PRISM

From WoT to Chain

Enabling Zero-Trust Oracles for Blockchain IoT Applications

Ivan Zyrianoff

Department of Computer Science and Engineering
University of Bologna
ivandimitry.ribeiro@unibo.it

IoT Prism Laboratory @ UNIBO

IOT-PRISM LABORATORY ↓ RESEARCH ↓ GRADUATE WITH US ↓ NEWS CONTACT FORM DARK MODE SEARCH

IoT-Prism Laboratory

Welcome to **IoTPrism Laboratory**, a cutting-edge research laboratory located within the Department of Computer Science and Engineering at the University of Bologna.

Founded in 2018 by **Prof. Marco Di Felice** and **Prof. Luciano Bononi**, our lab is comprised of esteemed faculty members and talented young researchers dedicated to pushing the boundaries of **scientific and industrial research and innovation on pervasive IoT systems**.

Our mission is to investigate, design, and develop **innovative IT solutions for next-generation pervasive and mobile systems**, characterized by ubiquitous and autonomous sensing and connectivity, decentralized data management, adaptive and extreme edge processing and intelligence, and seamless cloud integration in the continuum. With a focus on IoT interoperability, data trustworthiness, and edge computing, we have addressed cutting-edge research challenges that are crucial to realizing the full potential of the IoT paradigm.

Over the last five years, we have collaborated on several **national and European research projects** implementing the IoT paradigm for **smart agriculture, industrial automation, condition and structural monitoring applications**. We have also published

SEARCH

search

RECENT POSTS

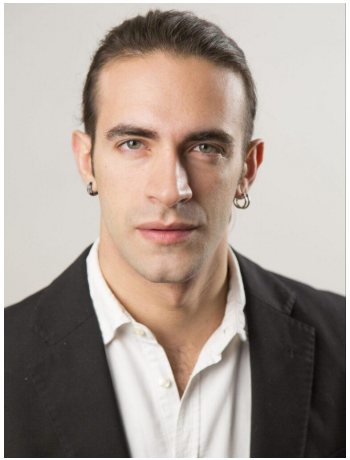
- Designing a Hybrid Push-Pull Architecture for Mobile Crowdsensing using the Web of Things
- A Decentralized Oracle Architecture for a Blockchain-Based IoT Global Market
- The 1st International Workshop on the Internet of Time-Critical Things (IoTime 2022)
- Mobile crowdsensing simulation

https://iotprismlab.com

Decentralized IoT Team

Prof. Marco Di Felice

Head of the lab



Prof. Federico Montori

Group Leader

Dr. Gigli

Blockchain Expert

Goal

Enable a Decentralized IoT Global Market

Towards an IoT Global Market

IoT is around since the 2010s, but the vision of the **Internet of Things being equivalent of The Internet** never came through

- Centralization and Lack of Transparency
- Low Trust in Data Quality
- Interoperability Problems

What is still missing?

The IoT market is vast, integrating diverse data sources from multiple providers and infrastructure owners.

In traditional cloud-based services, users' data requests are tightly coupled with specific data providers.

- Users primarily care about **WHAT** data to retrieve and **WHERE** it originates (geolocation), not WHO the provider is.

- **Trust and data quality** are critical — users should be able to rely on the marketplace itself, rather than knowing or trusting each individual (and potentially anonymous) data source.

How to enable full decoupling between users' requests and the IoT data providers?

WoT + Blockchain as a solution

Why Web of Things?

1. Team Experience

2. Standard Interface

3. Semantic Description

4. Discovery capabilities

With W3C WoT we can leverage the **Discovery** and **Thing Description** to:

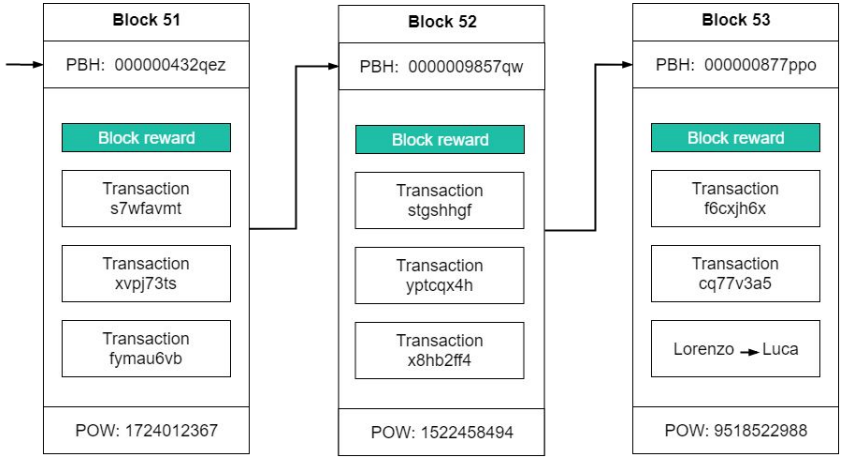
- WoT Discovery lets us query what **data we want**
- The Indexer (TDD) returns TDs whose **affordances provide that data**
- We can directly **consume these TDs** to access the data

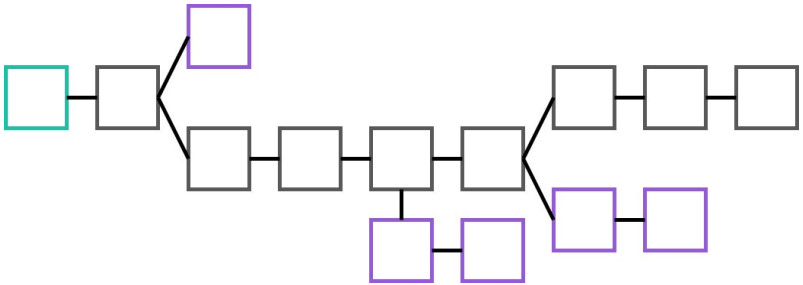
What is a blockchain?

A blockchain is a **distributed ledger** with growing lists of records (blocks) that are securely linked together via cryptographic hashes.

Distributed Ledger?

You can see a distributed ledger like a **distributed database, but without the need for a central administrator**





What is a Smart Contract?

Smart Contracts are software that can be deployed and executed inside a blockchain.

You can invoke smart contract's methods through transactions.

Inherit the blockchain's properties!

- Immutability
- Transparency
- Security
- Decentralization

```
contract SHMController is Initializable, Ownable {
    using SafeMath for uint256;
    using ArrayUtils for uint256[];

    // DATA VARIABLES

    struct Measurement {
        uint64 timestamp; // 1465839830100400200 (Unix time)
        string structureId; // dicam083010
        string sensorId; // DA00AW2K-13-measure
        string sensorType; // accelerometer
        string data; // x,y,z
    }

    mapping(uint256 => Measurement) private measurements; // 1 - Measurement
    mapping(uint256 => uint256[]) private dateMap; // 20210727 - [..., 1]
    mapping(string => uint256[]) private structureMap; // dicam083010 - [..., 1]

    uint256 measurementCount = 0;

    // EVENT DEFINITIONS

    event MeasurementAdded(uint256 measurementId);

    // SMART CONTRACT METHODS

    function initialize() public initializer {}
```

Why we utilize the Blockchain?

01

Decentralization

No single point of control

02

Immutability

Data cannot be altered retroactively

03

Transparency

All transactions are public and auditable

04

Permissionless

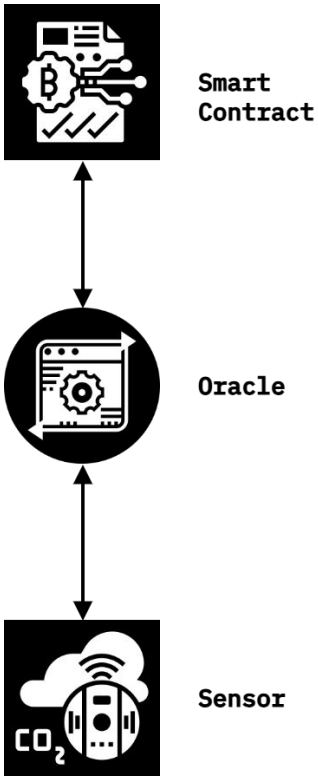
Anyone can join without centralized gatekeeper

Problems?

They can’t access external data!

The code execution **MUST** be deterministic in order to be verifiable by the network.

Blockchain Data Oracle



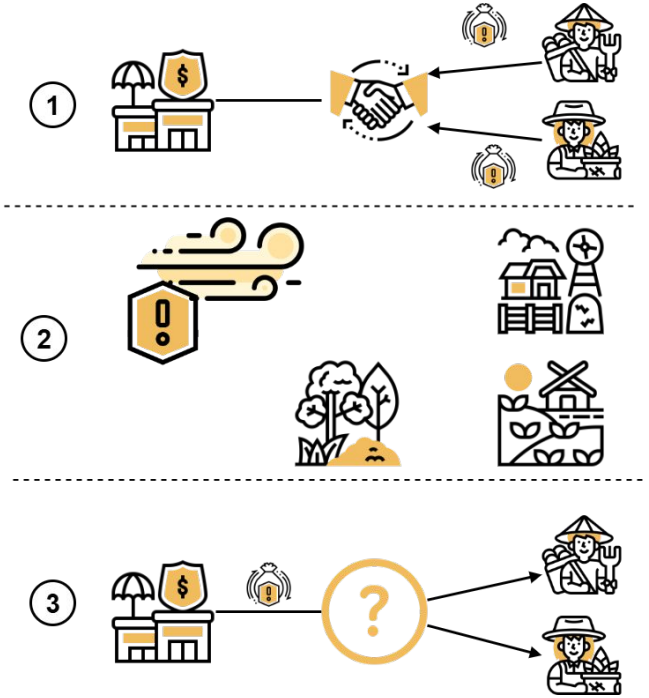
- Blockchains **cannot** access external information
- Acts as a **bridge** between the blockchain and the external world.
- Fetches **off-chain data** (e.g., sensor readings, weather data) and delivers it to the blockchain for use in smart contracts.

