Distributed Ledger Technologies Workshop (**DLT 2025**)

Efficient Query Verification for Blockchain Superlight Clients Using SNARKs

Stefano De Angelis
Nethermind

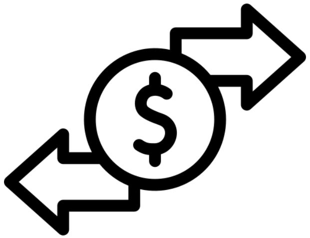
Ivan Visconti
Sapienza University of Rome

Andrea Vitaletti
Sapienza University of Rome

Marco Zecchini
Sapienza University of Rome

Contact me at:
zecchini@diag.uniroma1.it

June 12, 2025



Transactions

Transactions

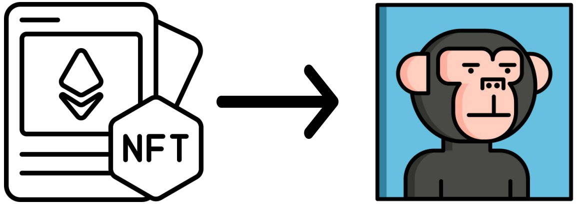
Defi

Transactions

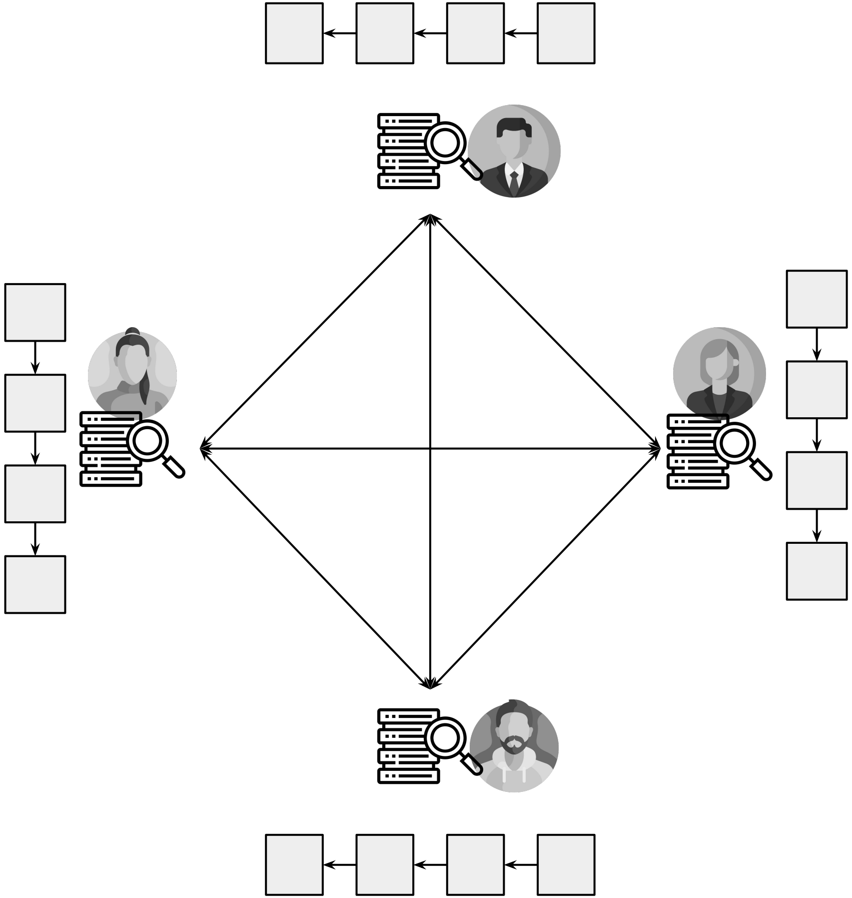
Defi

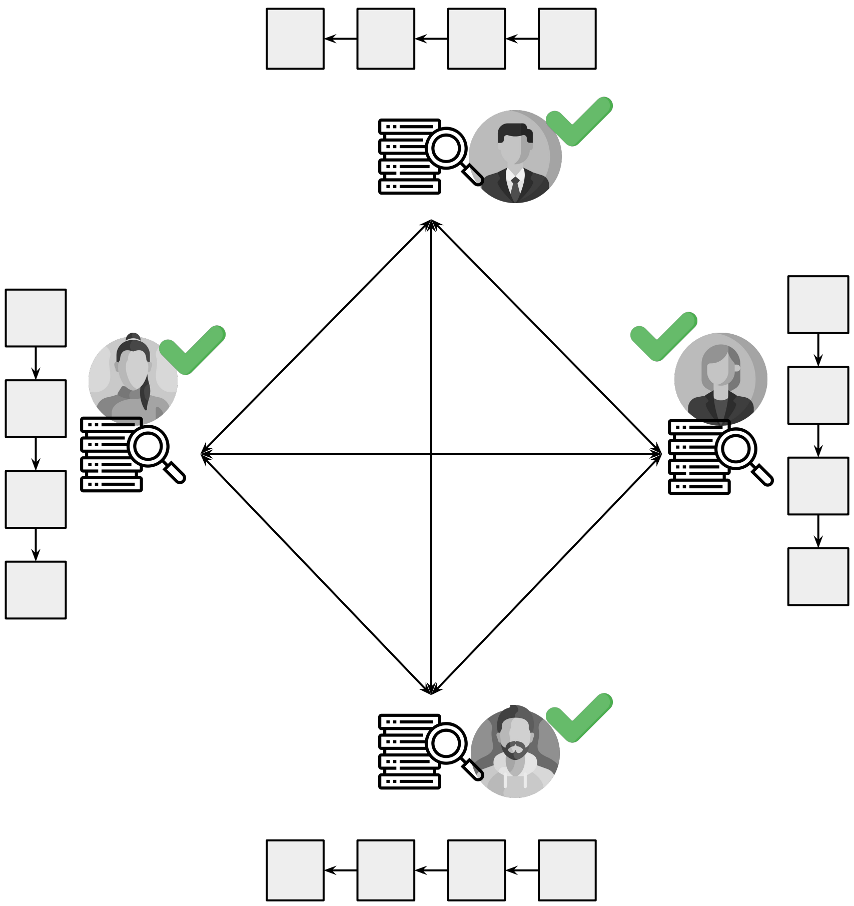
Auction

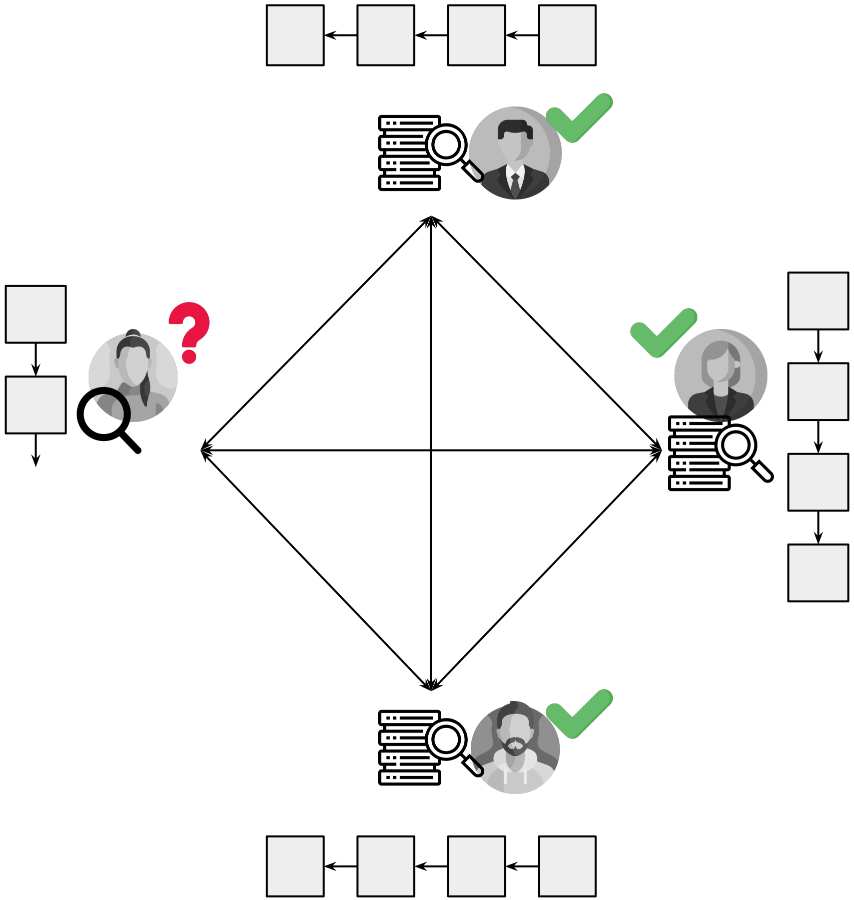
E-voting



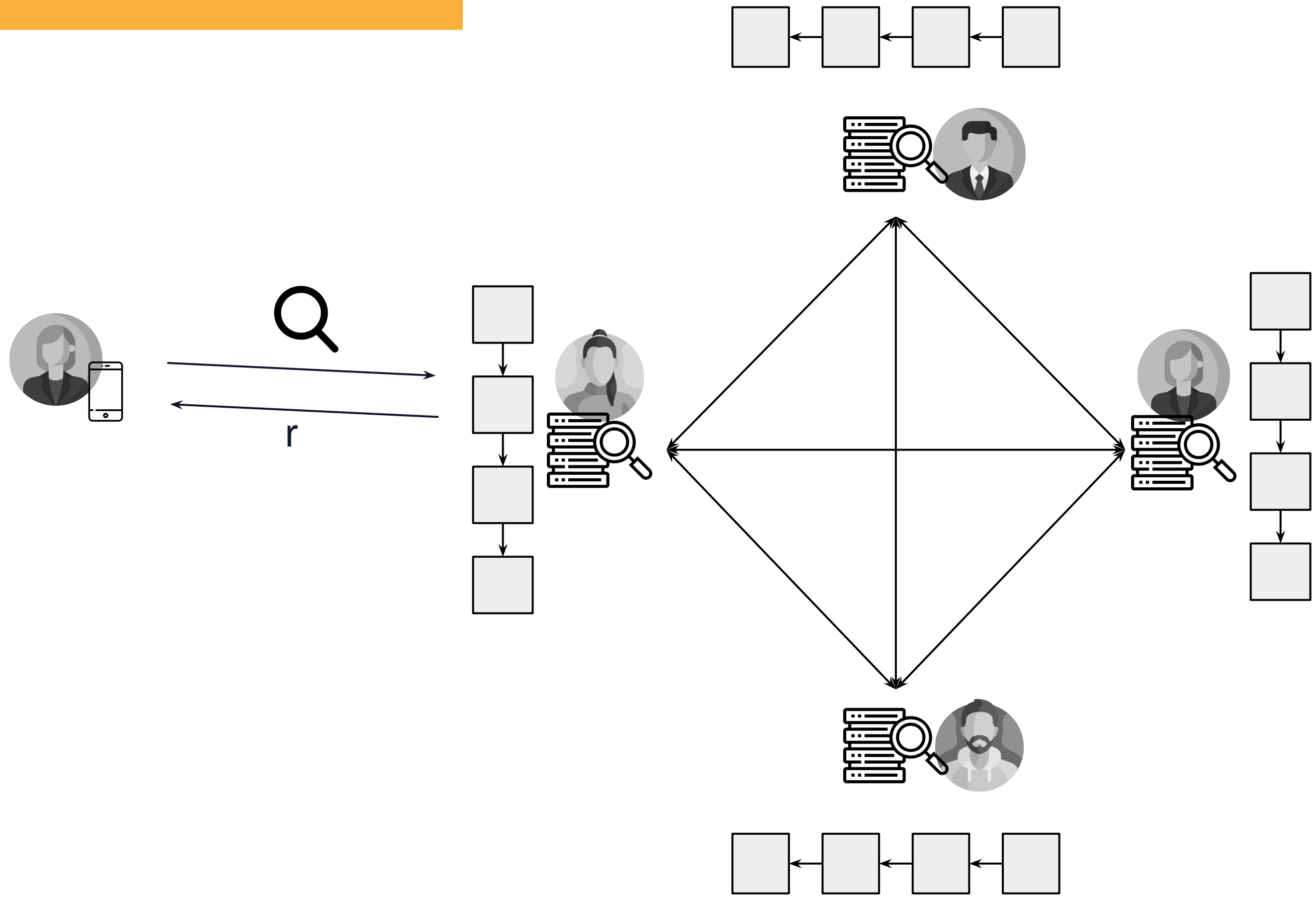
NFT management



