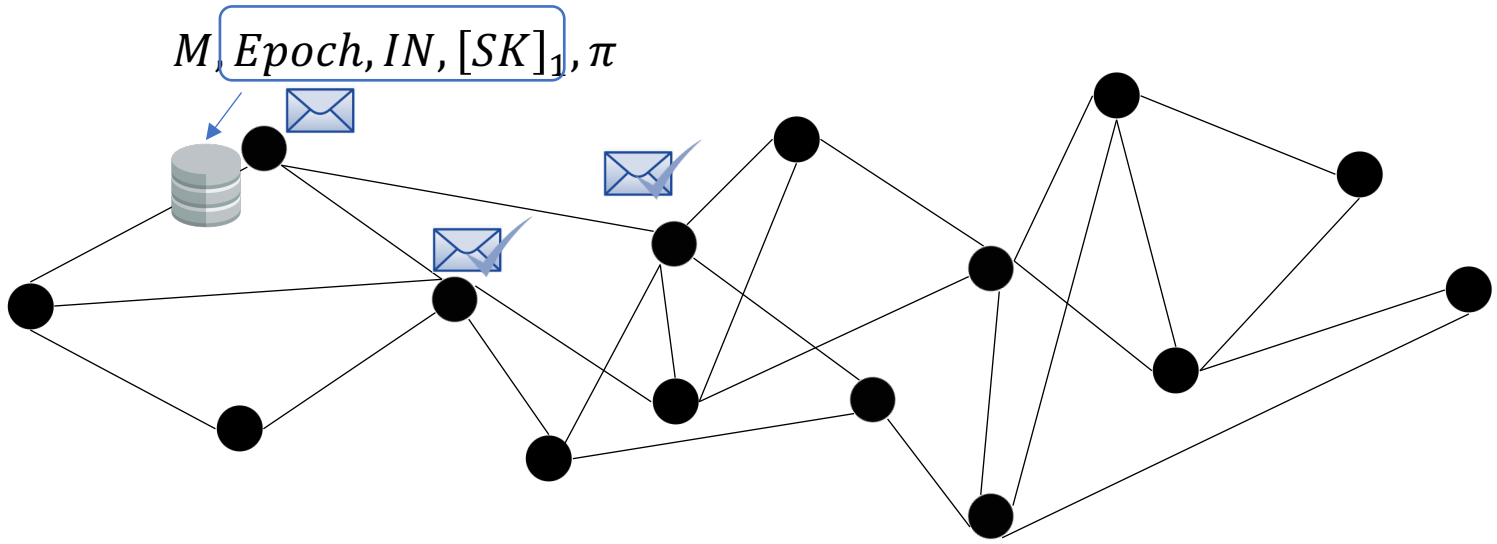
- Regular Gossip-based routing protocol + RLN signal verification (installed as a libp2p GossipSub topic validator).