IoT + Blockchain Example

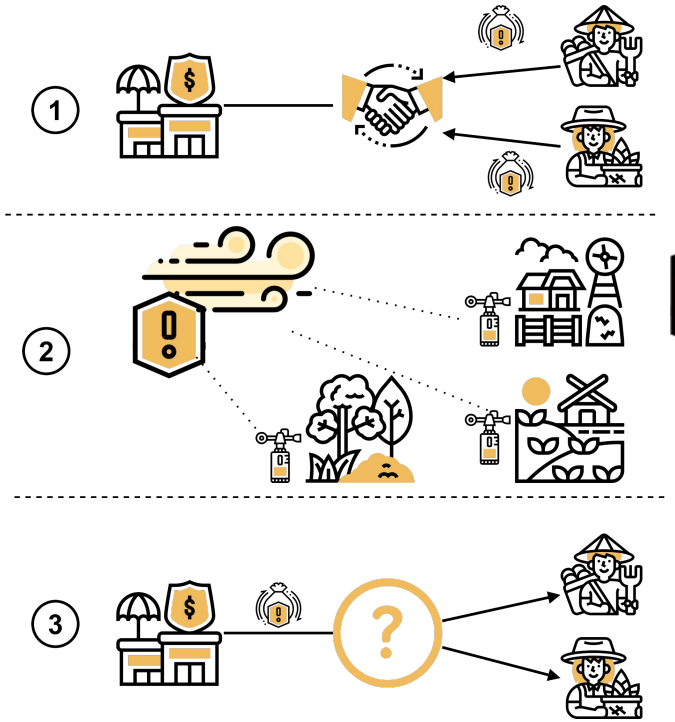
Scenario #1: Traditional Parametric Insurance

A farmer's insurance pays out if the rainfall reports are **below a certain threshold**

- **Basis Risk**: The received report may not reflect the conditions on the specific farm
- **Disputes & Ambiguity:** The claims process can be slow, cumbersome, and lead to disputes
- **High Administrative Costs**: Manual verification and processing are inefficient



Scenario #2: IoT-Based Parametric Insurance



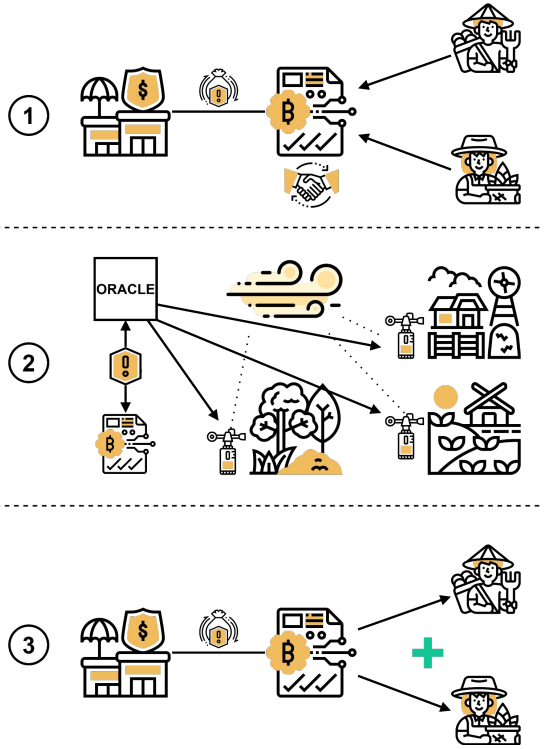
The insurance contract is linked directly to an **IoT weather** station on the farmer's field

- **Trust and Data Tampering**: The insurance company may not trust data from a *single source* owned by either party, who could *manipulate* it
- **Single Point of Failure**: The system is *vulnerable* to sensor malfunctions or localized tampering

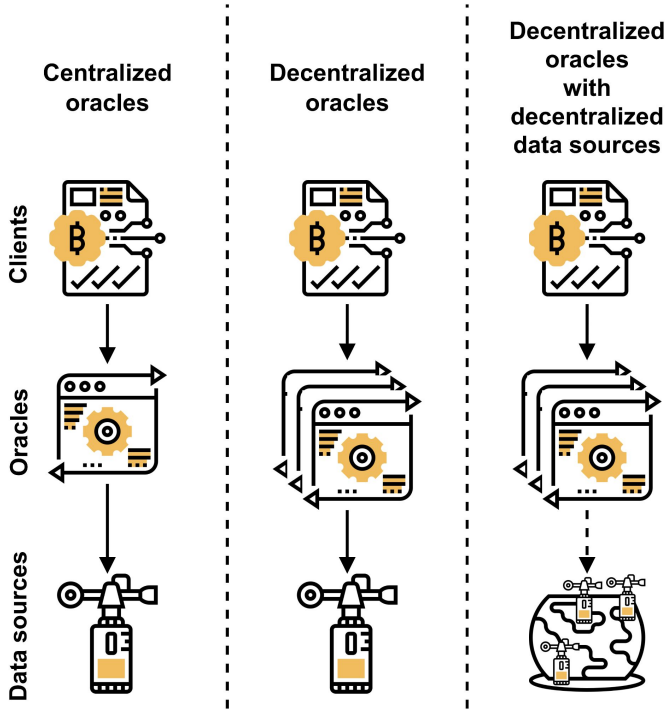
Scenario #3: Blockchain-based Smart Insurance

The insurance **smart contract** on the **blockchain** uses a **network of oracles** to query **multiple**, independent **IoT sensors**

- **Decentralized Trust**: several of sources prevents collusion and removes reliance on a single party
- **High Resilience**: Aggregating data from different sensors makes the system robust against failures
- **Automation & Transparency**: The smart contract issue funds instantly based on verifiable data



Blockchain Data Oracle



- Blockchains **cannot** access external information
- An **oracle** fetches **external data** and delivers it to the chain

Oracle Problem: *How can you trust the data provided by the oracle?*

- ➢ Oracle can be compromised
- ➢ Source can be compromised

Blockchain Data Oracle

Oracles in the state of art

- ~~Centralized~~
- ~~Trusted Execution Environment~~
- ~~Homogeneous data sources~~

OUR PROPOSAL

Fully Decentralized

Permissionless

Interoperability support

ZONIA-MC
Open Call #4

NGI TRUST CHAIN

L. Gigli, I. Zyrianoff, F. Montori, L. Sciullo, C. Kamienski, M. Di Felice ZONIA: a Zero-Trust Oracle System for Blockchain IoT Applications, **IEEE Internet of Things Journal**, 2025

L. Gigli, I. Zyrianoff, F. Montori, C. Aguzzi, L. Roffia, M. Di Felice, A decentralized oracle architecture for a blockchain-based iot global market, **IEEE Communication Magazine**, 2024

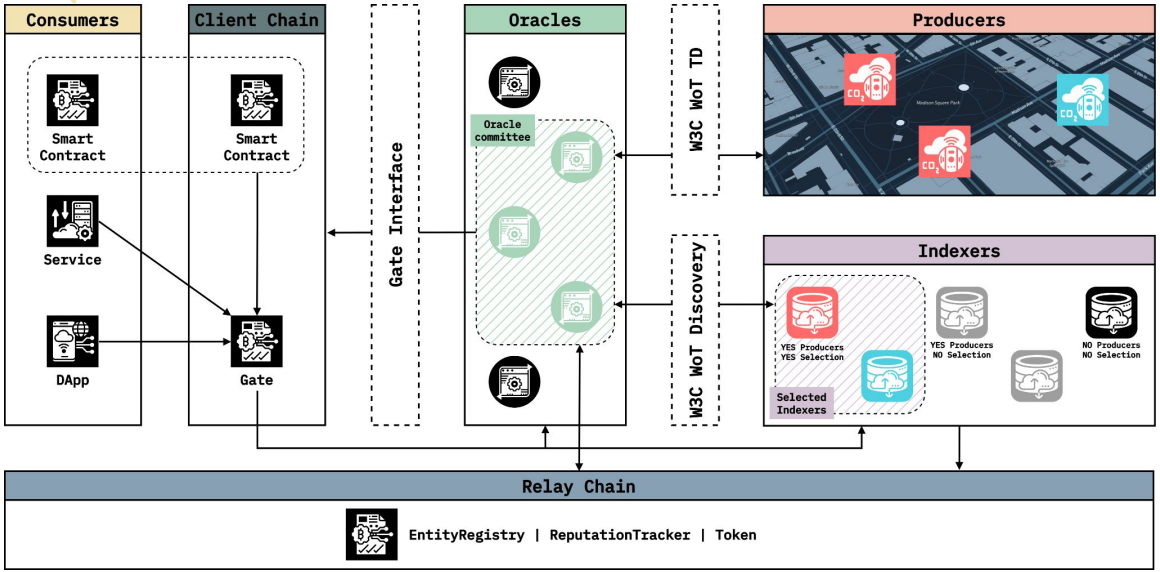
ZONIA

What is ZONIA?

Zero-Trust **O**racle **N**etwork for **I**oT **A**pplications is an oracle system that retrieves data from real-world sensors and devices.

- **Zero-Trust Decentralized Design:** anybody can join/leave
- **Advanced Requests:** semantic and geospatial
- **Fair and Secure Node Selection:** for each submitted request, the system automatically selects a group of nodes based on randomness and reputation
- **Hidden Data Sources:** clients cannot choose specific sources or their data needs
- **Reputation System:** nodes are evaluated based on their performance and honesty

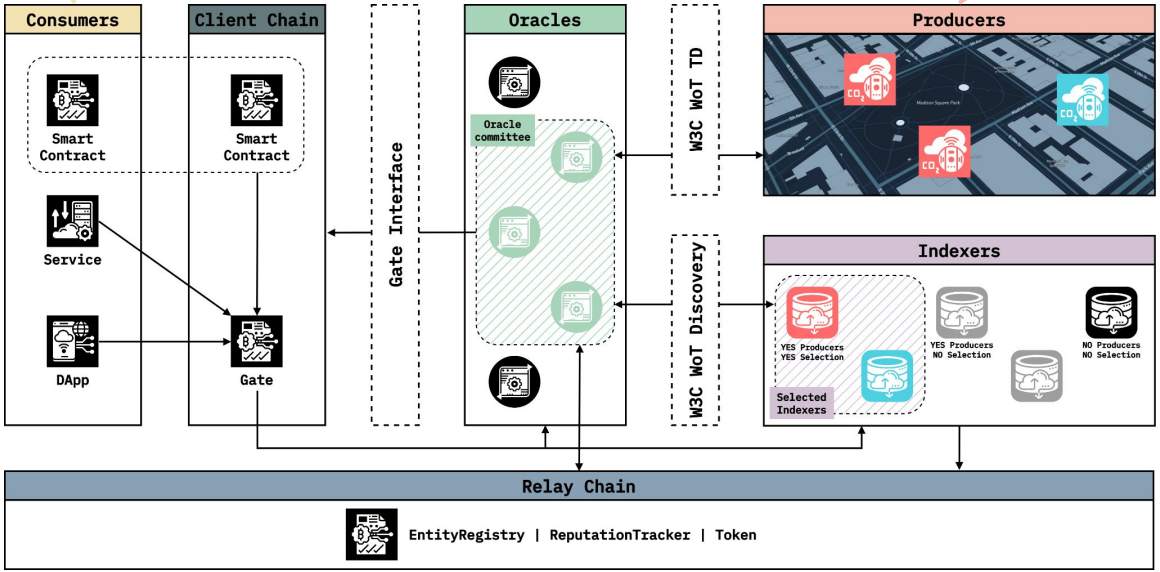
Initiate data request
Pay for the data



ZONIA Architecture

Initiate data request
Pay for the data

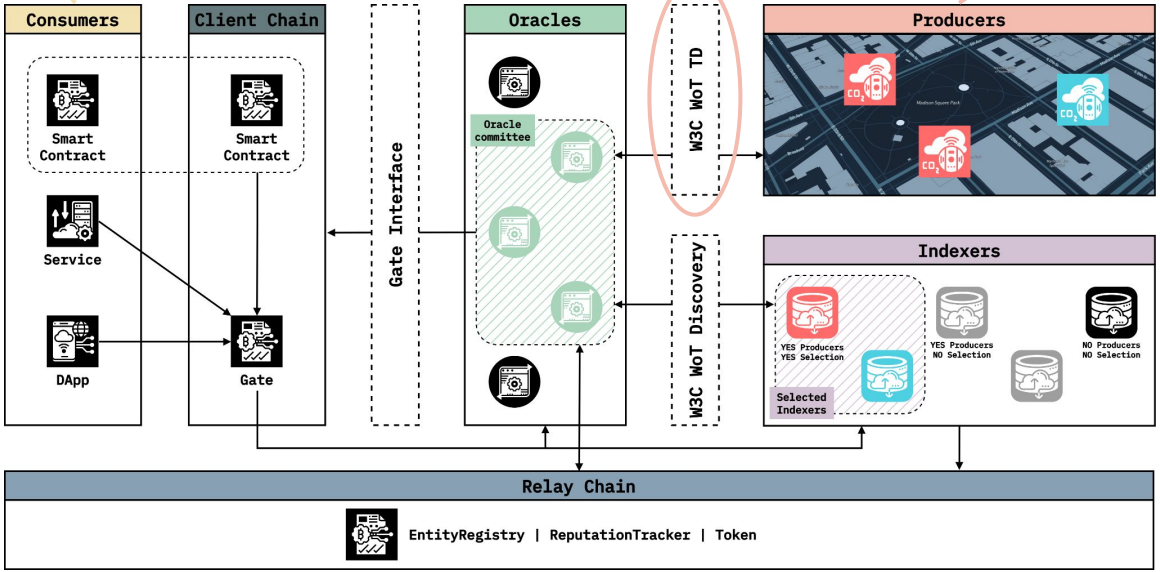
Static sensors,
drones, smartphones
Paid for the data