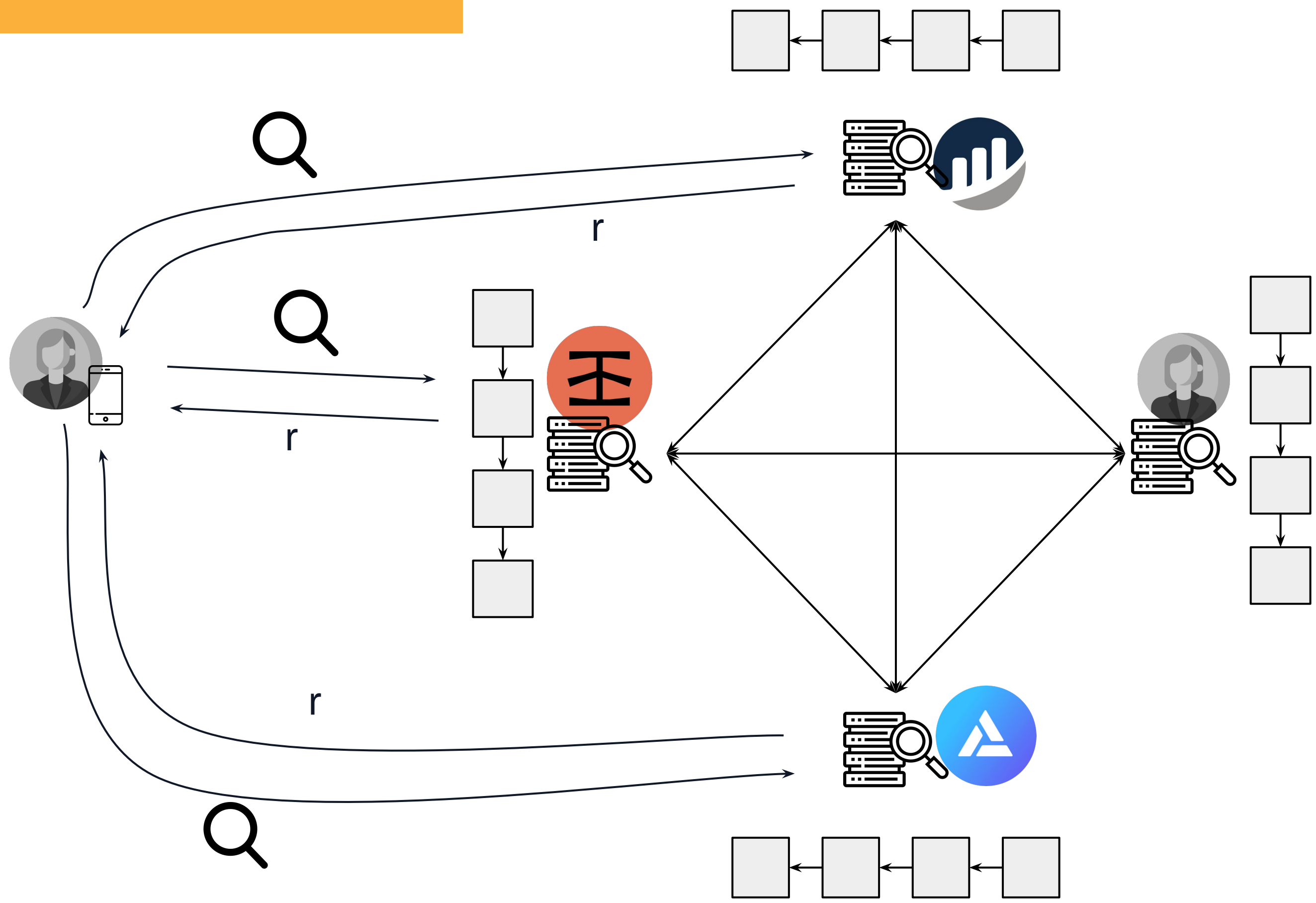




RPC **Model**

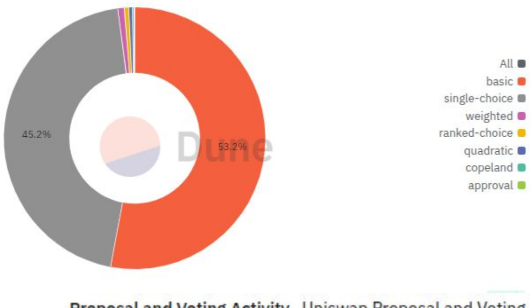


RPC **Model**

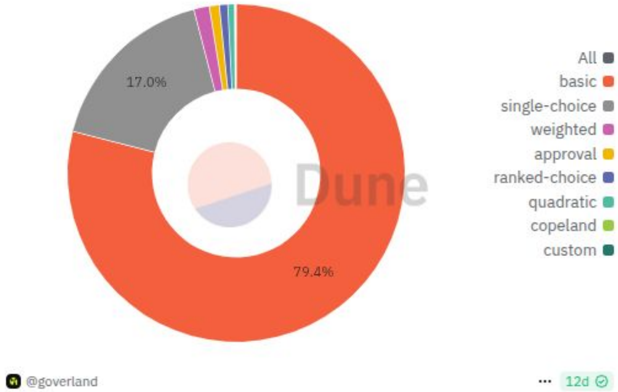


Applications might require **specific queries**

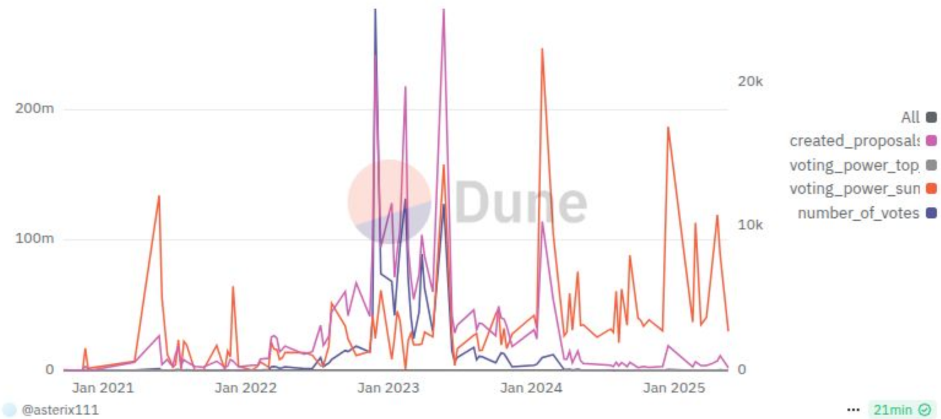
Voting systems popularity by VOTES amount the DAOs who started activity during the last month



