Matthieu BETTINGER

About me

Matthieu BETTINGER

- INSA-IF 2021
- Double-degree with Uni. Passau

2021 projects:

- *"Collusion-Resistant Worker Set Selection for Transparent and Verifiable Blockchain-Based Voting"* with Lucas Barbero & Omar Hasan
- *"Blockchain interoperability – Cross-chain oracles with Decentralized Cloud Computing"* with iExec, INSA Lyon & Uni.Passau

2022–2024:

- PhD thesis on *"Blockchain interoperability & composability"* in RedChainLab

Summary

- **iExec Decentralized Cloud Computing**
- **Using blockchains to secure applicative protocols**
  - *“Collusion-resistant worker set selection”*
- **RedChainLab**
- **Oracles: Connecting blockchains & the outside world**
  - Theory & Practice
  - Standardization & Interoperability
  - Data provisioning trade-offs
  - *“Oracles with Decentralized Cloud Computing”*
- **Project/Practical work**

Goals

- Real-world applications of blockchain/smart contract technology, how they secure protocols
- Trustworthy data provisioning issues and countermeasures
- Event-based programming with Solidity/Web3J(S)

A Trusted, Decentralized Marketplace for Cloud Computing

By **Anthony Simonet-Boulogne, PhD**
Research Scientist, iExec Blockchain Tech

Revisited by **Matthieu Bettinger**

iExec

INSA Lyon
Dec. 7th 2021

What we do

iExec provides an **open** and **decentralized** cloud computing **marketplace**.

Connects **providers** with **users**: anyone can trade **computing power**, **datasets**, and **applications**.

No need to trust iExec or anyone else: just **trust the blockchain** and the code.

Blockchain & Decentralisation

A decentralised, immutable and verifiable digital ledger consisting of transaction records distributed across many computers.

Decentralised

Immutable

Verifiable

Smart contracts?

Self-executing and self-enforcing programs that can read and write the state of a blockchain.

- ✅ Transparent
- ✅ Auditable
- ✅ Autonomous
- ❌ Hard to program
- ❌ Limited in size
- ❌ Often extremely critical
- ❌ **Cannot access external data**

```
contract Coin {
    address public minter;
    mapping (address => uint) public balances;

    function Coin() public {
        minter = msg.sender;
    }

    function mint(address receiver, uint amount) public {
        if (msg.sender != minter) return;
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) public {
        if (balances[msg.sender] < amount) return;
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
    }
}
```

Blockchains

What they are

- ✅ A way for arbitrary people and organisations to collaborate without having to trust anyone.
- ✅ A tool for transparency and democracy.
- ✅ A platform for deploying unstoppable programs.

What they (usually) are not

- ❌ Fast
- ❌ Cheap
- ❌ Easy to program
- ❌ User friendly

But we’re working on it ;-)

iExec history

Founded in 2016 by Gilles Fedak (Inria) & Haiwu He (Chinese Academy of Sciences)

April 2017: ICO raised 10,000 Bitcoins within 3 hours

Based in Lyon

A not so new idea...

We described a computational model based upon the classic science-fiction film, ***The Blob: a program that started out running in one machine, but as its appetite for computing cycles grew, it could reach out, find unused machines, and grow to encompass those resources.*** *In the middle of the night, such a program could mobilize hundreds of machines in one building; in the morning, as users reclaimed their machines, the "blob" would have to retreat in an orderly manner, gathering up the intermediate results of its computation. (This affinity for night-time exploration led one researcher to describe these as "vampire programs.")*

(John F. Shoch and Jon A. Hupp, 1982)

iExec allows individuals and enterprises to monetize their computing power, applications and datasets.

Why decentralise the cloud?