ZONIA Architecture

Initiate data request
Pay for the data

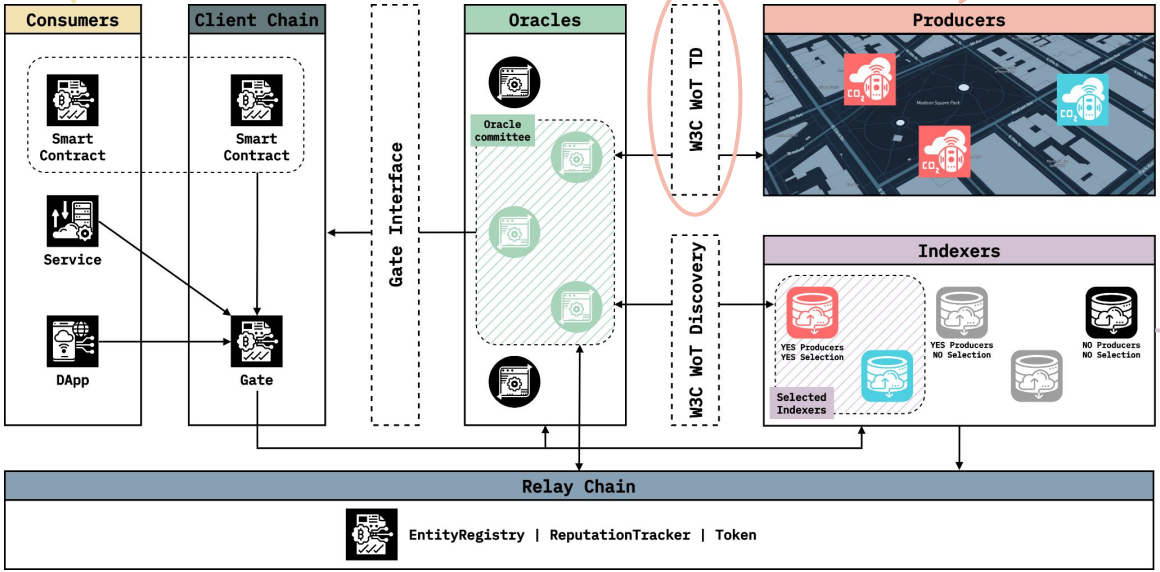
Static sensors,
drones, smartphones
Paid for the data