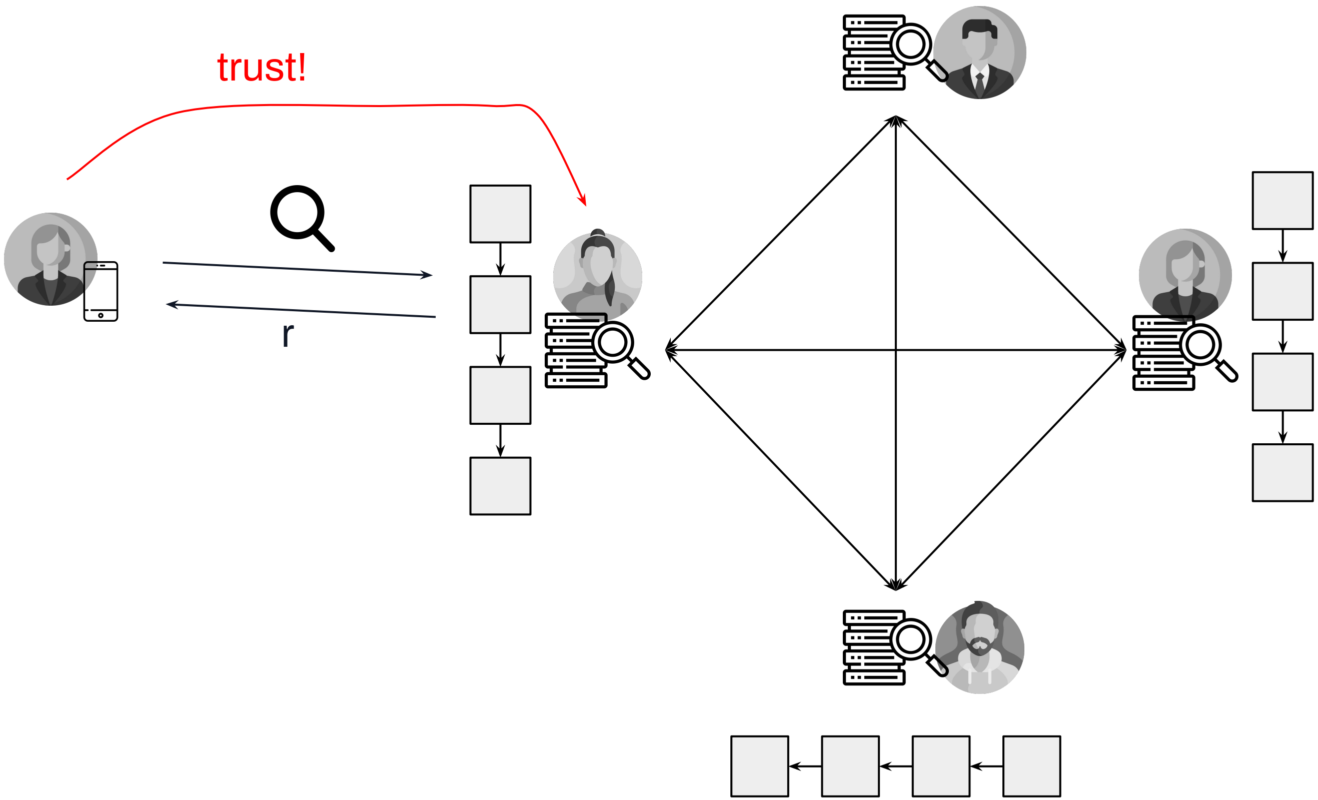
Number of votes per voting system Voting system populatiry across the whole snapshot data