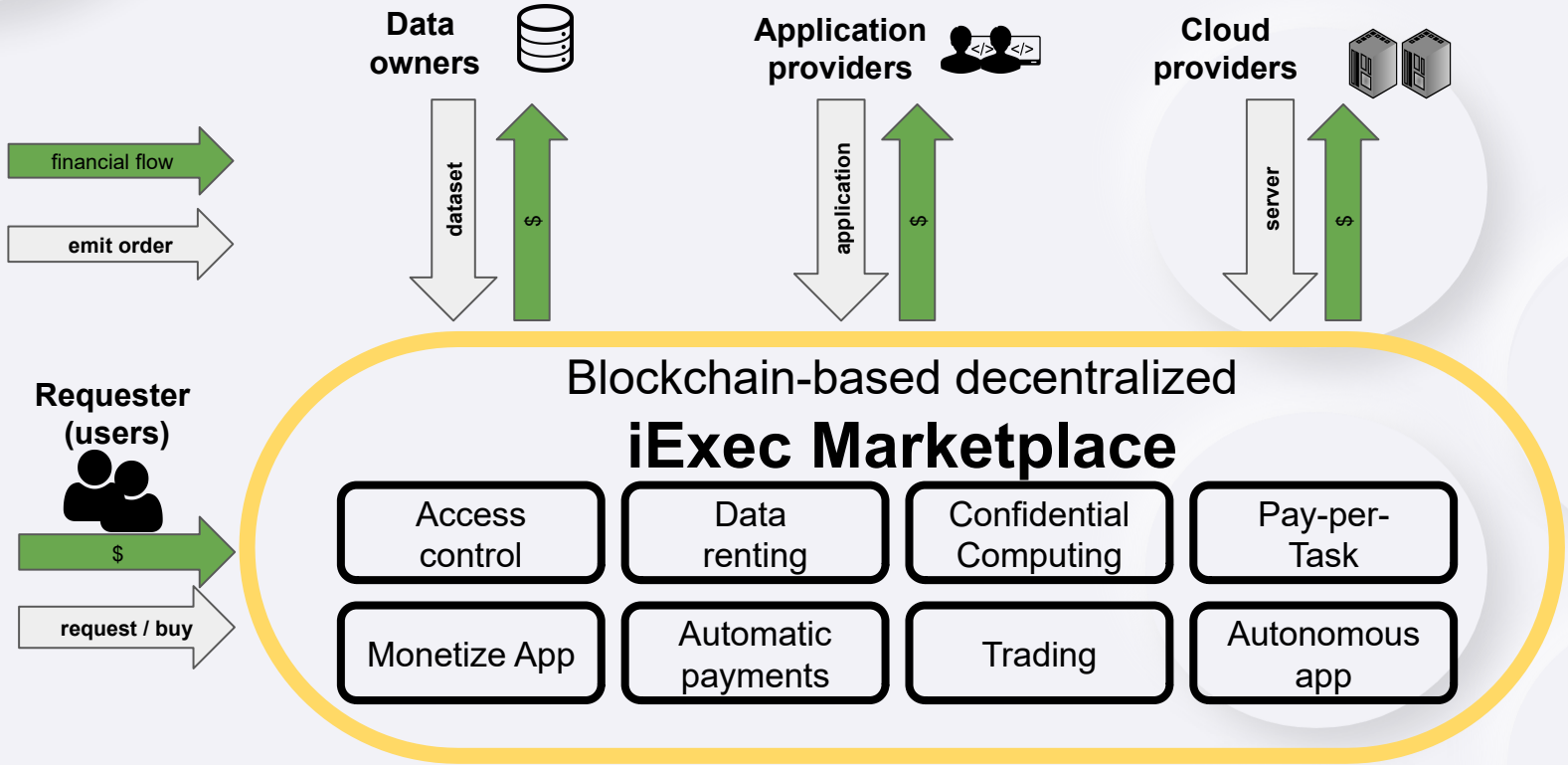
Centralized Computing

- ❌ Unfair pricing
- ❌ Vendor lock-in
- ❌ Limited transparency
- ❌ Limited accountability
- ❌ No provenance information
- ❌ Possible censorship

Decentralised Cloud Computing

- ✅ Market-based prices
- ✅ Fair competition between providers
- ✅ Smooth business agreements
- ✅ Complete execution history on the blockchain
- ✅ *Unstoppable* marketplace: censorship is impossible

iExec overview



The iExec token: RLC

RLC is an ERC-20 utility token.