ZONIA Architecture

Initiate data request
Pay for the data

Static sensors,
drones, smartphones
Paid for the data

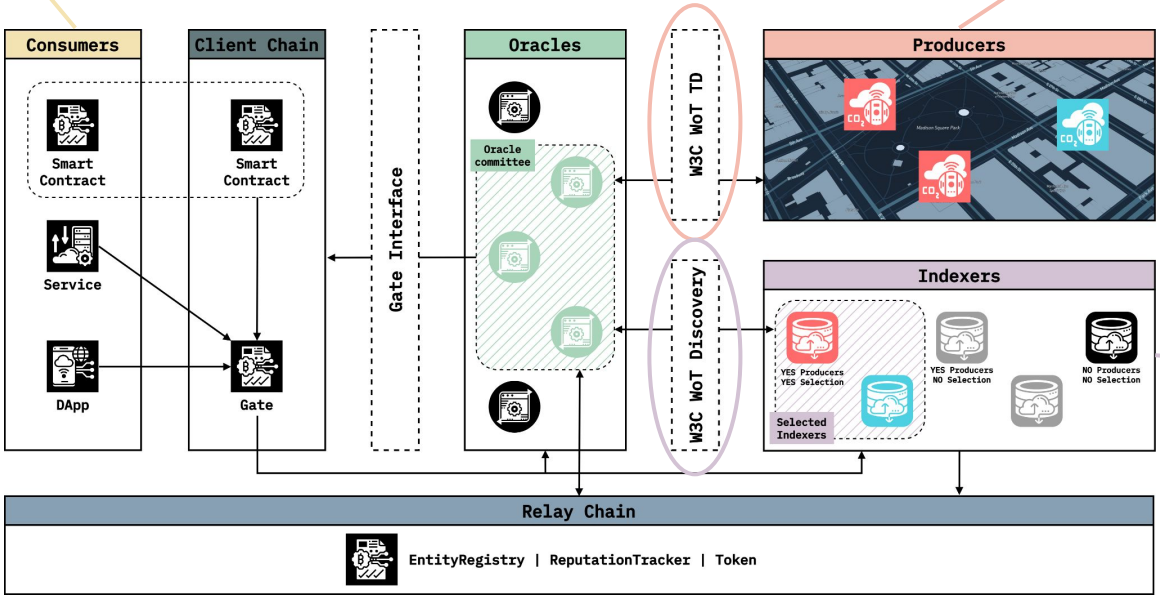


Keep track of
Producers
metadata

ZONIA Architecture

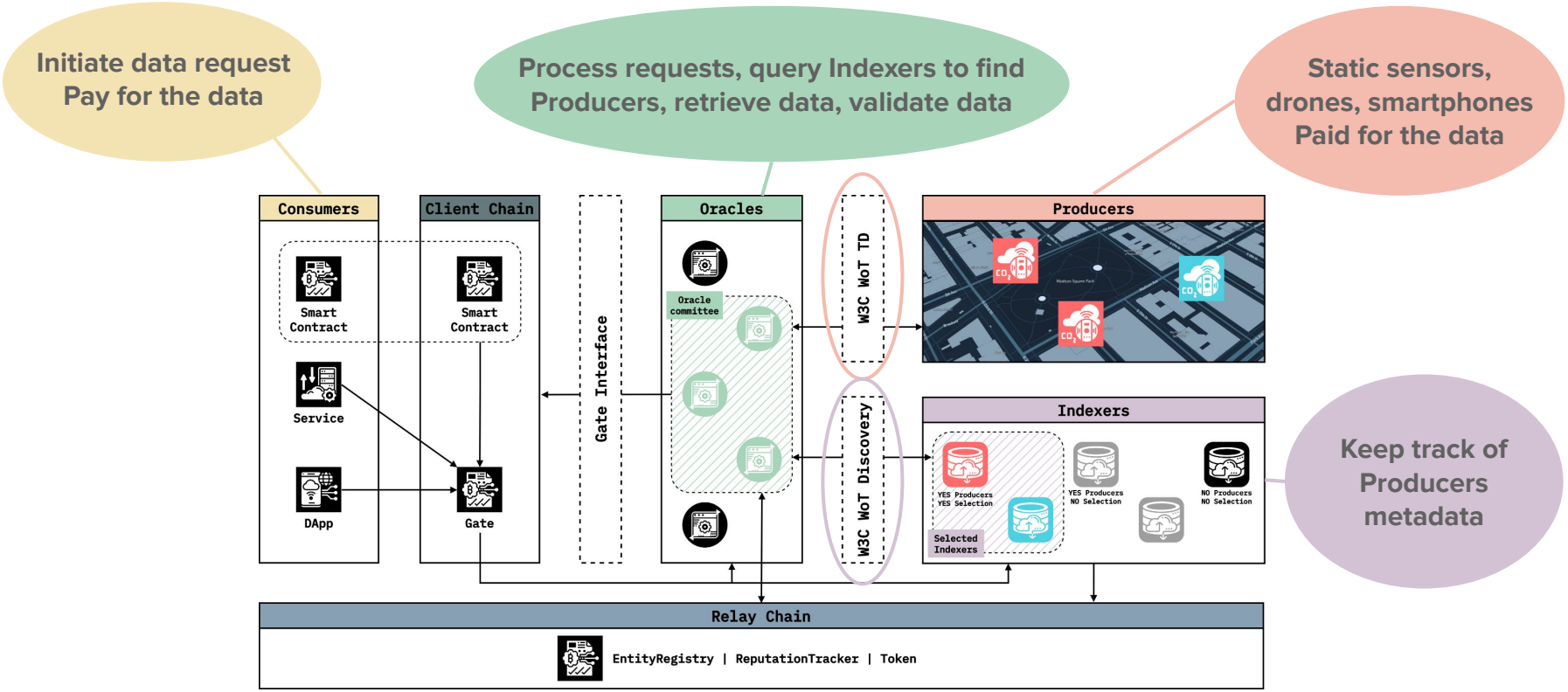
Initiate data request
Pay for the data

Static sensors,
drones, smartphones
Paid for the data

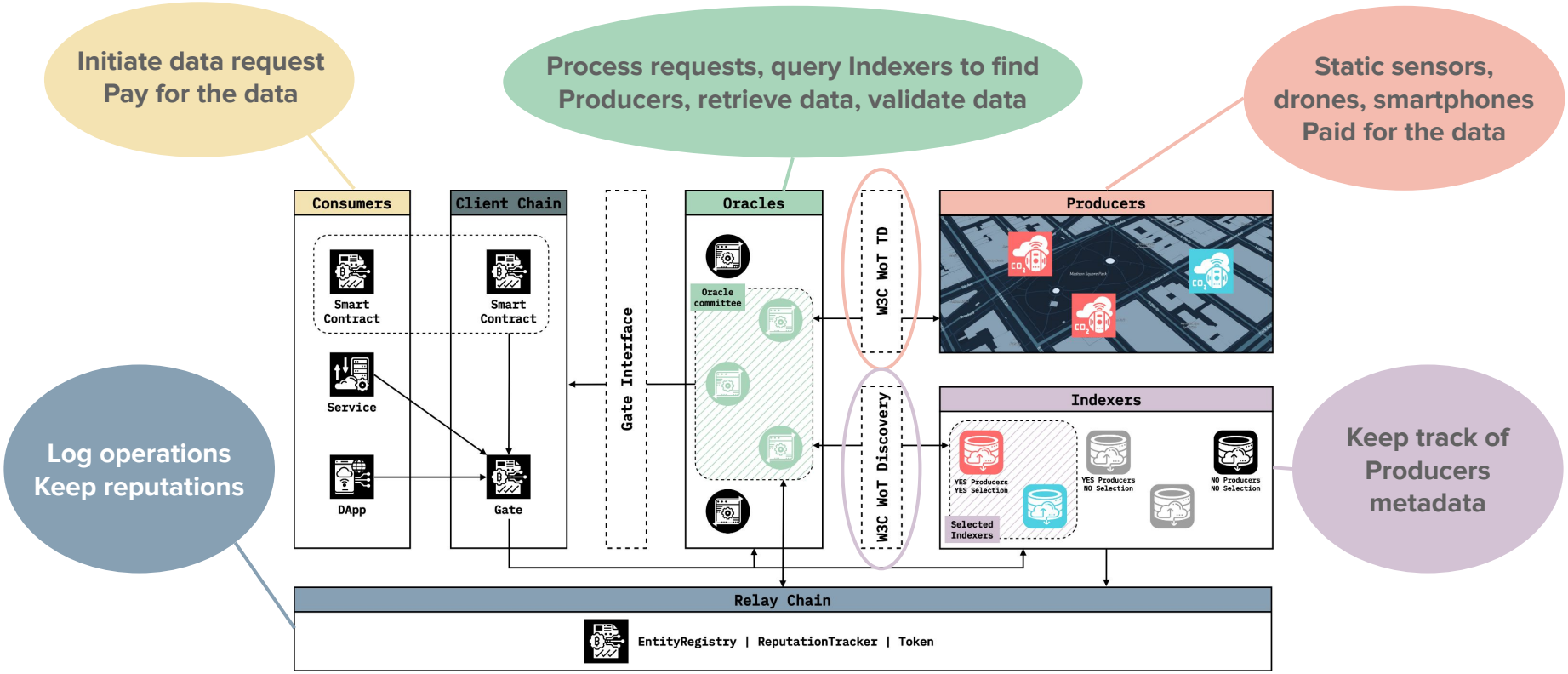


Keep track of
Producers
metadata

ZONIA Architecture

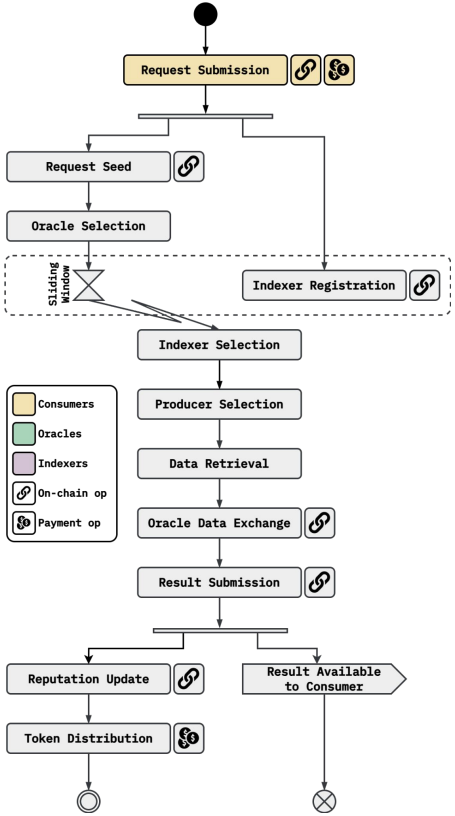


ZONIA Architecture

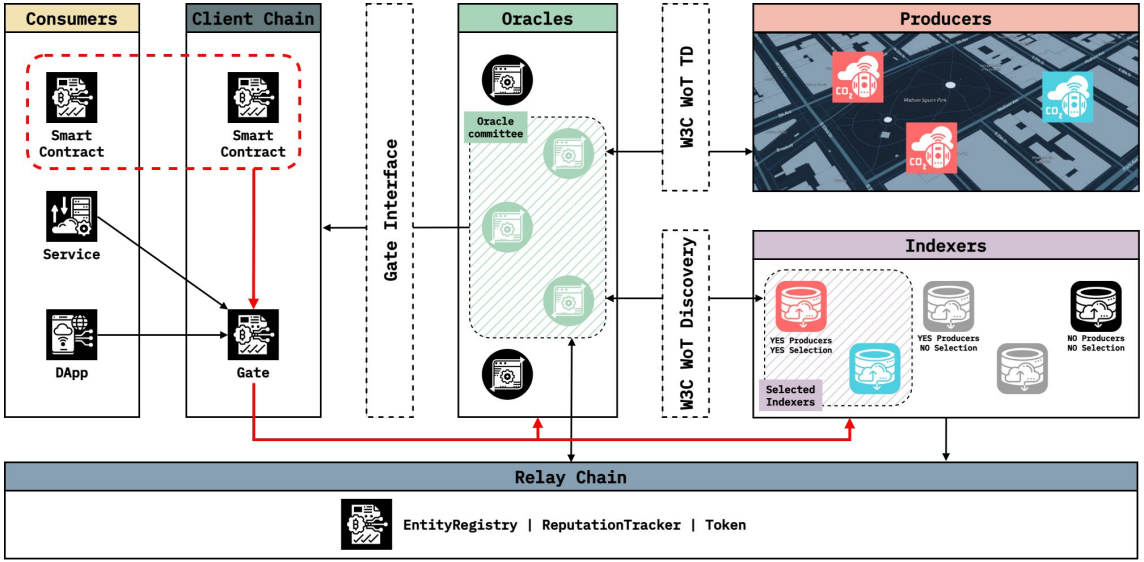


ZONIA Architecture

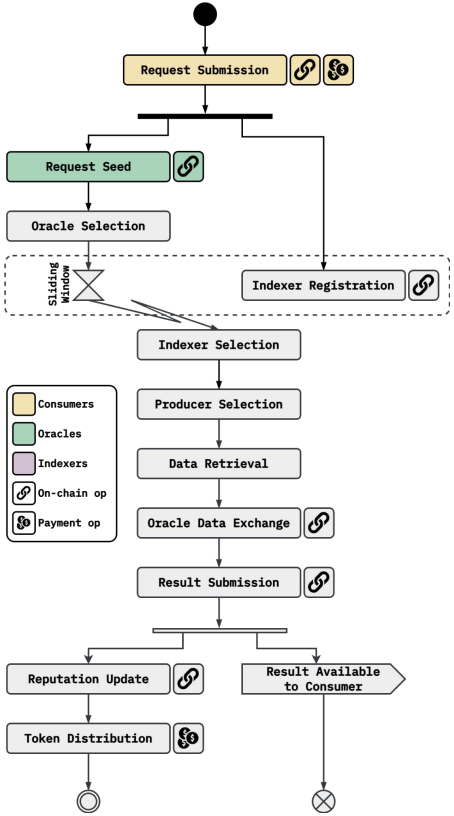
How does it work?



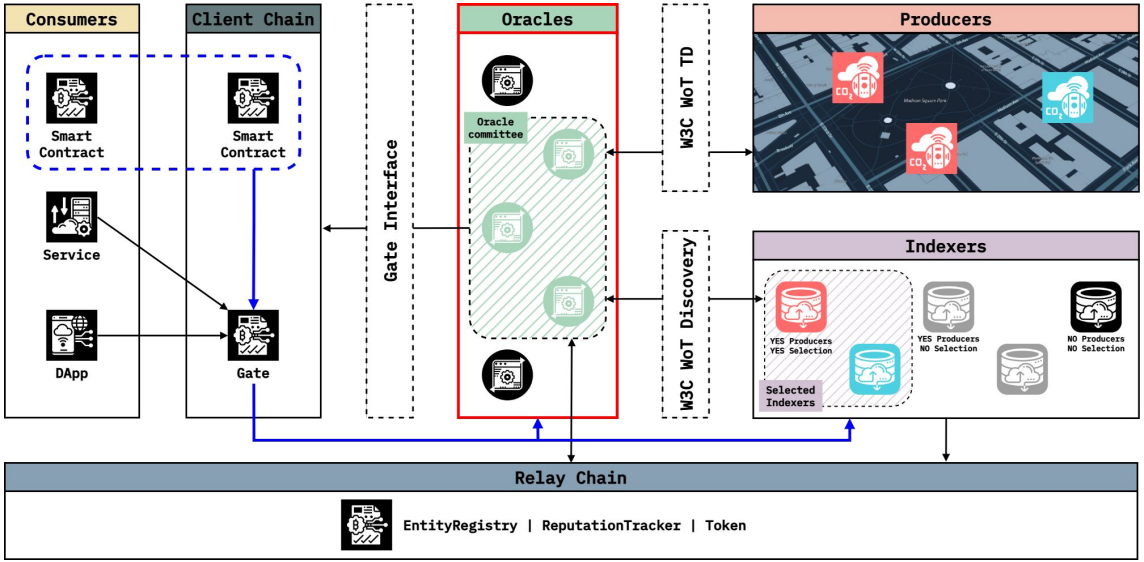
?q = “Which is the Co2 level in NYC?”



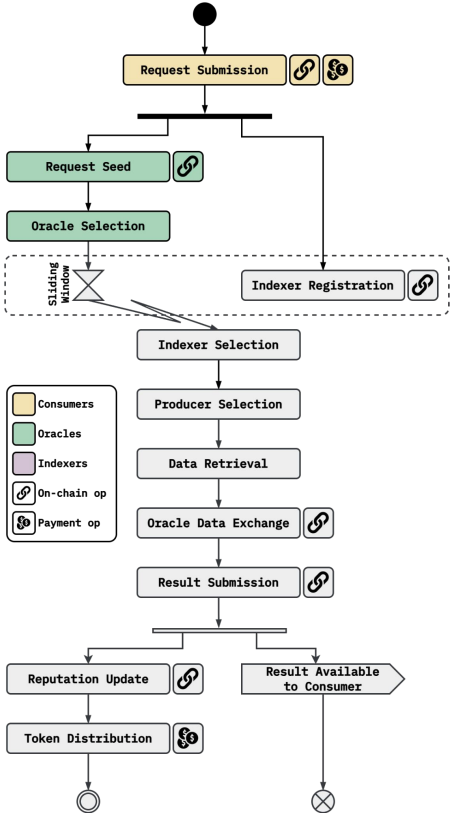
Request resolution flow



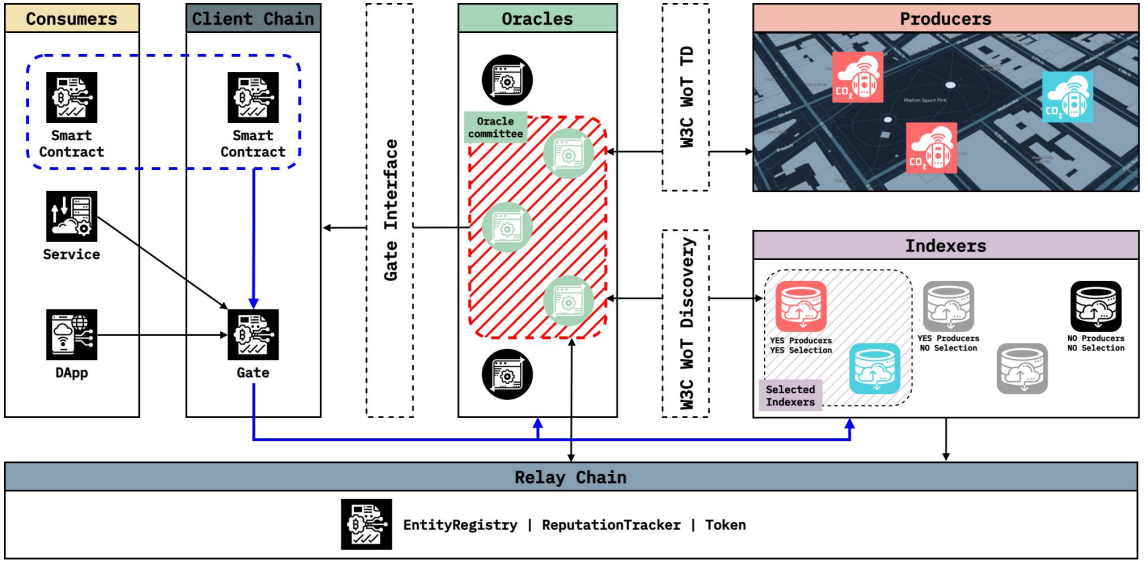
?q = “Which is the Co2 level in NYC?”