Proposal and Voting Activity Uniswap Proposal and Voting Activity

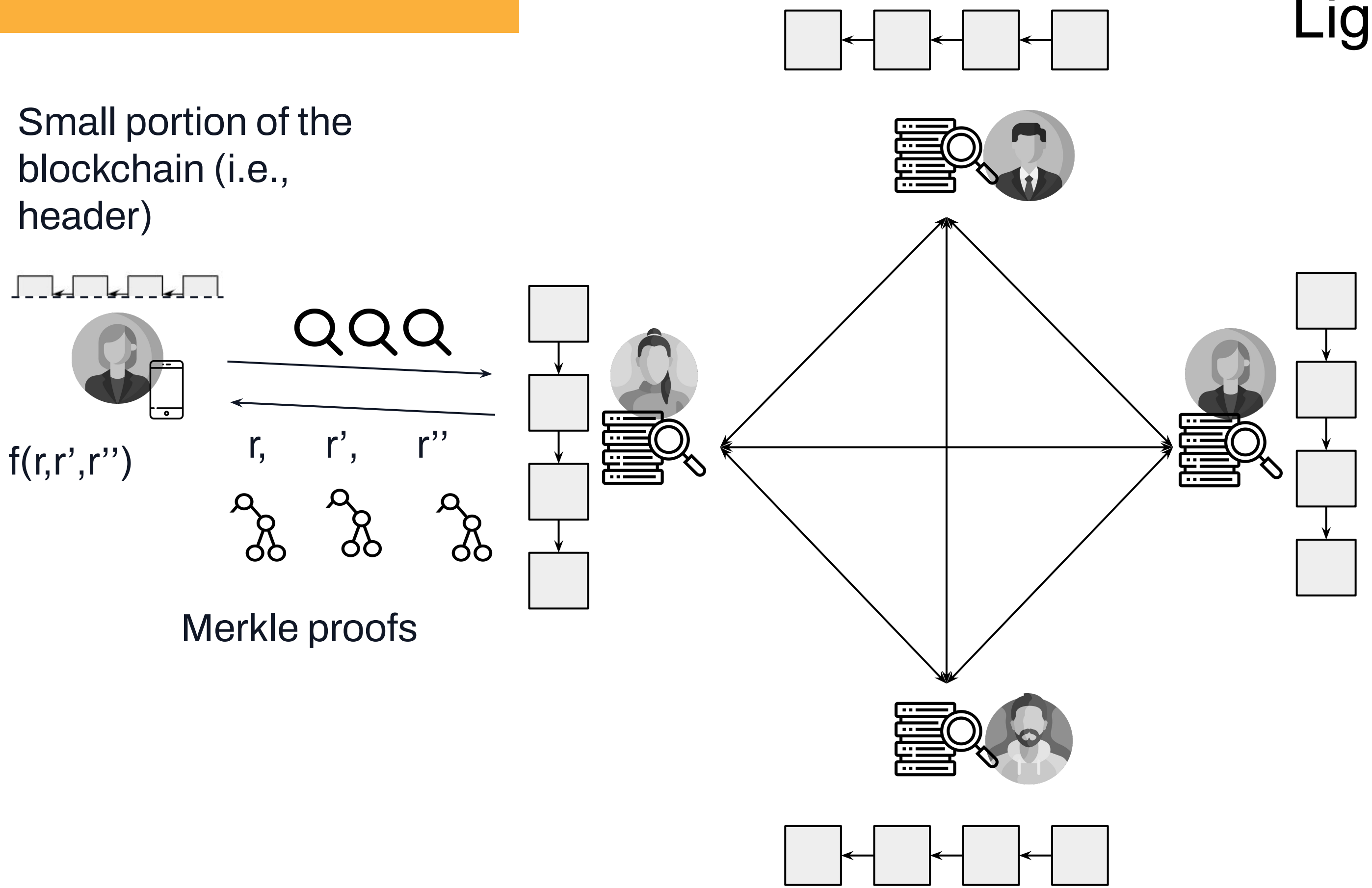


~~RPC Model~~



Light Client **Model**

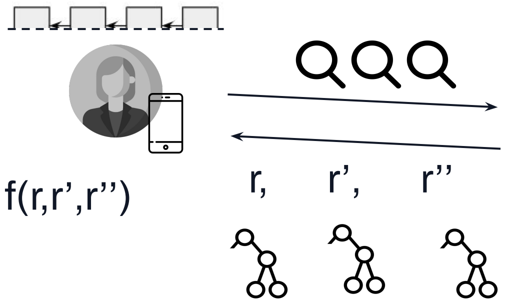
Small portion of the blockchain (i.e., header)

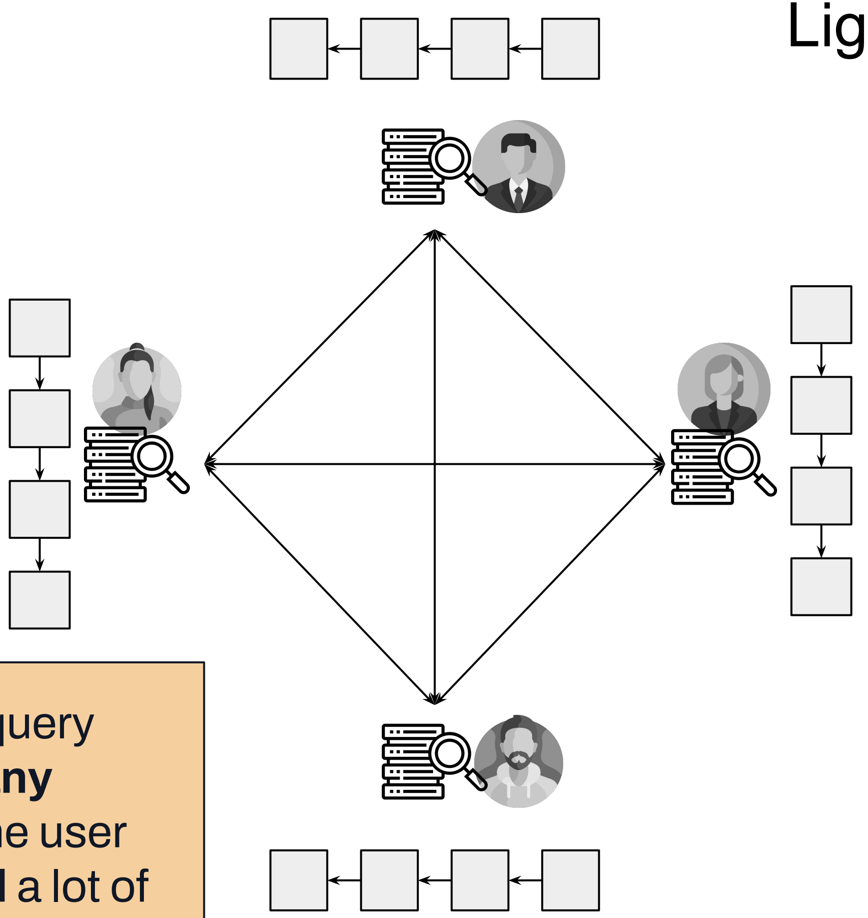


[Nak 2009] S. Nakamoto, “**Bitcoin: A peer-to-peer electronic cash system**” - 2009

Light Client Model

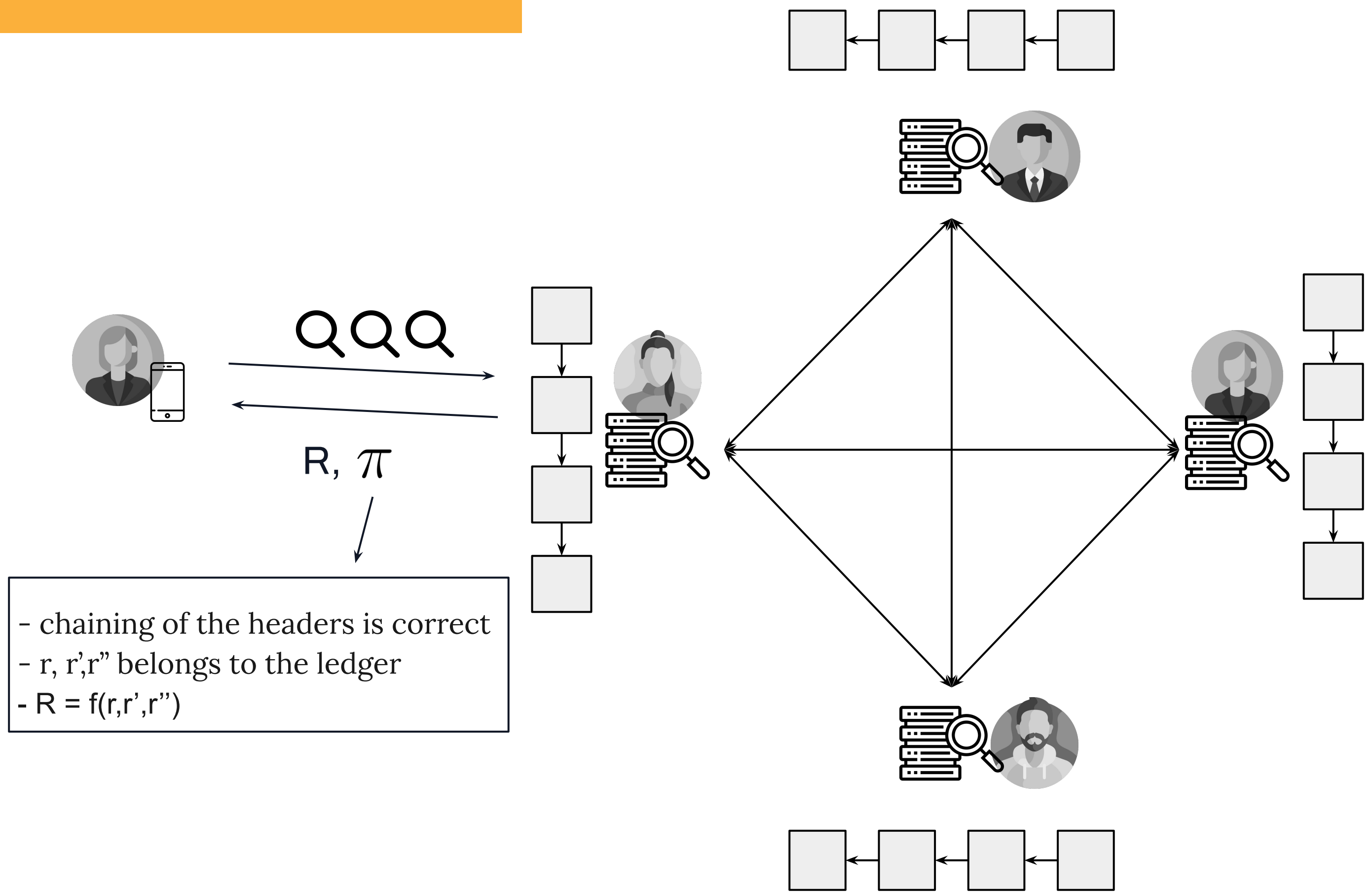
Small portion of the blockchain (i.e., header)