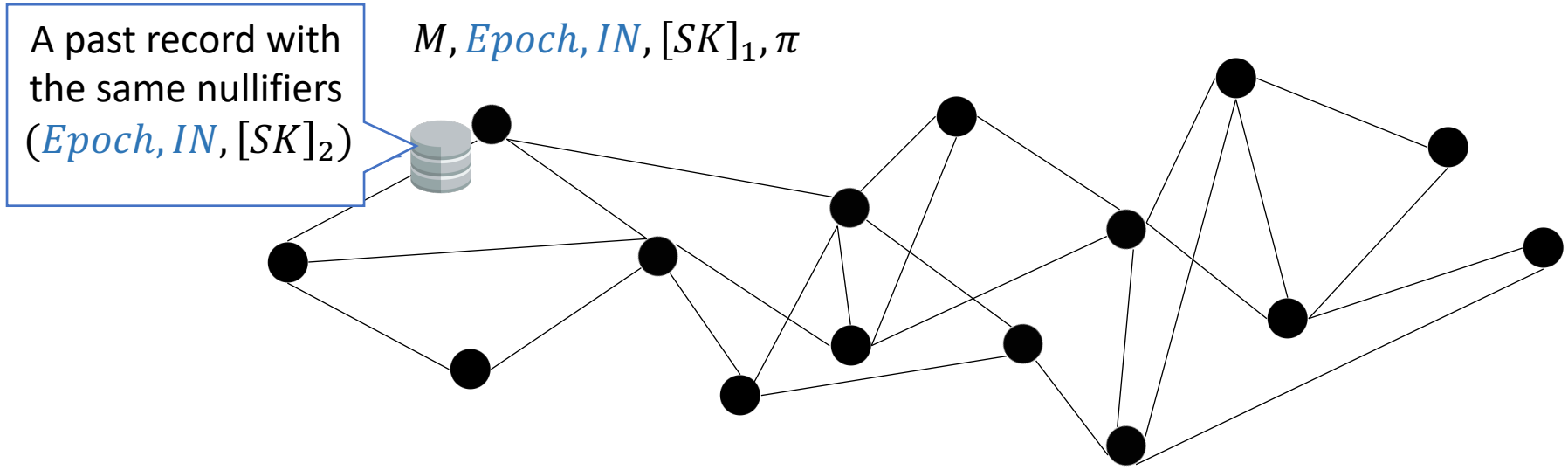
WAKU-RLN-RELAY: Routing



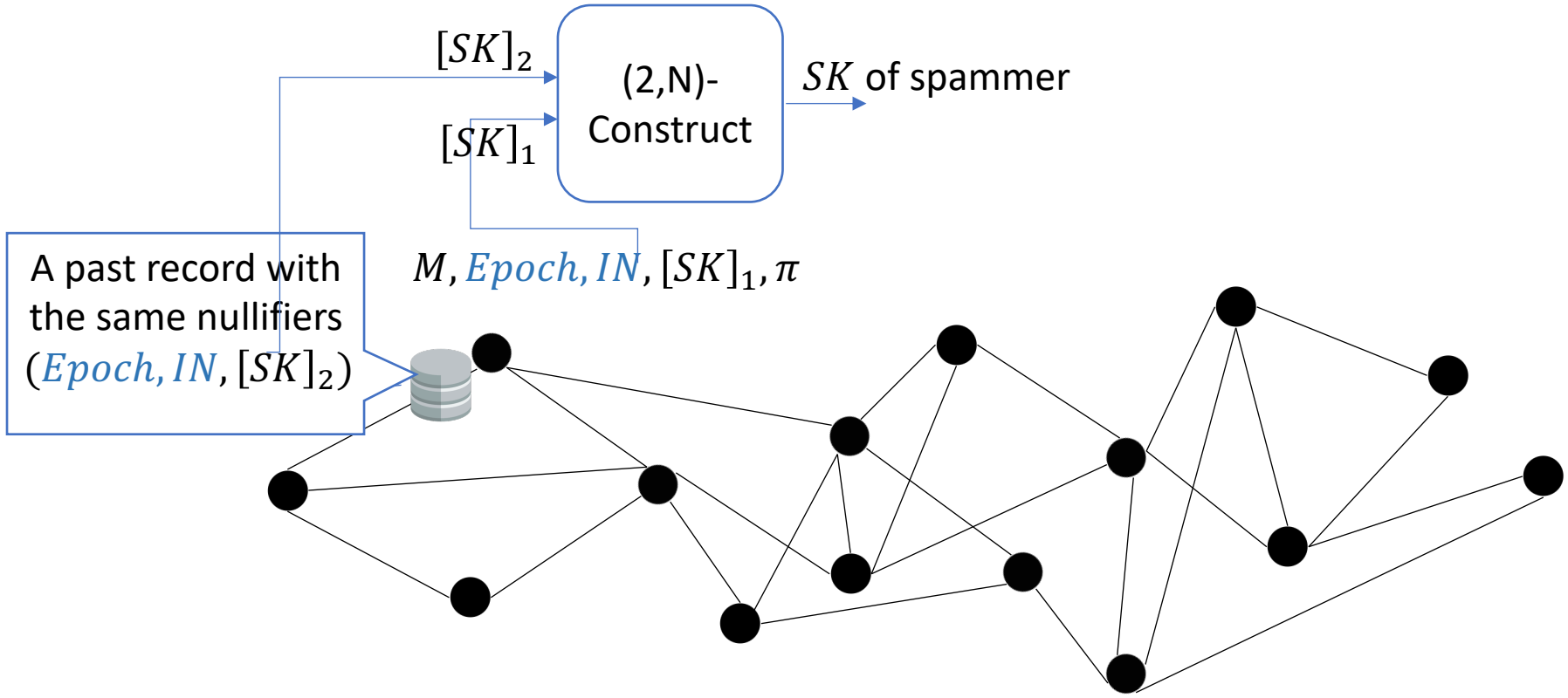
WAKU-RLN-RELAY: Routing



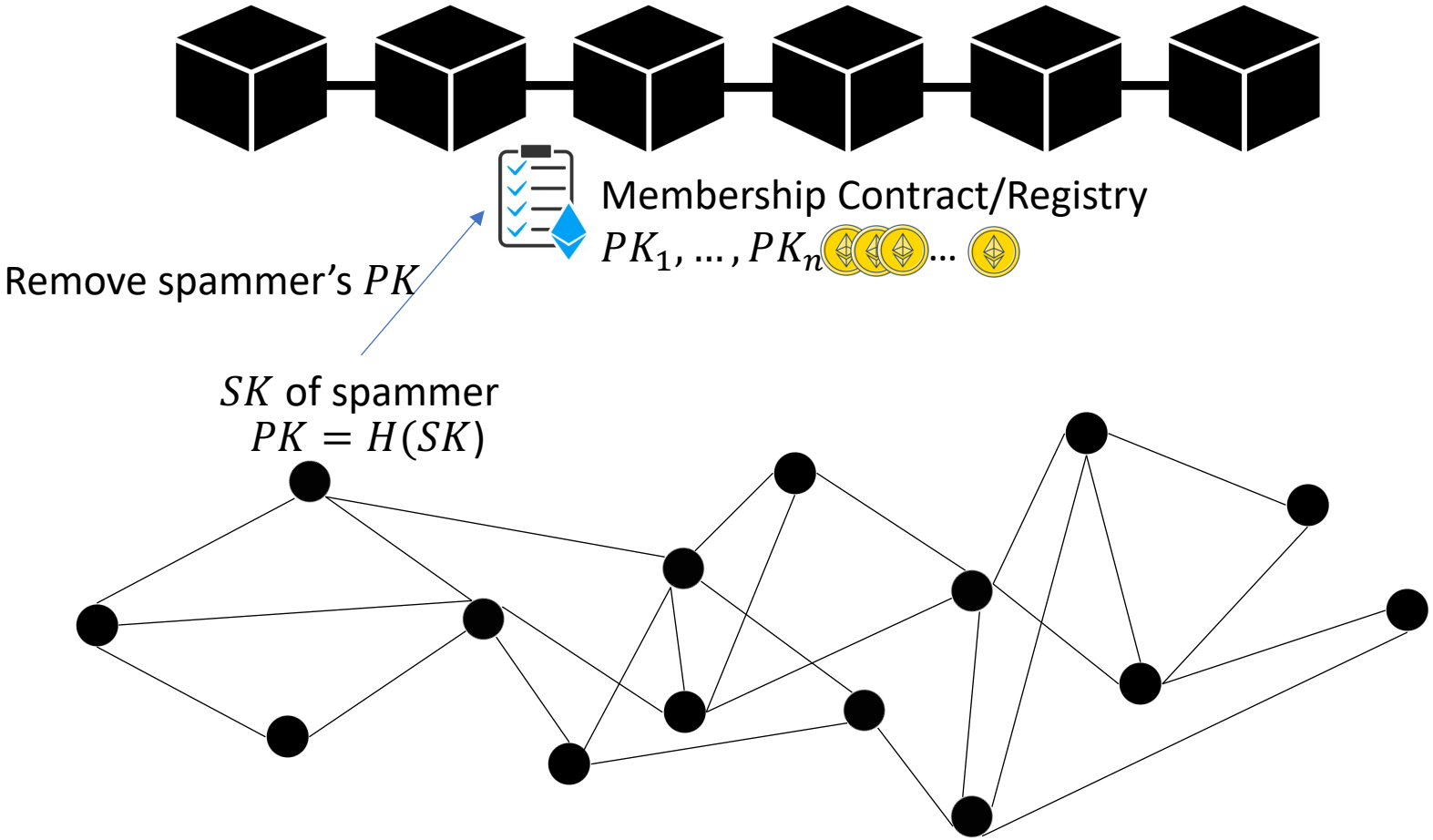
WAKU-RLN-RELAY: Slashing