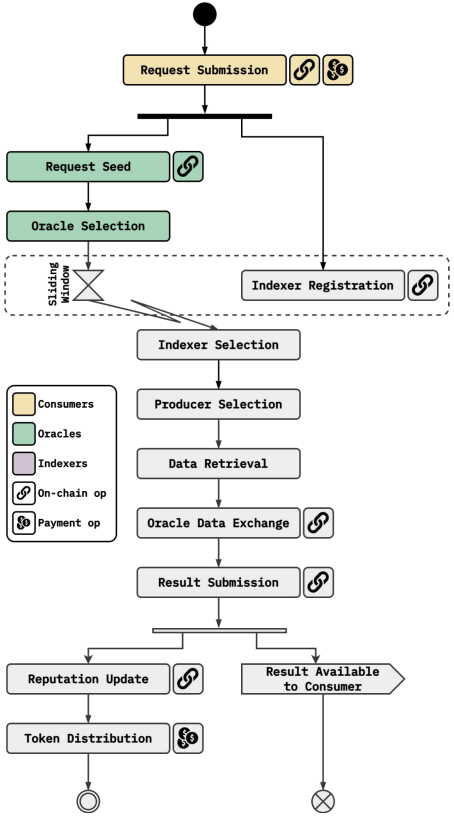
Request resolution flow



?q = “Which is the Co2 level in NYC?”



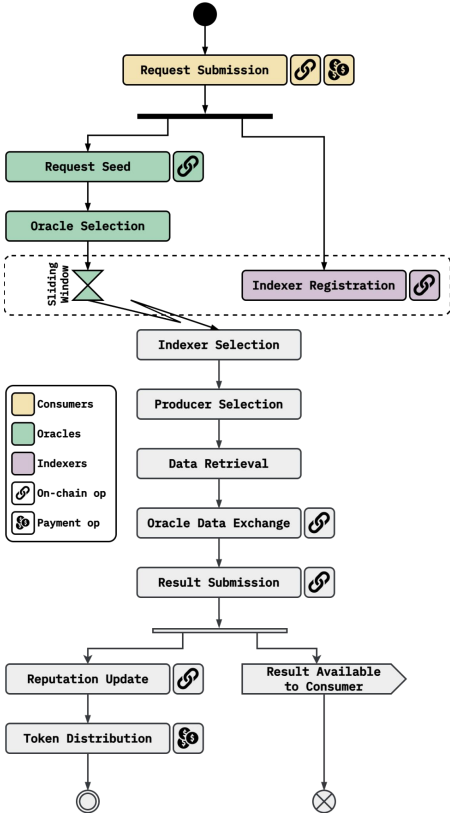
Request resolution flow



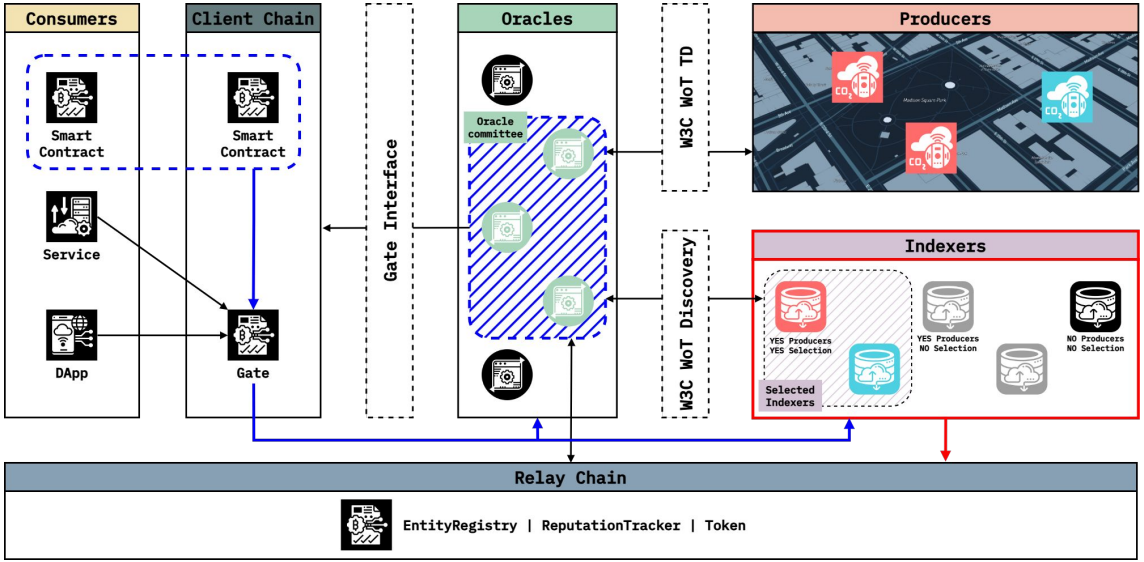
Oracle Selection:

- Nodes are selected based on a combination of **reputation and randomness**
- Fully **deterministic and verifiable**
- **No communication** between nodes
- Each node discovers whether it is part of the committee and identifies the other selected nodes

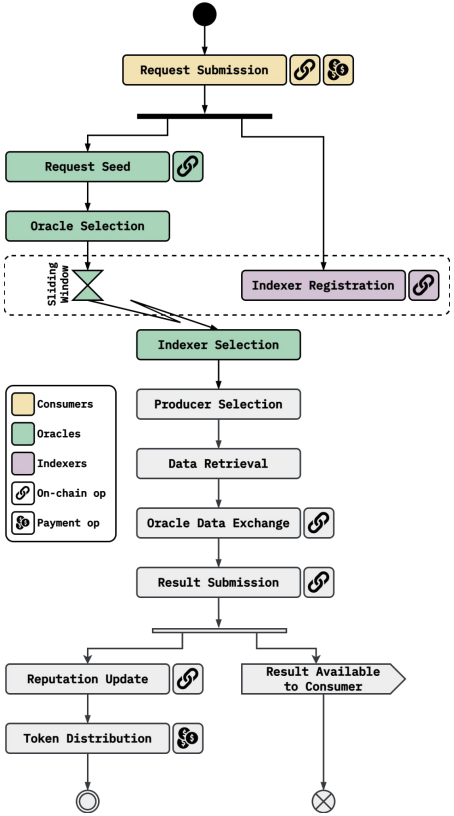
Request resolution flow



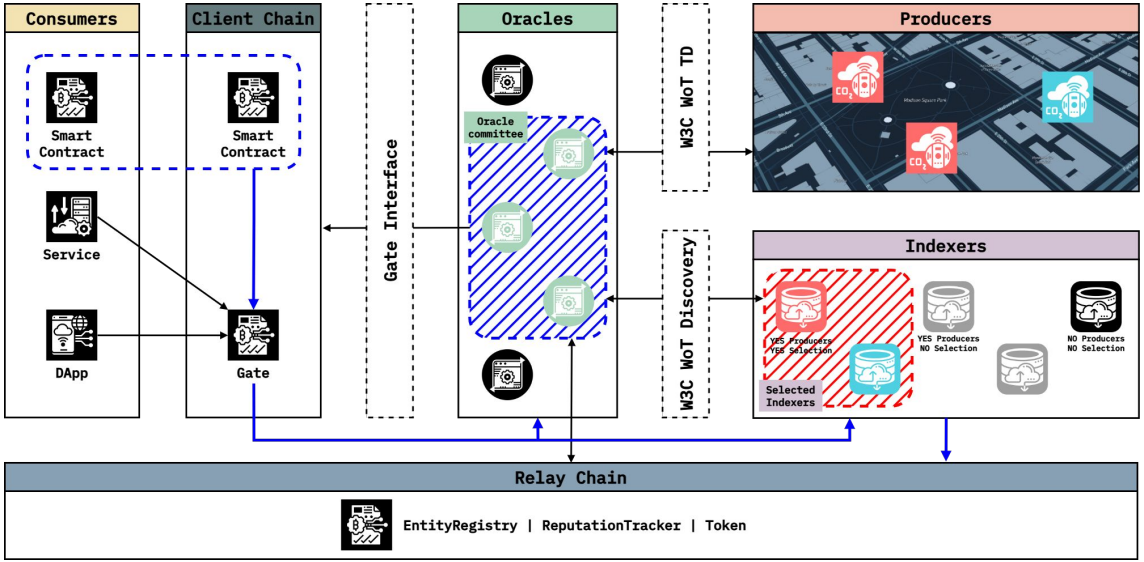
?q = “Which is the Co2 level in NYC?”



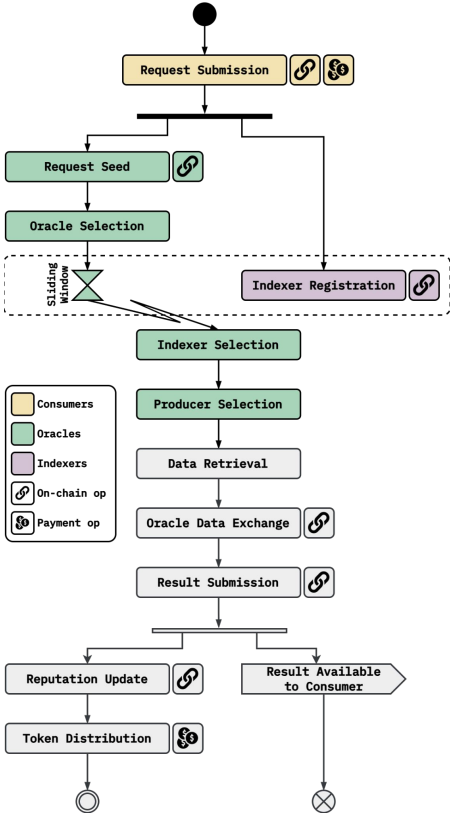
Request resolution flow



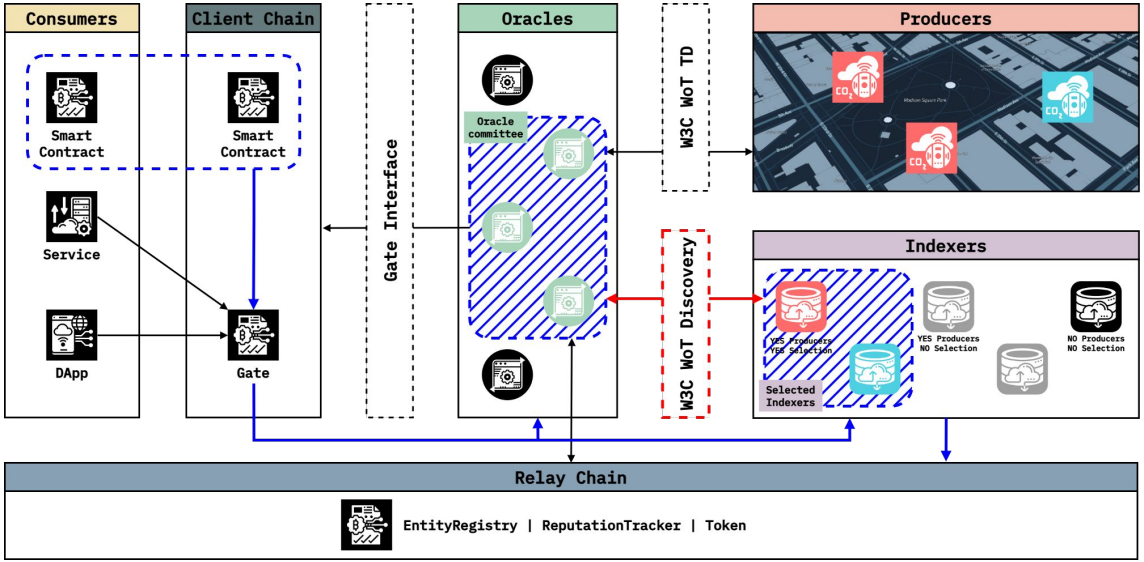
?q = “Which is the Co2 level in NYC?”