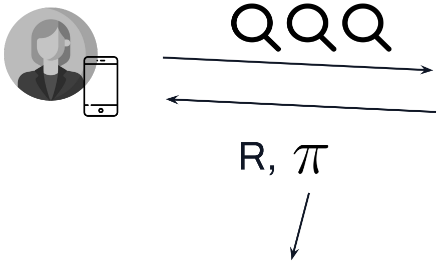
If application query involves **many transactions** the user has to **download** a lot of data!

Proof **Model**

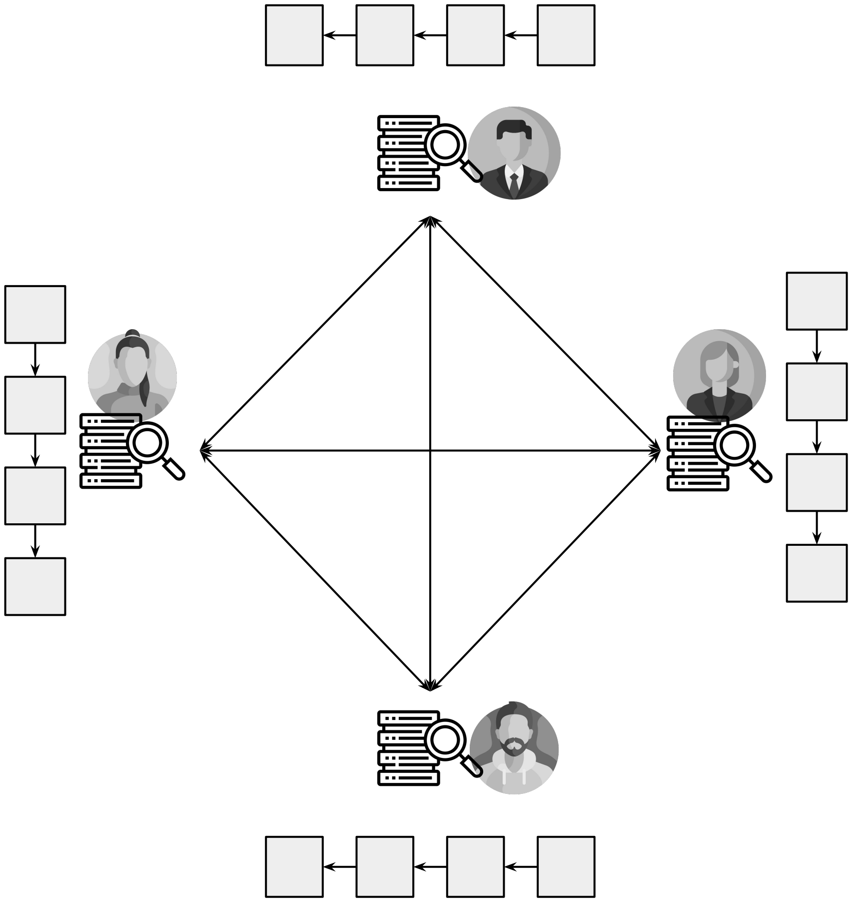


[XZC+ 2022] *T. Xie, J. Zhang, Z. Cheng et al., "**zkBridge: Trustless Cross-chain Bridges Made Practical**" - CCS - 2022*