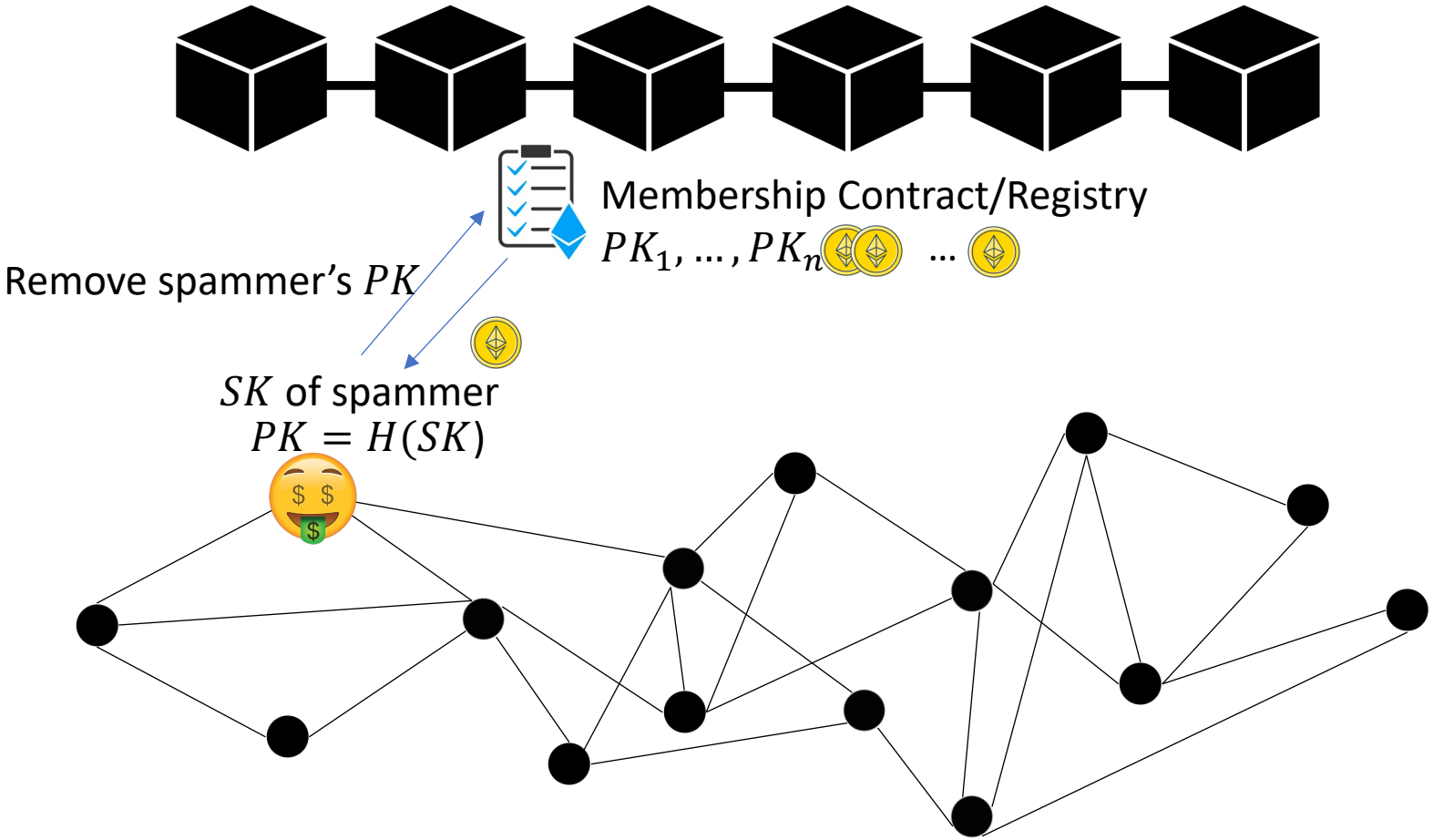
WAKU-RLN-RELAY: Slashing



WAKU-RLN-RELAY: Slashing



WAKU-RLN-RELAY: Slashing



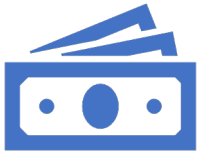
WAKU-RLN-RELAY

Global Spam Protection

Spammers are detected and removed from the network instantly

Privacy-Preservation

Peers identity remain hidden



Economic Incentives

Financial punishment for the spammers and a financial reward for those who catch spammers