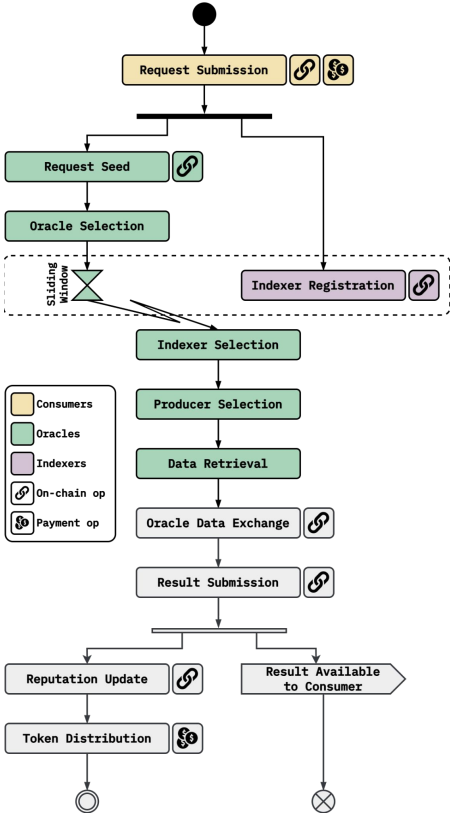
Request resolution flow



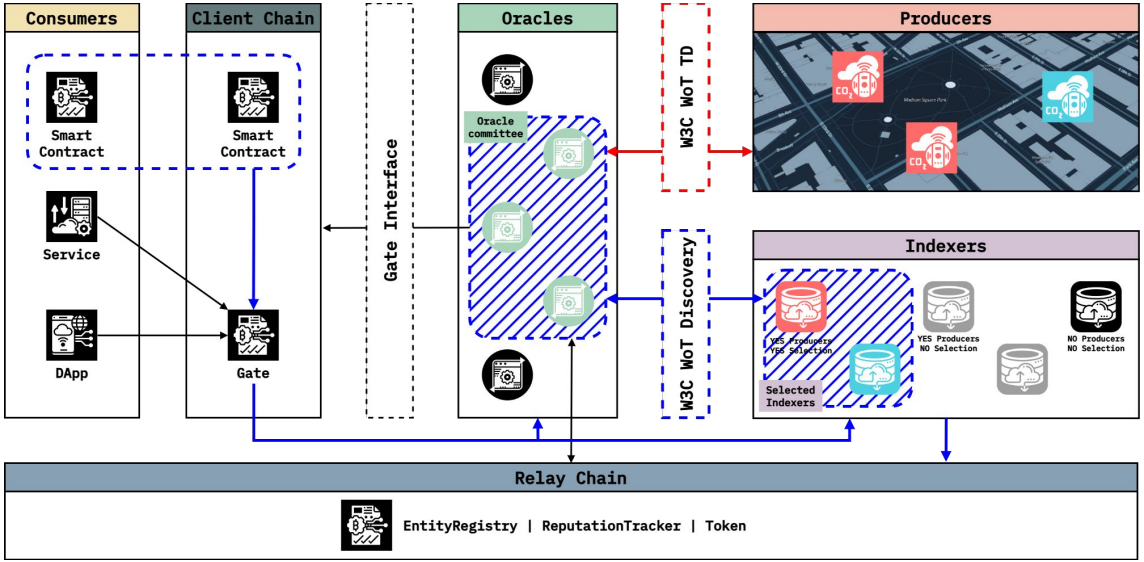
?q = “Which is the Co2 level in NYC?”



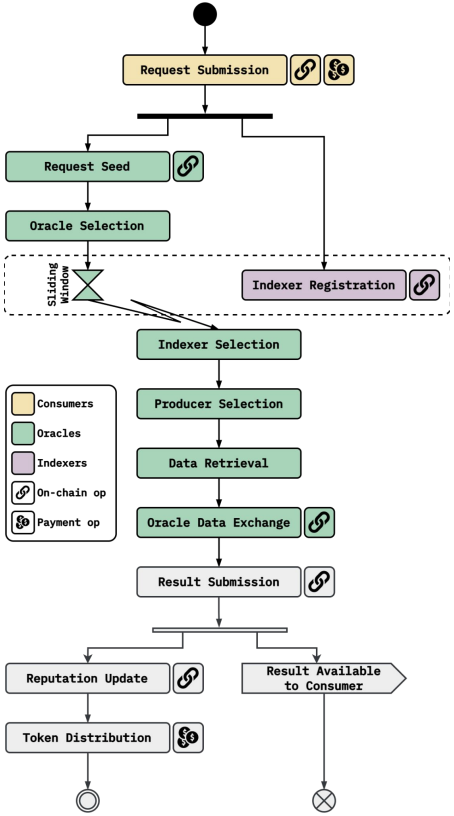
Request resolution flow



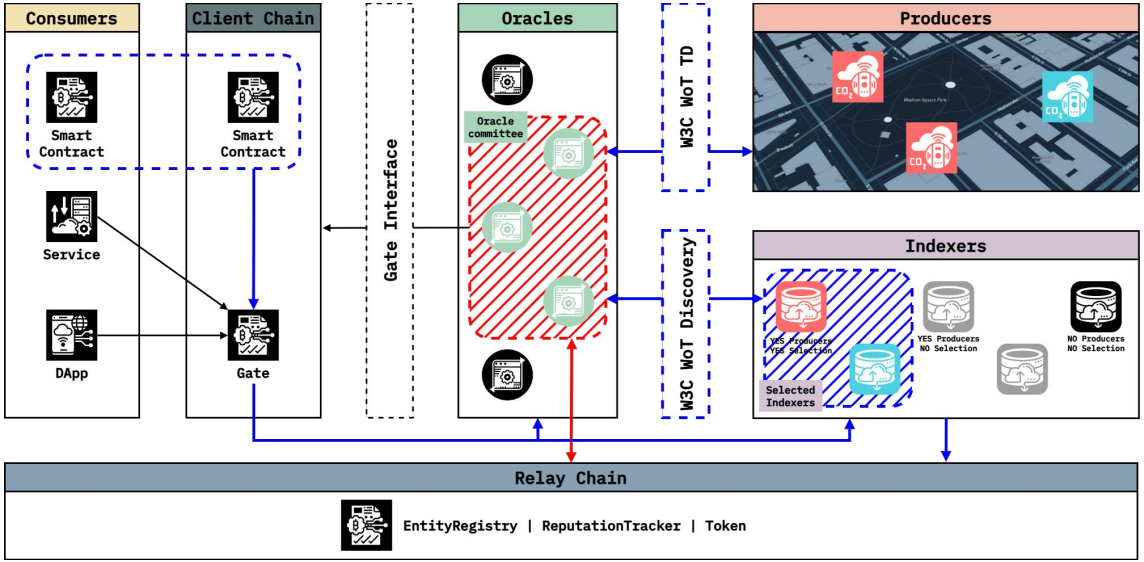
?q = “Which is the Co2 level in NYC?”



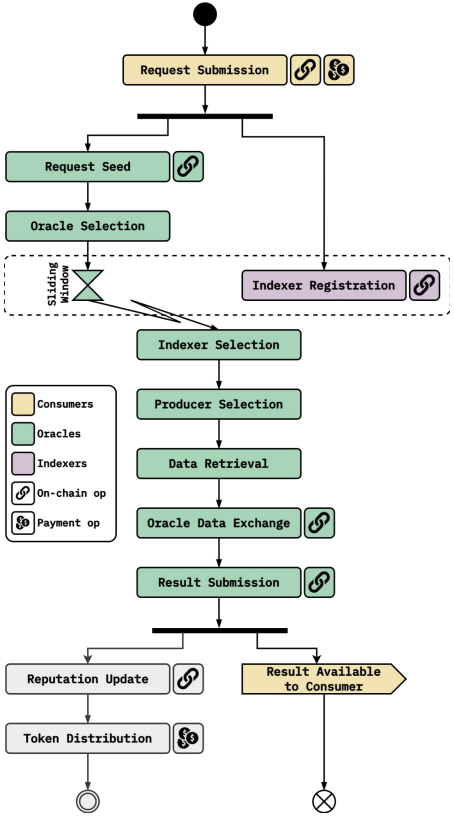
Request resolution flow



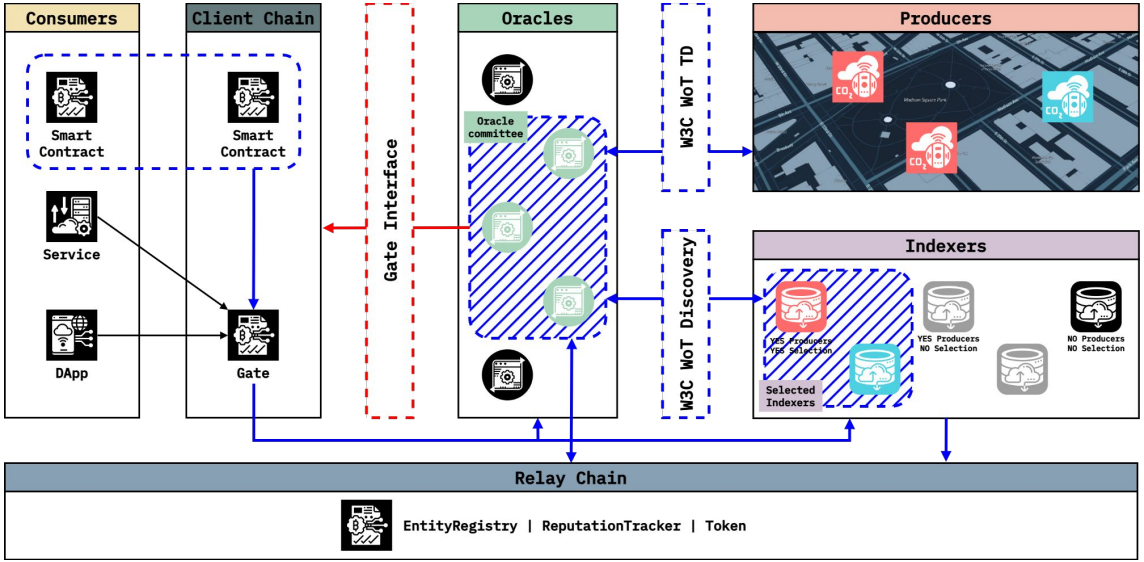
?q = “Which is the Co2 level in NYC?”



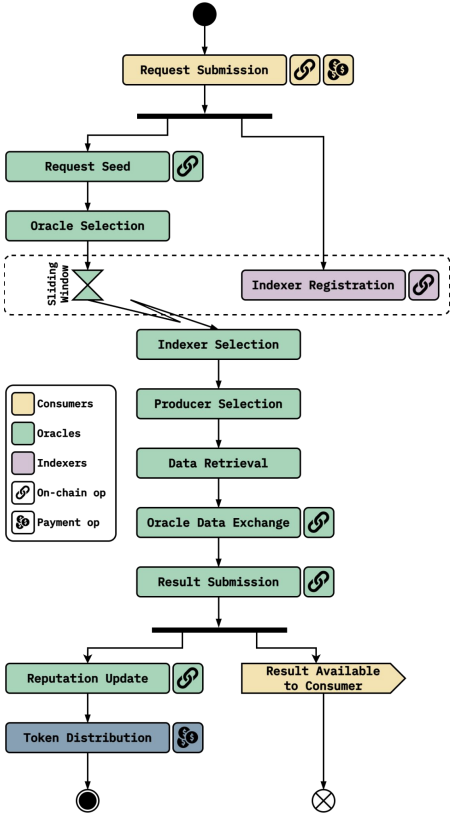
Request resolution flow



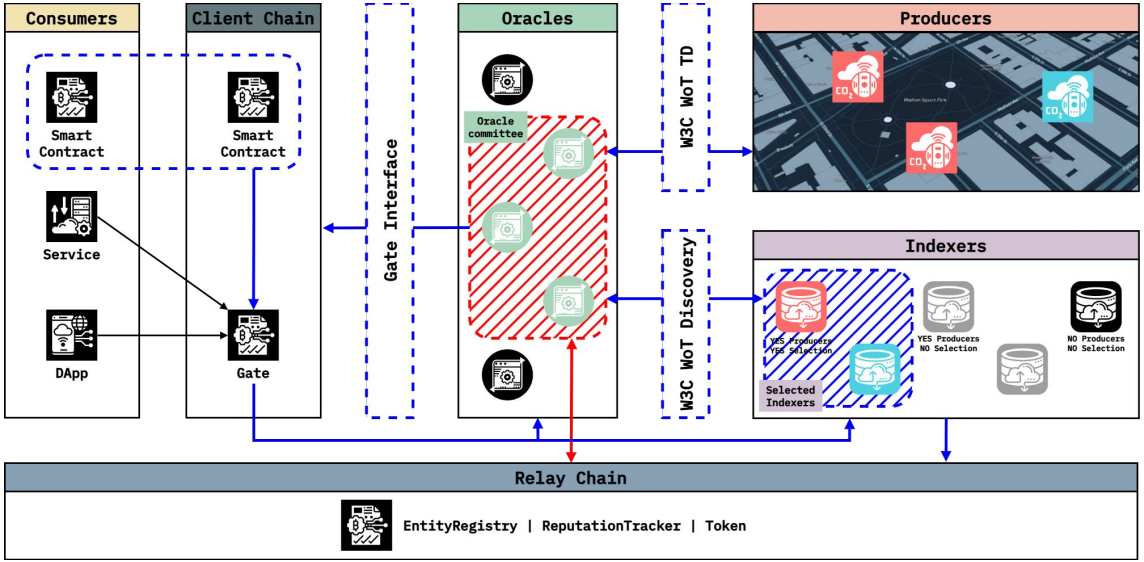
?q = “Which is the Co2 level in NYC?”