- ➜ RLC is necessary to access the iExec decentralised cloud
- ➜ Providers are paid with RLC
- ➜ RLC allows to build incentives in the network
- ➜ RLC creates a specific market for cloud computing

Two types of tasks

with configurable confidence and privacy

Standard tasks

Run on untrusted resources, delegate trust to the blockchain

- Replication level depending on desired confidence
- Decentralized consensus
- On-chain reputation
- Staking & economic incentives
- Deterministic

TEE tasks

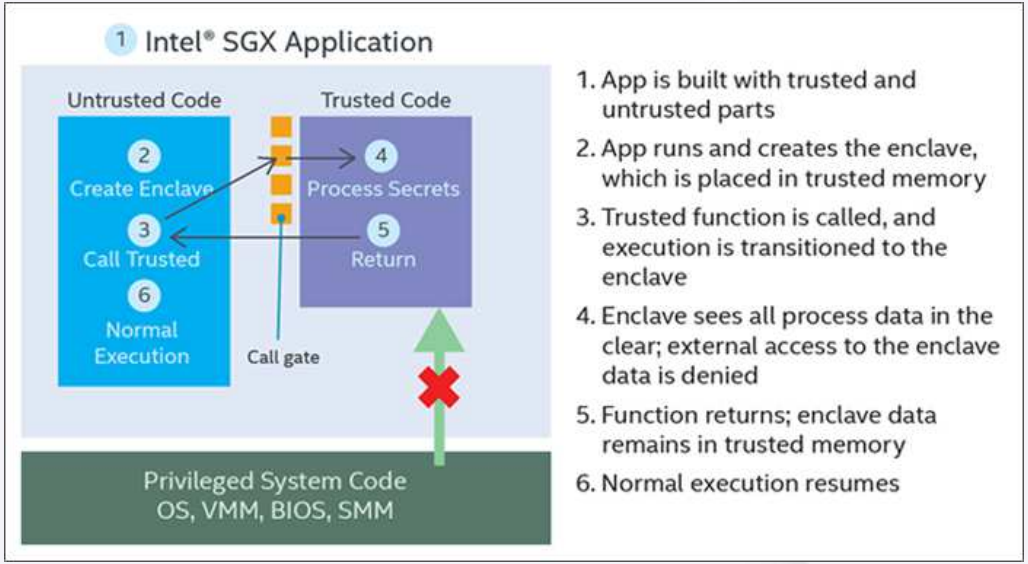
Run isolated within an Intel SGX TEE (Trusted Execution Environments)

+

- End-to-end encryption of data & result
- Enclave attestation proves that the task was run in TEE
- Result signature with enclave key: no need for replication
- Determinism not required

Trusted Execution Environments?