WAKU-RLN-RELAY Proof-Of-Concept

- The PoC of WAKU-RLN-RELAY protocol is implemented in Nim and is available in the nwaku repository:

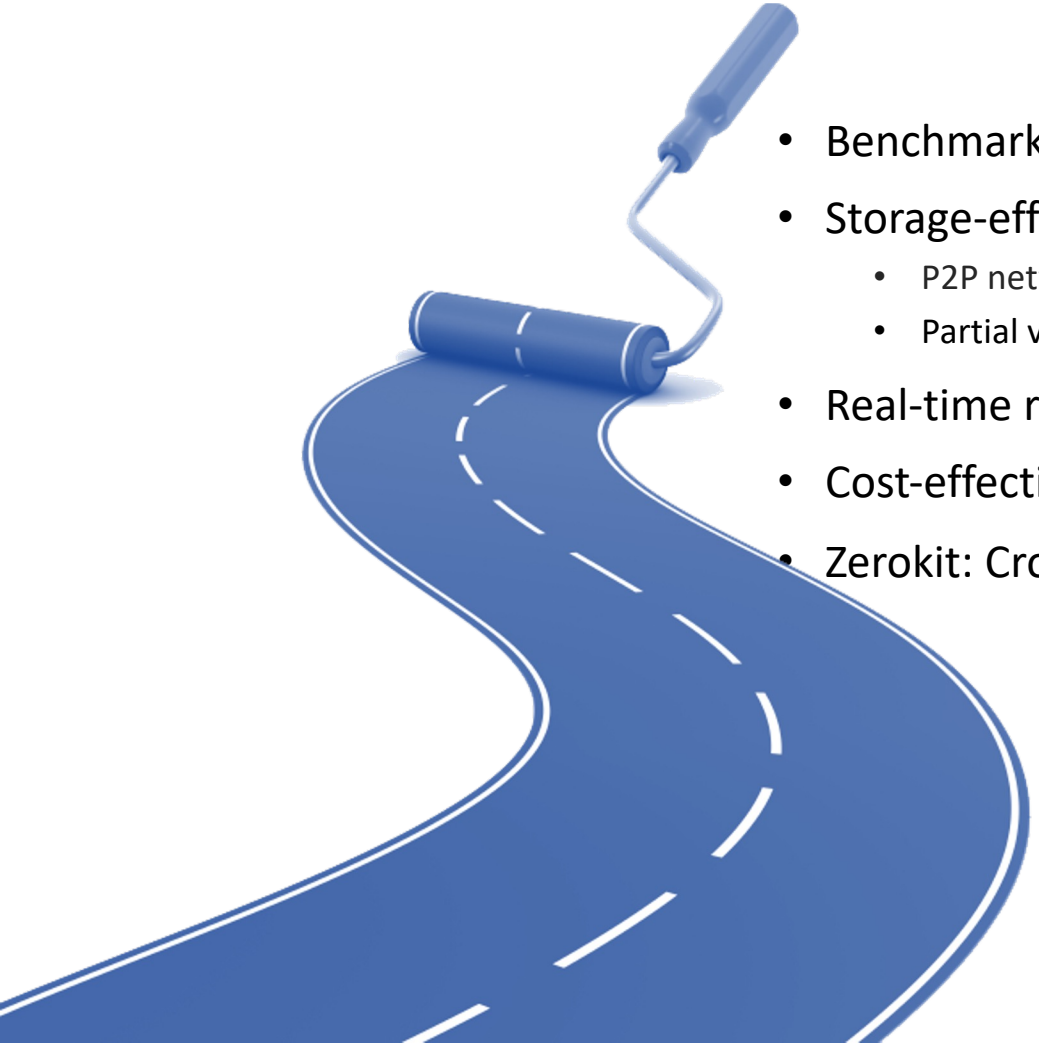
- A PoC rate-limited/spam-protected command-line chat application using WAKU-RLN-RELAY:

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
- Zerokit: Cross-client interoperability, ability to run in browsers and in nim