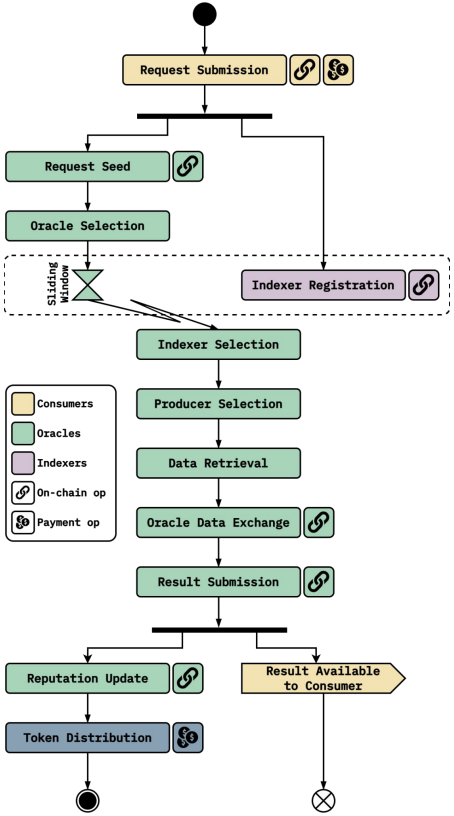
Request resolution flow



?q = “Which is the Co2 level in NYC?”



Request resolution flow



Reputation Update:

- Reputation **updates after each request** resolution
- Oracles and Indexers are rated by **speed, consistency, and accuracy**
- Higher reputation **improves future selection chances**
- Low reputation can lead to **banning**

Request resolution flow

ZONIA WoT Integration

Customized WoT Indexer

1. Blockchain Registration

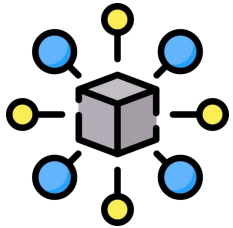
2. Query resolution flow

TDD features

- Indexes **W3C Web Things**
- **JSONPath** discovery
+ **Geospatial** query
+ Blockchain **wallet address**

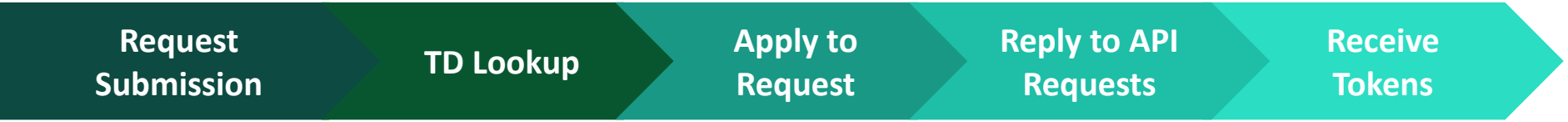
Indexer Registration

Indexer **stakes** an amount of **ZONIA tokens** for be eligible for registration



Register with a **DID** that points to its **wallet address** and **API endpoint**

Indexer in ZONIA Query Resolution Flow



An user submits a data request via the smart contract, specifying the **data type**, **geographical area**, and the **number of oracles** and **indexers** to engage.

All indexers registered received an **event** for that request. They scan their local repository for **matching TDs** and **cache** them into memory with the request ID key.

If an indexer has matching TDs it **applies for the request** calling a smart contract function

If the indexer is selected, **each Oracle** in the query committee will call the indexer API passing the associated request ID. The indexer must **return the cache set of producers**

After the final value is written to the blockchain**.** Indexer are **rewarded** for their participation and their reputation is updated.

Use Case Analysis

Parametric Insurance: Data Collection

10
Km² of area

12
Monitored Crops

8
Weather Stations

- Real-world dataset collected from 8 Milesight WTS506 weather stations
- Weather Stations were deployed on 12 medicinal plant farms in northern Italy
- Data collection started on March 27, 2025

I. Zyrianoff, L. Gigli, M. Di Felice, V. A. Siris, T. G. Papaioannou, and F. Montori, “From field to chain: A blockchain-based automatic insurance framework for smart agriculture,” *presented at the 2025 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, Bologna, Italy, 2025.

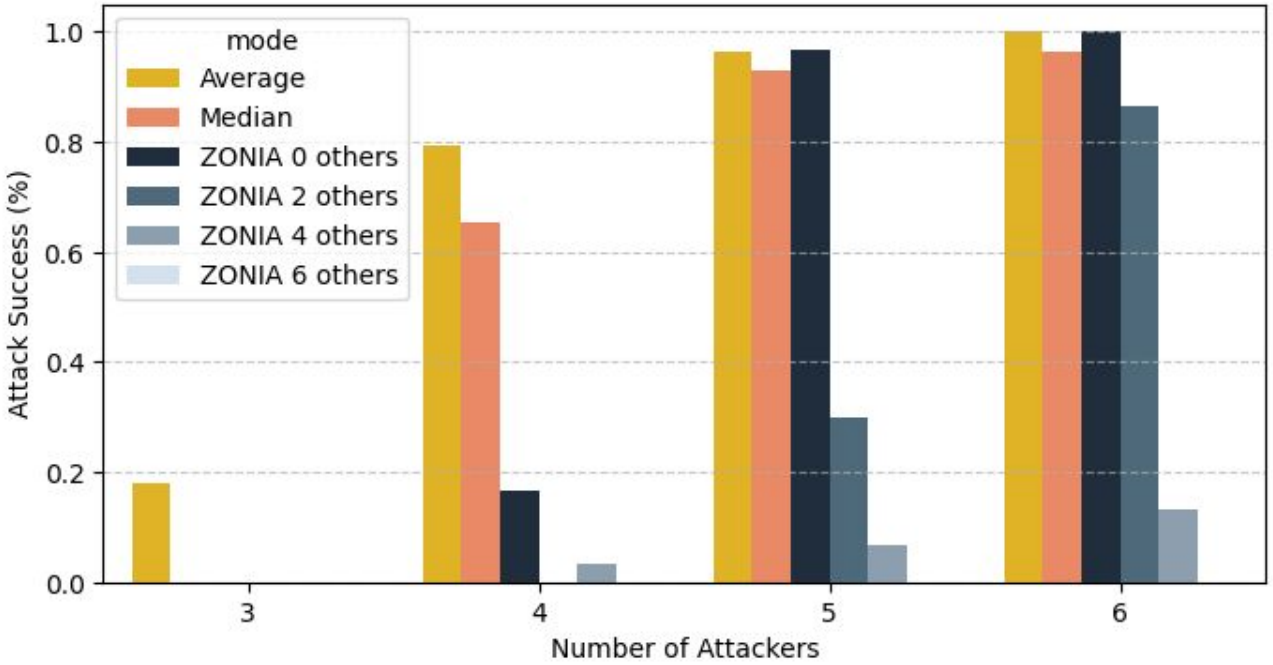
Experiments: Evaluation Scenario

- **Scenario**: A subset farmers data producers collude to report artificially high temperatures (around 80°C) in an to trigger payout
- **Payout Trigge**r: payment is triggered if the temperature < 50°C for 5 consecutive queries
- Experiments were replicated **30 times**
- Baseline for Comparison:
  - **Avg Baseline:** computes the average of reported values
  - **Median Baseline**: computes the median of reported values

Experiments: Virtual Data Sources

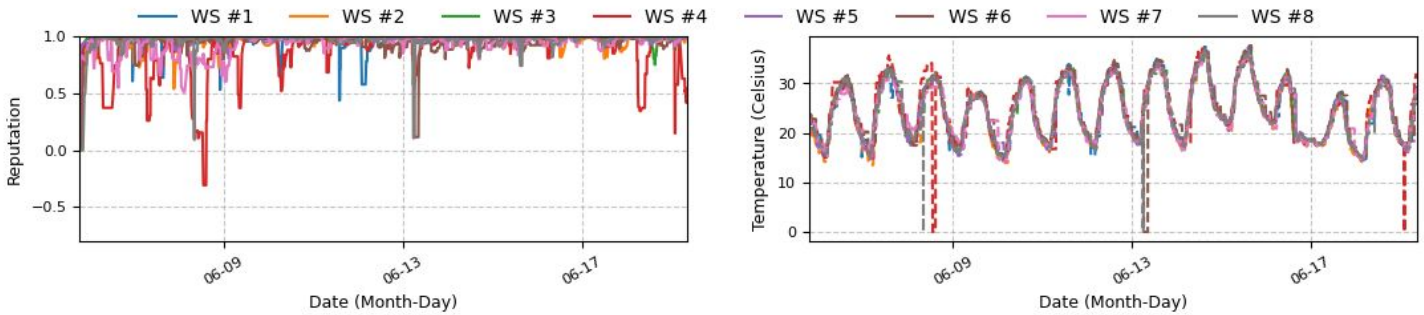
- **Key design**: *decoupling of data sources and clients*
- We introduce **Virtual Data Sources**
  - Represent **honest**, third-party data providers who are not party to the insurance contract
  - Example: *a neighboring farm selling its weather data*
- In the simulation, scenarios were evaluated with **0, 2, 4, and 6** of these **honest** virtual sources