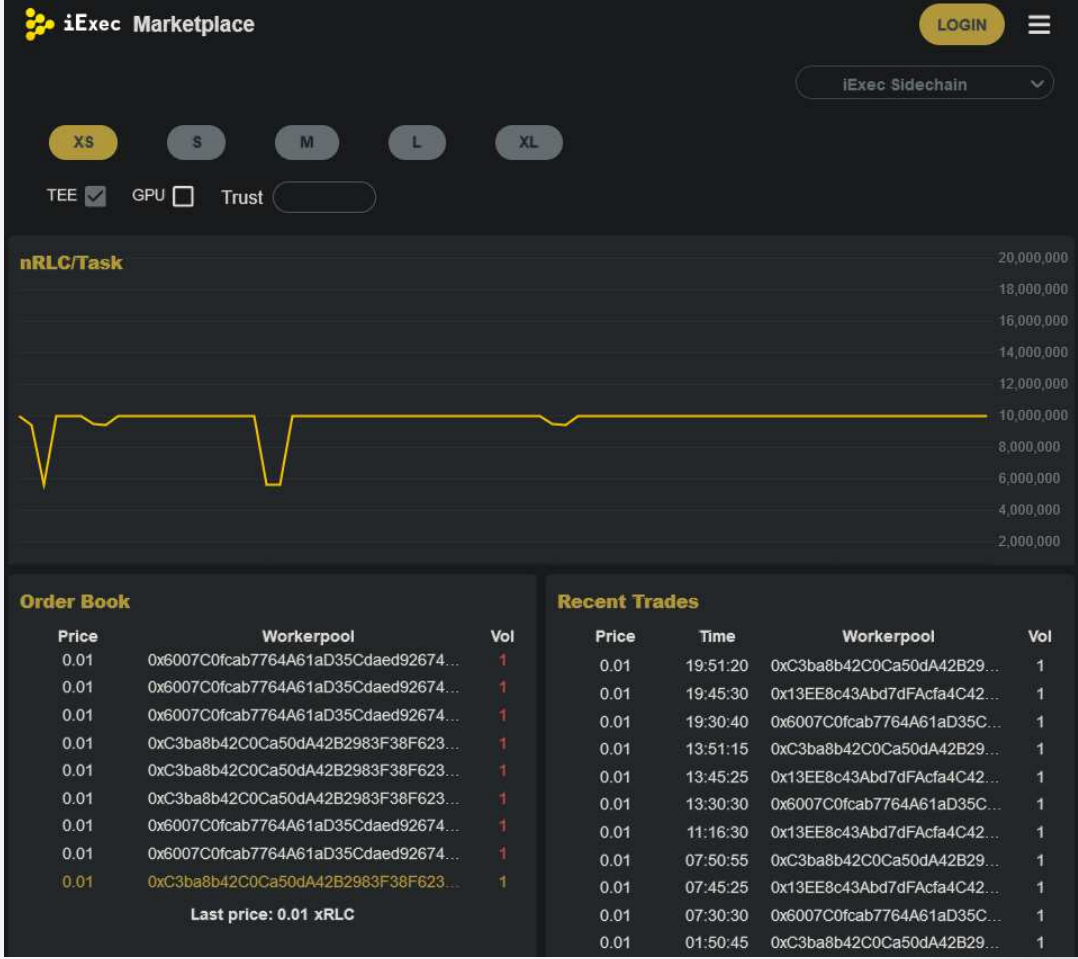
- Secure part of a CPU with encrypted memory space
- Memory & Code protected from host (even root)
- Hardware based security (private key in silico)
- Can be remotely attested
- (Soon) available on hardware from various vendors



Intel® Software Guard Extensions application execution flow.

Components: iExec hub

- ➜ Managed by Smart contracts
- ➜ Repository of registered resources (dApps, workerpools, ...)
- ➜ Storage of task results and metadata
  - Task specifications
  - Execution details
  - Off-chain storage link



Components: Workerpool

- Composed of a scheduler and multiple workers

- Incentives:
  - Staking
  - Reputation

- Scheduler objective:
  - Listen to incoming work
  - Distribute work fairly among workers

- Worker objective:
  - Correctly execute tasks given by the scheduler

Components:
Secret Management Service

- ➜ Secured inside in a Trusted Execution Environment (not yet, but in V7)