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: $\left\lceil \frac{message\ propagation\ delay + clock\ asynchrony}{epoch\ length} \right\rceil$
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 $\boldsymbol{O(d)}$ **hashing** operations (where $\boldsymbol{d = 20}$) to calculate the tree root and the authentication path.
- **Bootstrapping** takes $\boldsymbol{O(2^d)}$ **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

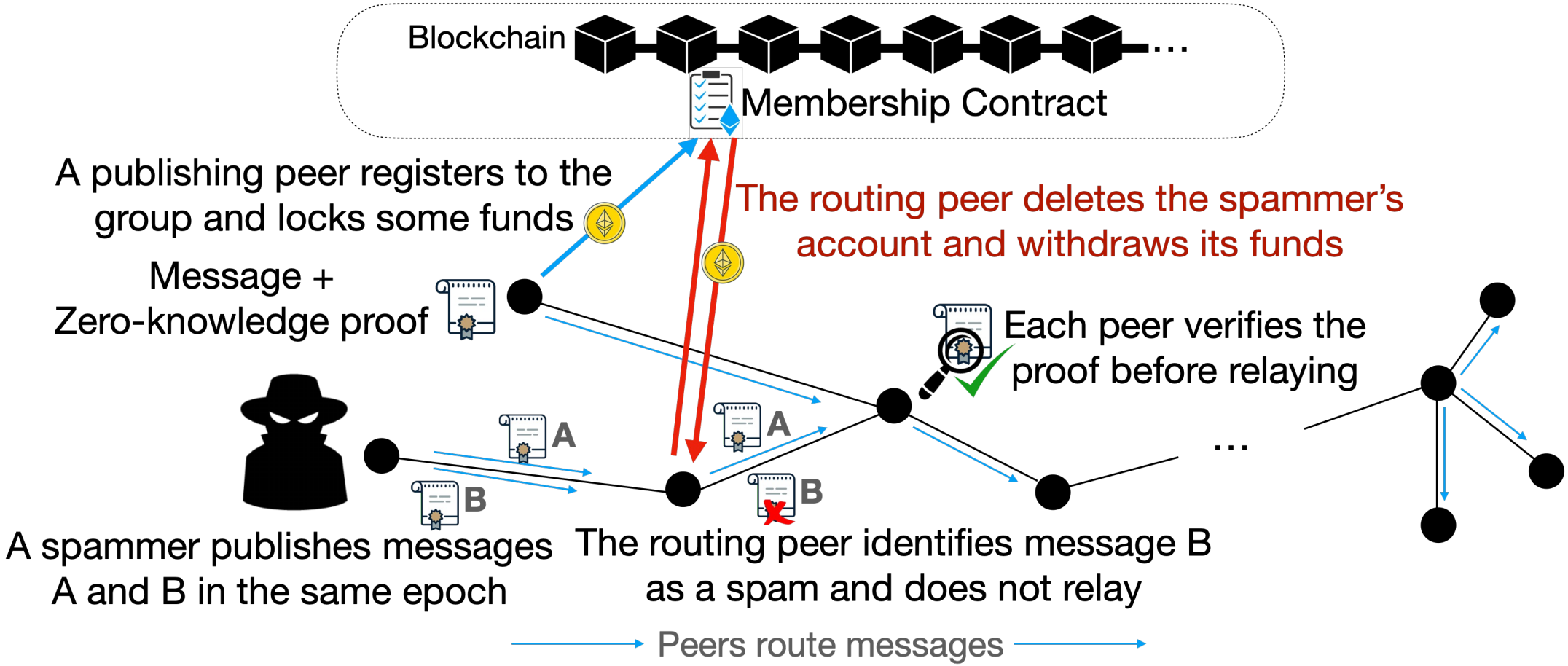
- The **Merkle tree** with **depth 20** takes up **~67MB** storage. With some optimizations in can be reduced to the order of tens of KBs.
- **Identity keys and identity commitment** keys are each of size **32 Bytes**.
- **Prover key** size is approximately **~3.24 MB**.
- **Nullifier map** consists 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 ~ 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


Blockchain
Membership Contract
A publishing peer registers to the group and locks some funds
Message + Zero-knowledge proof
The routing peer deletes the spammer's account and withdraws its funds
Each peer verifies the proof before relaying
A
A
B
B
A spammer publishes messages A and B in the same epoch
The routing peer identifies message B as a spam and does not relay
Peers route messages