~~Proof Model~~



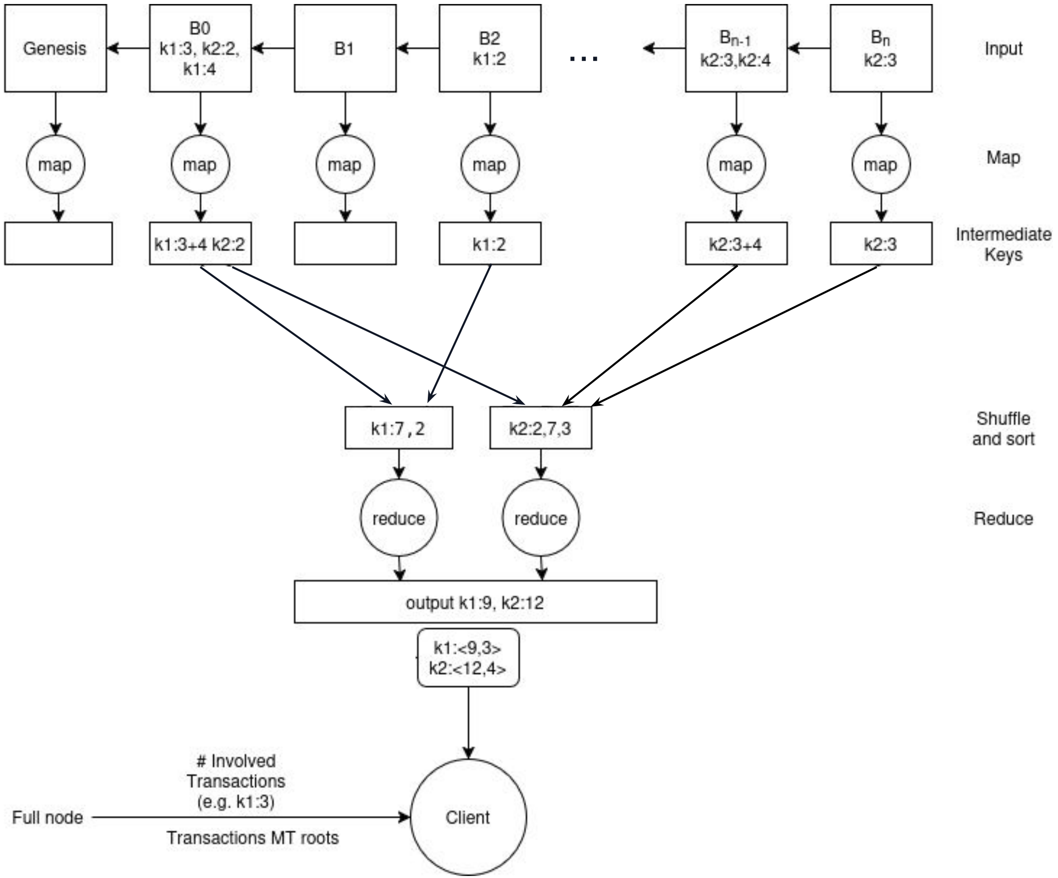
- chaining of the headers is correct
- r, r',r'' belongs to the ledger
- R = f(r,r',r'')



If application query involves **many transaction** generating the proof costs a lot (~ 50 Million dollars per year **[TZYT 2022]**)

[XZC+ 2022] *T. Xie, J. Zhang, Z. Cheng et al., "**zkBridge: Trustless Cross-chain Bridges Made Practical**" - CCS - 2022*
[TZYT 2022] *E. N. Tas, D. Zindros, L. Yang, D. Tse, "**Light clients for lazy blockchains**" - FC - 2024*

Map-Reduce queries



Our Results

We propose a new **stateless superlight client**:

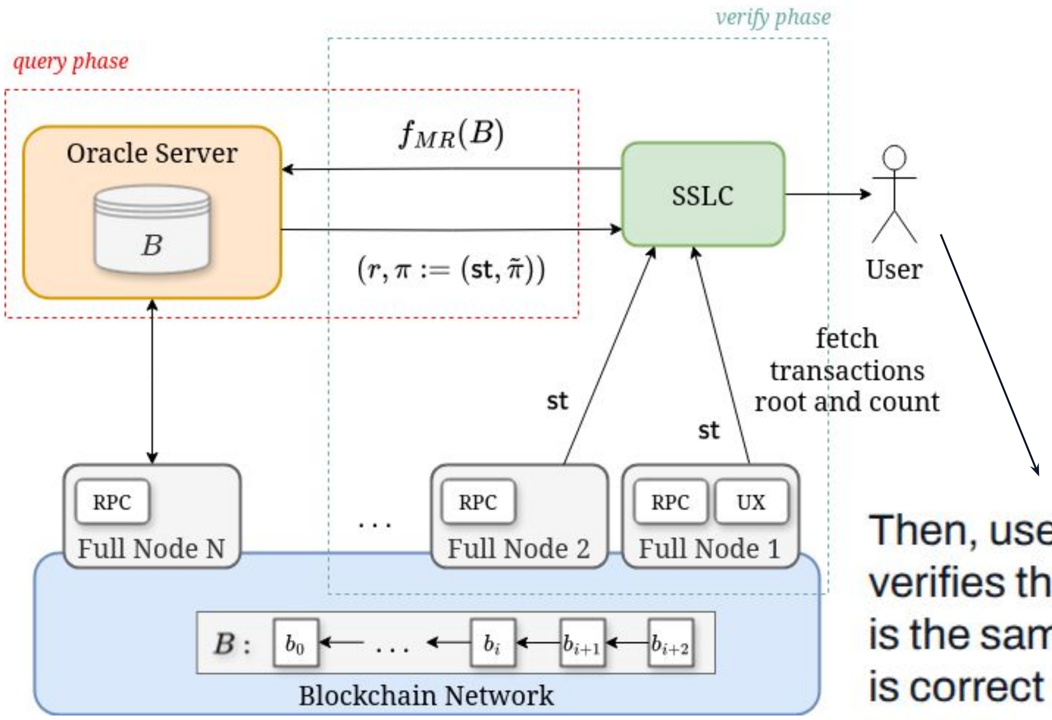
1 **Low resource constraints** for the client obtaining answers to **application specific** queries without **trusting** any server (or oracle)

2 **Feasible** for the server (or oracle) computing a proof proving the **correctness** of **application specific** queries

Stateless Superlight Client architecture

EFFICIENCY:

We took the best of **RPC** and **proof models**



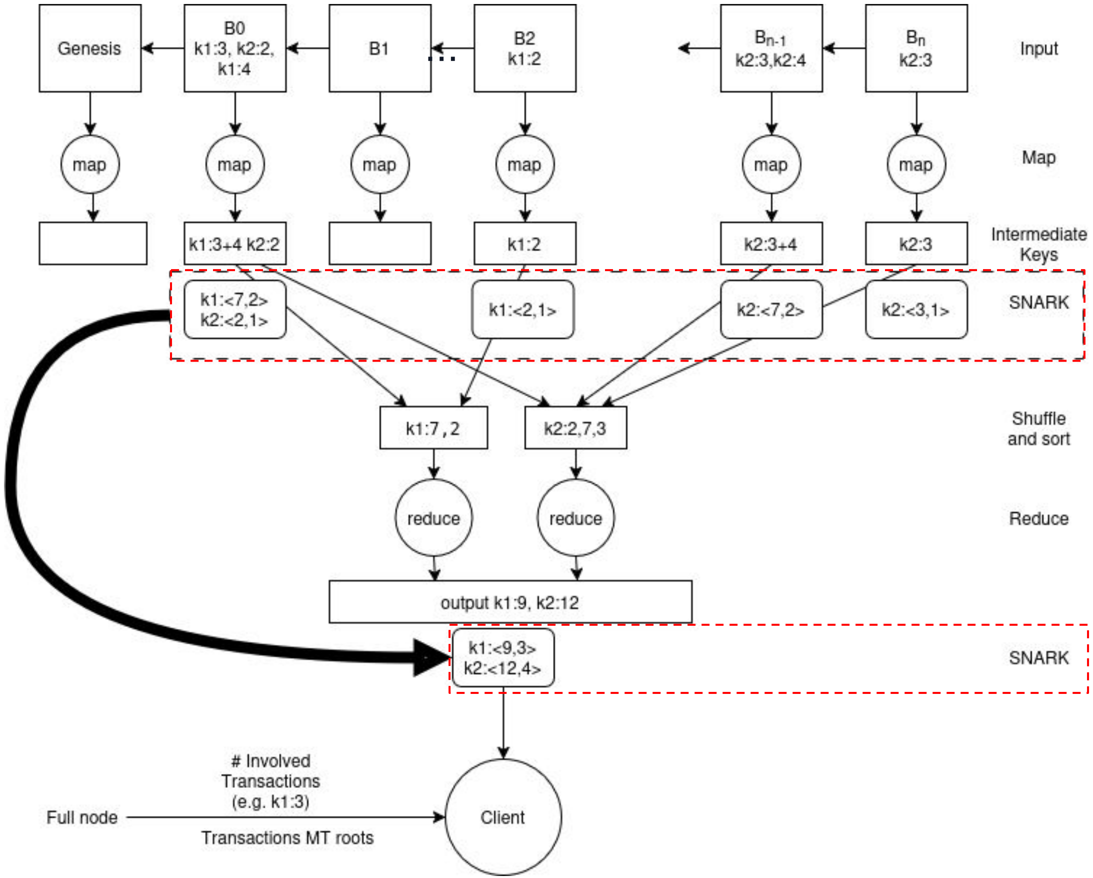
Then, user verifies that 1) st is the same, 2) $\tilde{\pi}$ is correct w.r.t. st to and *r*

Map-Reduce queries

FEASIBILITY:

One proof per **map** execution

It is possible to aggregate them in a single proof



SSLC Evaluation

COMMUNICATION EFFICIENCY IN E-VOTING USE CASE

Verification of **statistics** on an on-chain voting system governed by a smart contract.
We consider an election lasting one day, **~7000 blocks**, with a candidate receiving **70k votes** (i.e., 10 per block)

Nakamoto Light Client (NLC) performs the following step:

1. The NLC stores all the **headers** of the voting period;
2. Through **RPC,** NLC downloads all the **transactions** (70k) and all the **Merkle tree proofs**;

SSLC Evaluation

COMMUNICATION EFFICIENCY IN E-VOTING USE CASE

Verification of **statistics** on an on-chain voting system governed by a smart contract.
We consider an election lasting one day, **~7000 blocks**, with a candidate receiving **70k votes** (i.e., 10 per block)

Nakamoto Light Client (NLC) performs the following step:

1. The NLC stores all the **headers** of the voting period;
2. Through **RPC,** NLC downloads all the **transactions** (70k) and all the **Merkle tree proofs**;

SSLC performs the following step:

1. The SSLC queries the smart contract through an **RPC** to retrieve the **number of votes** for a candidate;
2. SSLC obtains from the oracle server the computed **answer** along with a **proof** that demonstrates how the retrieved transactions contribute to the final result;
3. SSLC **fetches and compares** the **relevant block headers** from a set of full nodes;
4. Given the trusted headers, the SSLC **verifies the proof** provided by the **oracle**

SSLC Evaluation

COMMUNICATION EFFICIENCY IN E-VOTING USE CASE

Verification of **statistics** on an on-chain voting system governed by a smart contract.
We consider an election lasting **one day**, **~7000 blocks**, with a candidate receiving **70k votes** (i.e., 10 per block)

	NLC	SSLC
Download Data	1.1 GB ⚠️	9 MB ✔️

Experiments

We **verify** Merkle membership proofs while **computing the average value** of a set of Bitcoin transfer transactions

The following experiments were conducted using **Plonky2** as ZK-SNARK instantiation, because it easily allows to aggregate proofs (through its recursive approach).

Experiments

FEASIBILITY ON INCREASING BATCH OF TRANSACTIONS

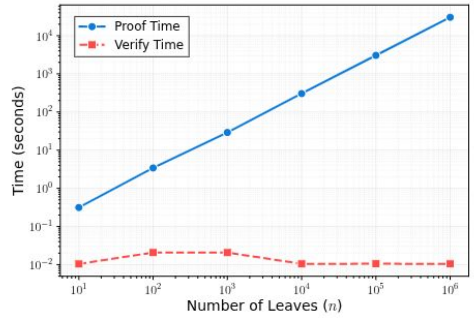
Workload	Proof Size (KB)	Prover		Verifier	
		Time (s)	Memory (GB)	Time (s)	Memory (GB)
10^1	96.5	0.314	0.213	0.010	0.213
10^2	124.8	3.418	0.320	0.020	0.270
10^3	156.73	29.088	1.110	0.020	0.717
10^4	156.98	304.273	1.420	0.010	1.040
10^5	156.98	3052.504	1.480	0.010	1.090
10^6	156.98	30434.426	2.060	0.010	1.660

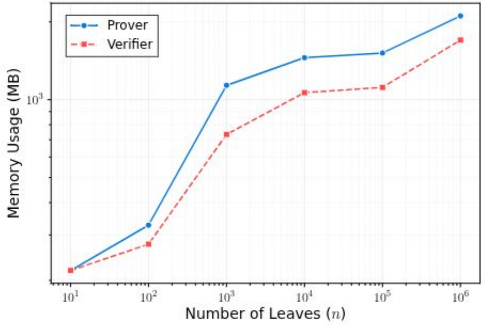
Table 2: Performance measurements for different numbers of leaves.

Recursive proving approach:

- Each recursive step handles a **batch** of **1000 transactions** (i.e., it simulates #txn selected per block by during the map function)
- We observed that the average recursion time per step to generate the proof was approximately 30 seconds.

We run the test on an Intel(R) Xeon(R) Silver 4216@2.1 GHz, 64 cores and 512 GB of RAM





Consumption increases **linearly** with the size of the batch

Discussion

Map-reduce queries that involve a large number of transactions, which is a **reasonable scenario in the future**, distributed across a **limited number of blocks**, our solution **outperforms** the NLC approach

When queries span **over a large number** of blocks, the benefits of this approach become **less apparent.** In these cases, the SSLC has to download more transaction roots/block hashes, making it **similar to that of NLC**.

Pre-print

THANKS!

This presentation includes icons from Flaticon