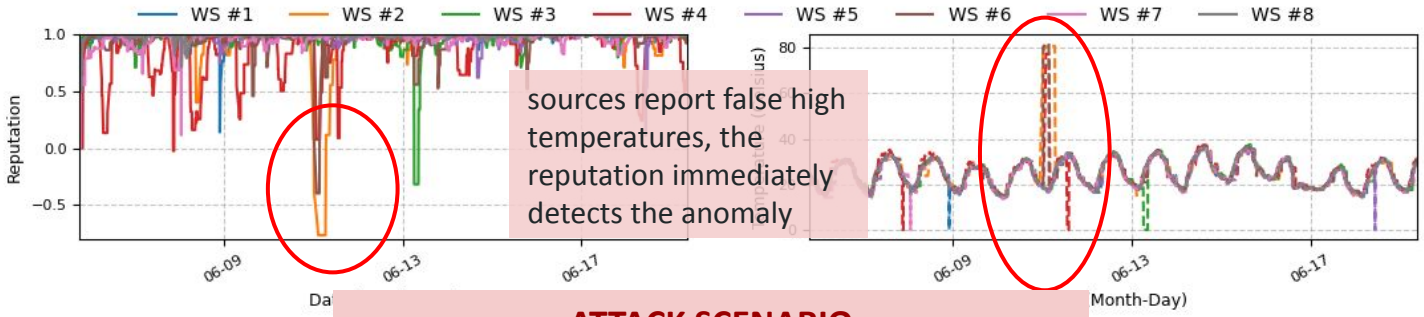
Results: Attack Success Rate



Results: Reputation Dynamics

NORMAL SCENARIO



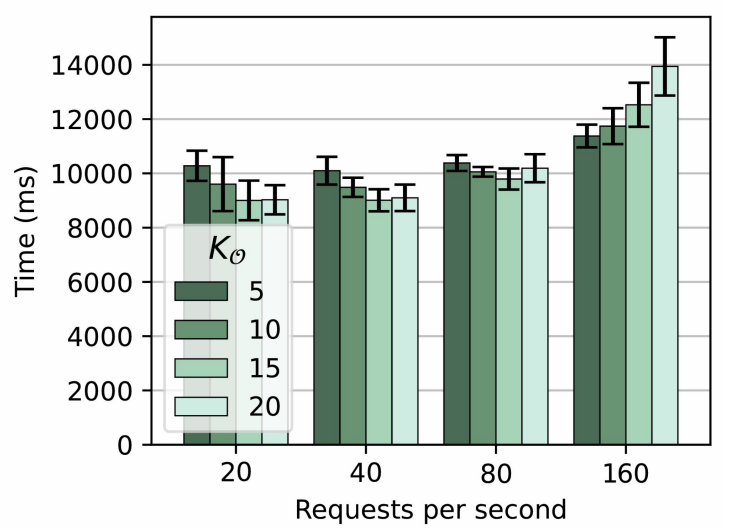


sources report false high temperatures, the reputation immediately detects the anomaly

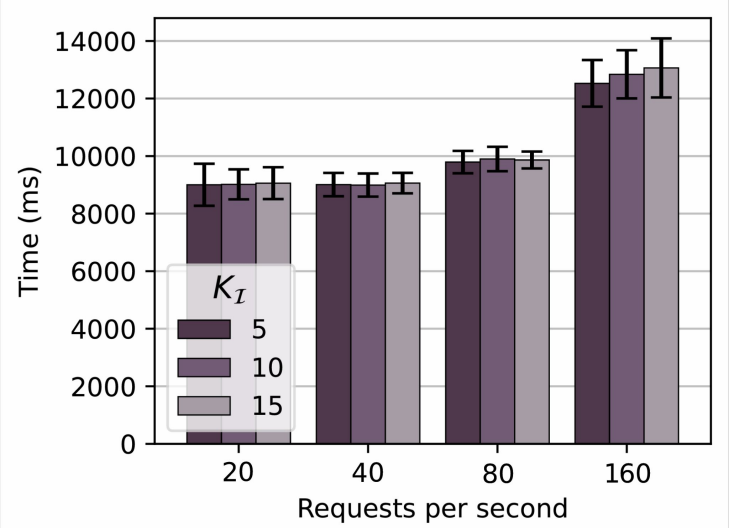
ATTACK SCENARIO

ZONIA Performance

Performance Evaluation



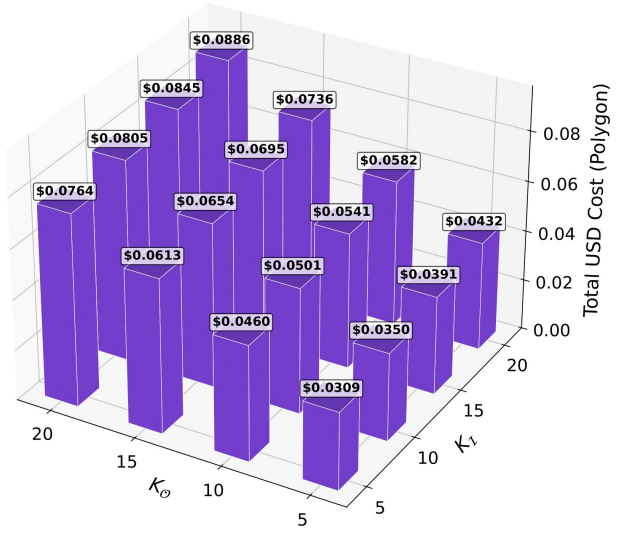
Ko = Number of selected Oracles for a single request

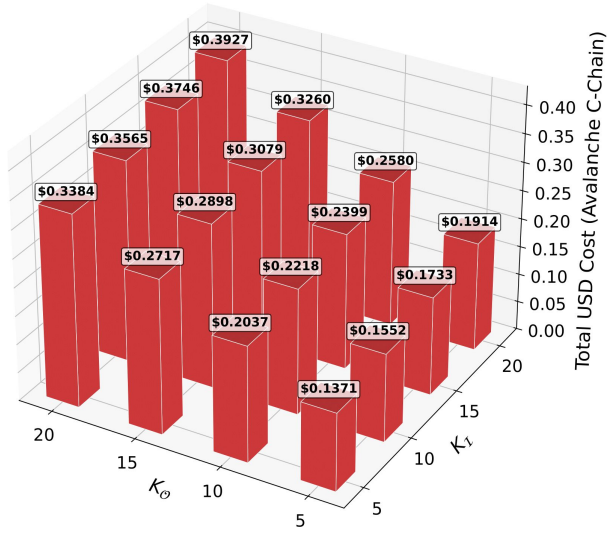


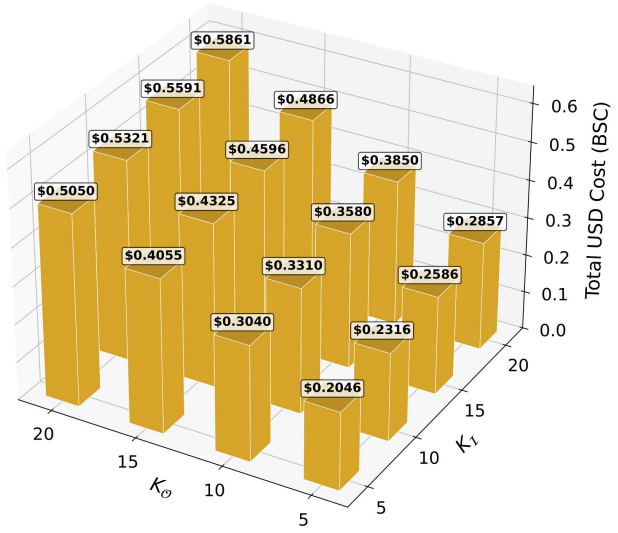
Ki = Number of selected Indexers for a single request

Mean end-to-end latency for different combinations of Ko and Ki

Cost Evaluation In USD







Total gas cost per request in USD as a function of Oracle committee size Ko and Indexer selection size Ki for different chains (Polygon, Avalanche C-Chain, BSC)

Future Works

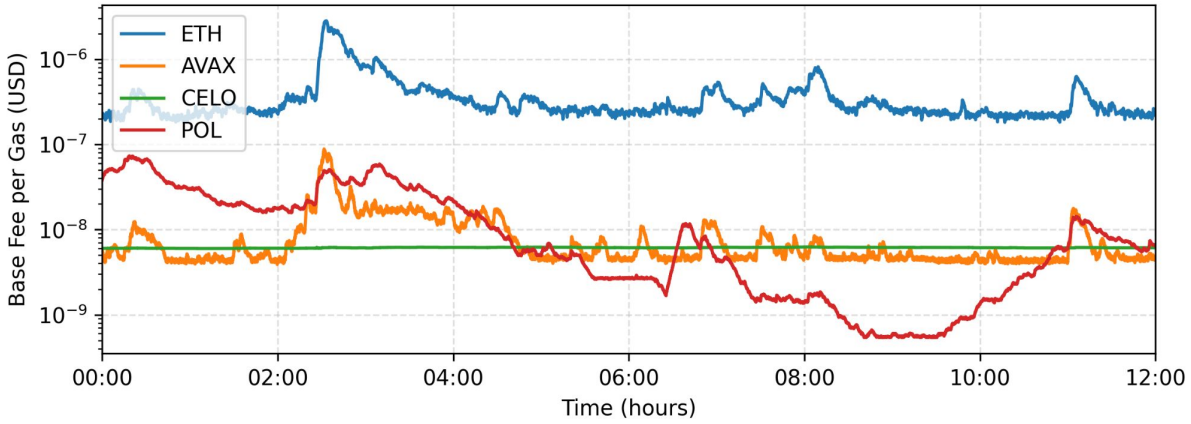
Future Works: Parametric insurance

- We achieved a *Proof-of-Concept*
  - **Real deployment is way more challenging**
- Multidisciplinary research field with many **open challenges**
  - Farmers really want it! ***If devices produce trusted data…***
  - Which are the parameters to agree on? It is crop/climate dependent?
  - What is the minimum number of devices (and owners!)?
  - Etc…
- Current expanding and preparing a submission for an EU Project

Thank you! Questions?

Future Works: Multichain Development

- Transactions cost money
- **Idea**: choose at the **runtime** the cheaper chain to execute the ZONIA
  - Big savings on the long run



Selection Algorithms

Algorithm 1: Oracle selection algorithm, executed by each $O_i \in \mathcal{O}$

Input: Oracle nodes $\mathcal{O}$; request $\mathbf{R}$; parameter $K_{\mathcal{O}}$
Output: Selection of a committee of $K_{\mathcal{O}}$ Oracle nodes

$\mathbf{R}' \leftarrow$ SHA-256($\mathbf{R}$)
$(v_i, p_i) \leftarrow$ VRF($\mathbf{R}'$)
Attempt to write (v_i, p_i) to the blockchain
$(v^*, p^*) \leftarrow$ First successfully written VRF output on the blockchain
for $O_i \in \mathcal{O}$ **do**
    Calculate the normalized reputation $\hat{\vartheta}_i$
    Calculate the pseudorandom r_i using v^* as seed
    Calculate the score SO_i using Eq. (1)
Sort Oracles in $\mathcal{O}$ based on scores SO_i in descending order
Select top $K_{\mathcal{O}}$ nodes from sorted list
return *Selected* $K_{\mathcal{O}}$ *Oracle nodes* $\widehat{\mathcal{O}}$

Algorithm 2: Indexer selection algorithm, executed by each $O_i \in \widehat{\mathcal{O}}$

Input: Indexer nodes $\mathcal{I}$; request $\mathbf{R}$, parameter $K_{\mathcal{I}}$
Output: Selection of $K_{\mathcal{I}}$ Indexer nodes

{ *Each Indexer checks if it has at least* M *Producers that match* $dtype^{\mathbf{R}}$ *and* $geo^{\mathbf{R}}$*, if so it registers to the request, becoming part of* $\mathcal{I}_{reg}$ }
Wait for the sliding window closure, calculated using Eq. (2) and Eq. (3)
for $I_j \in \mathcal{I}_{reg}$ **do**
    Calculate the normalized reputations $\hat{\varrho}_j$
    Calculate the pseudorandom r_j using v^* as seed
    Calculate the score SI_j using Eq. (4)
Sort nodes in $\mathcal{I}_{\text{reg}}$ based on scores SI_j in descending order
Select top $K_{\mathcal{I}}$ nodes from sorted list
return *Selected* $K_{\mathcal{I}}$ *Indexer nodes* $\widehat{\mathcal{I}}$