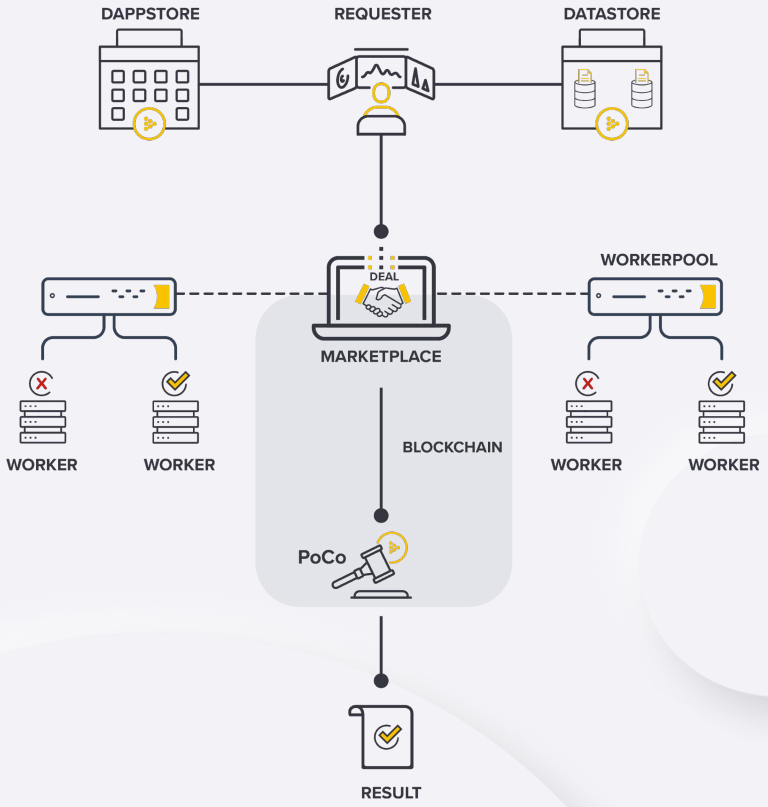
- ➜ Stores secrets of stakeholders:
  - Dataset decryption keys
  - Input files decryption keys
  - Output files/Results encryption keys

- ➜ Attests TEE workers

Proof-of-Contribution (PoCo)

On-chain validation than an off-chain task was performed correctly

1. One task = 4 orders, signed off-chain with an Ethereum wallet:
   - `apporder` signed by the developer
   - (`datasetorder` signed by the dataset provider)
   - `workerpoolorder` signed by a worker pool scheduler
   - `requestorder` signed by a requester
2. Orders are matched on-chain: [poco.matchOrders()] (Check signatures, parameters, balances, …)
3. PoCo seals a deal & workers start computing
4. Workers send result hash back to PoCo
5. PoCo compares results, manages reputation, triggers payments.



Adjusting trust: Sarmenta Voting

- ➜ Deterministic execution replicated **N** times: only one correct result exists
- ➜ Given **R** distinct results (1≤**R**≤**N**):
  - For each result **r**, obtained by **n** among **N** workers:
    - Probability that **r** is correct and all others are false
- ➜ User-defined threshold **t**:
  - If no proposed result has Crt(**r**)>**t**
  - Then more workers are dispatched

$$Cr_t(r) = \frac{\tilde{P}_t(r)}{1 + \sum_{h \in R_t} \tilde{P}_t(h)}$$

https://github.com/iExecBlockchainComputing/PoCo/blob/v5/audit/docs/iExec_PoCo_and_trustmanagement_v1.pdf

iExec Research Projects

H2020 ONTOCHAIN

Building an ecosystem for trustworthy content handling & information exchange

Keywords: Semantic Web, Oracles, Decentralized Identities, integration, applications

2020−2023

H2020 DATACLOUD

Enabling the Big Data Pipeline Lifecycle on the Computing Continuum

Keywords: Fog/Edge Computing, Big Data pipelines, self-* cloud computing, Industry 4.0

2021−2024

ANR RedChainLab

Scalable, trusted and privacy preserving decentralized marketplaces

Joint laboratory between the **DRIM research team** (LIRIS, CNRS) and **iExec**

Keywords: blockchain, decentralized cloud computing, edge computing, security, TEE, Federated Learning

2021−2024

Try us!

https://developers.iex.ec/

https://iex.ec/grants/

Join us!

https://iexec.flatchr.io/

https://gitcoin.co/explorer?network=mainnet&idx_status=open&applicants=ALL&key

iExec

Anthony Simonet-Boulogne
anthony@iex.ec

Collusion-Resistant Worker Set Selection for Transparent and Verifiable Blockchain-Based Voting

Matthieu BETTINGER
Lucas BARBERO
Omar Hasan

Context

- **Schiedermeier's blockchain-based voting protocol (2020)**
  - P participants, and W workers chosen among them
  - Workers compute the referendum's result
    - Using SMPC:
      - Shamir's Secret Sharing scheme
      - Homomorphic encryption
  - All messages are recorded on-chain

Context

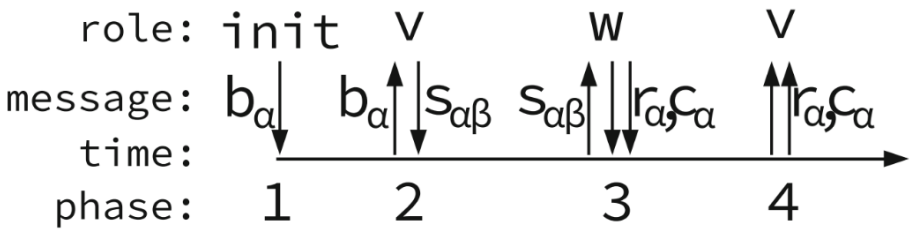
- However: workers are chosen by a single trusted entity

  → Risk of collusion

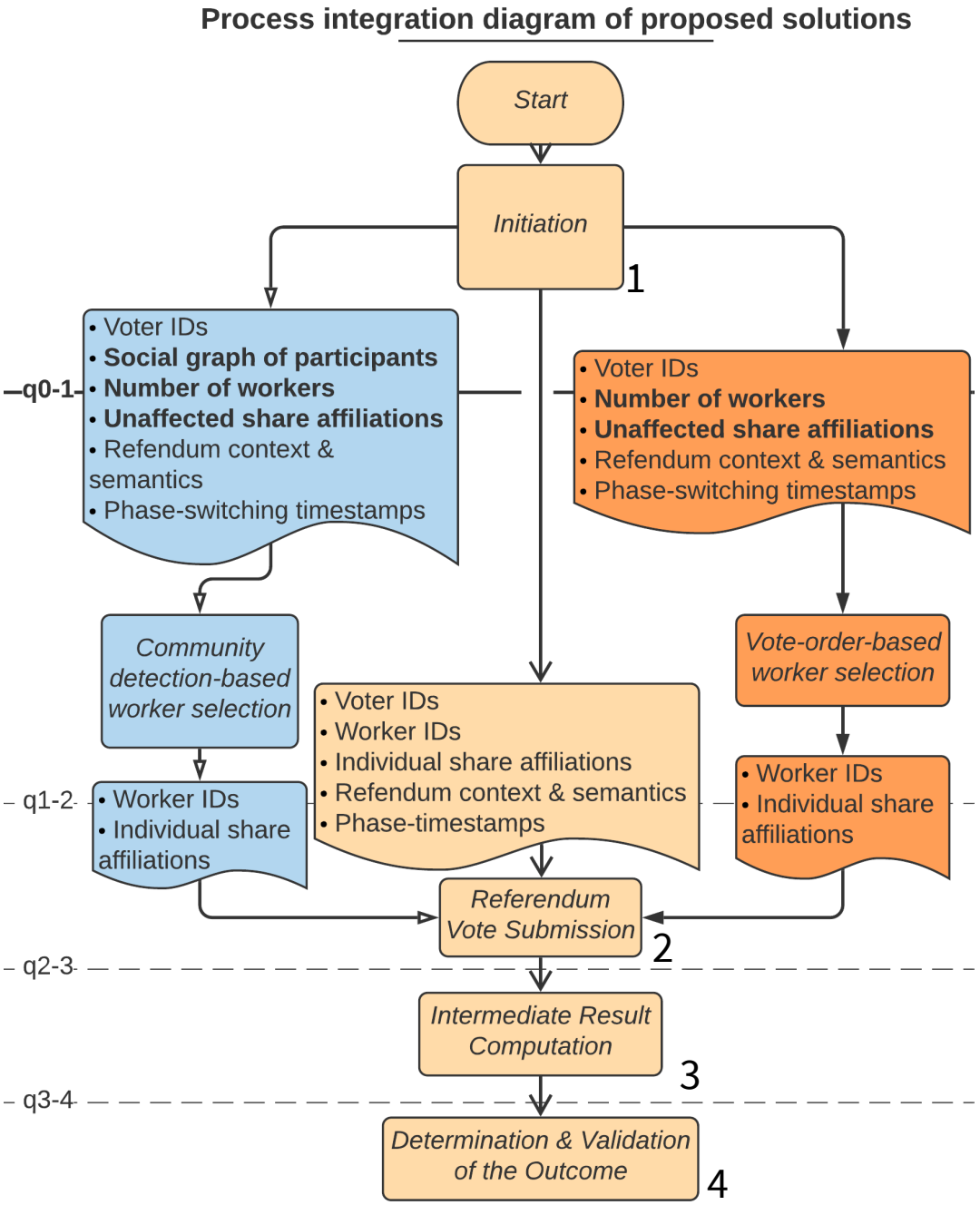
- Goal:

  → Collusion-resistant & verifiable selection of W workers

Process



Schiedermeier et al. (2019)



Bettinger et al. (2021)

Proposal 1:
Verifiable random number generation

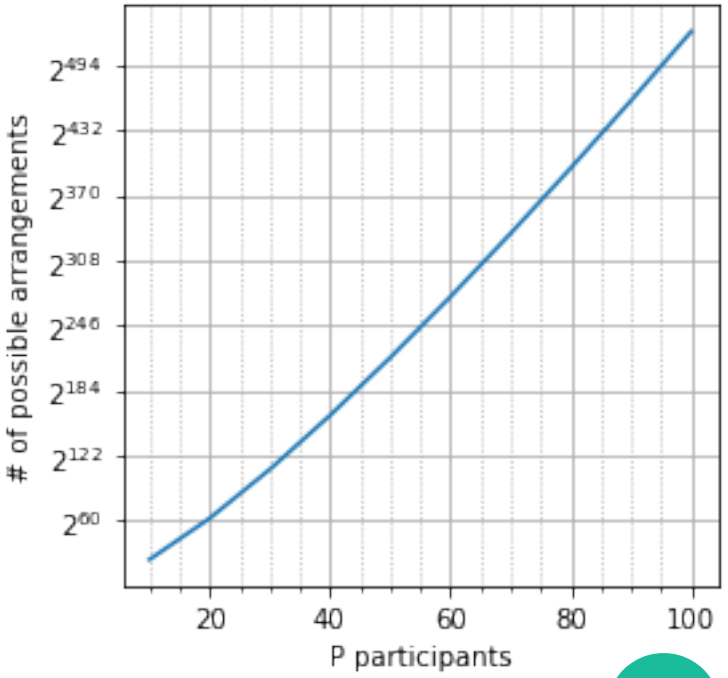
Table 1 Three methods to attribute a number to a list V of comparable non-reoccurring elements (integers, character strings,...) knowing its superlist P.

Numbering Method	*Expression*	*Output Space*
Permutation $PN : V \to \mathbb{N}$	$\sum_{i=0}^{\lvert V\rvert-1}(i!\sum_{k=0}^{i-1}\mathbb{1}_{x_k<x_i})$	$[\![0;\lvert V\rvert![\![$
Combination $CN : V, P \to \mathbb{N}$	$\sum_{k=0}^{j(0)-1}\binom{\lvert P\rvert-j(0)+k}{\lvert V\rvert-1}$ $+\sum_{i=1}^{\lvert V\rvert-1}\sum_{k=0}^{j(i)-j(i-1)-2}\binom{\lvert P\rvert-j(i)+k}{\lvert V\rvert-i-1}$	$[\![0;\binom{\lvert P\rvert}{\lvert V\rvert}[\![$
Arrangement $AN : V, P \to \mathbb{N}$	$\sum_{i=0}^{\lvert V\rvert-1}(\frac{\lvert P\rvert!}{(\lvert P\rvert-i)!}+CN(V,P)*\lvert V\rvert!+PN(V))$	$[\![0;\sum_{v=0}^{\lvert P\rvert}A_{\lvert P\rvert}^{v}[\![$

- **PN : Ordering of *V* voters**
- **CN : Which subset of *V* participants voted**
- **AN : PN & CN + Sum(AN with fewer voters)**

10 participants : 10^6 possibilities

100 participants : 10^{158} possibilities

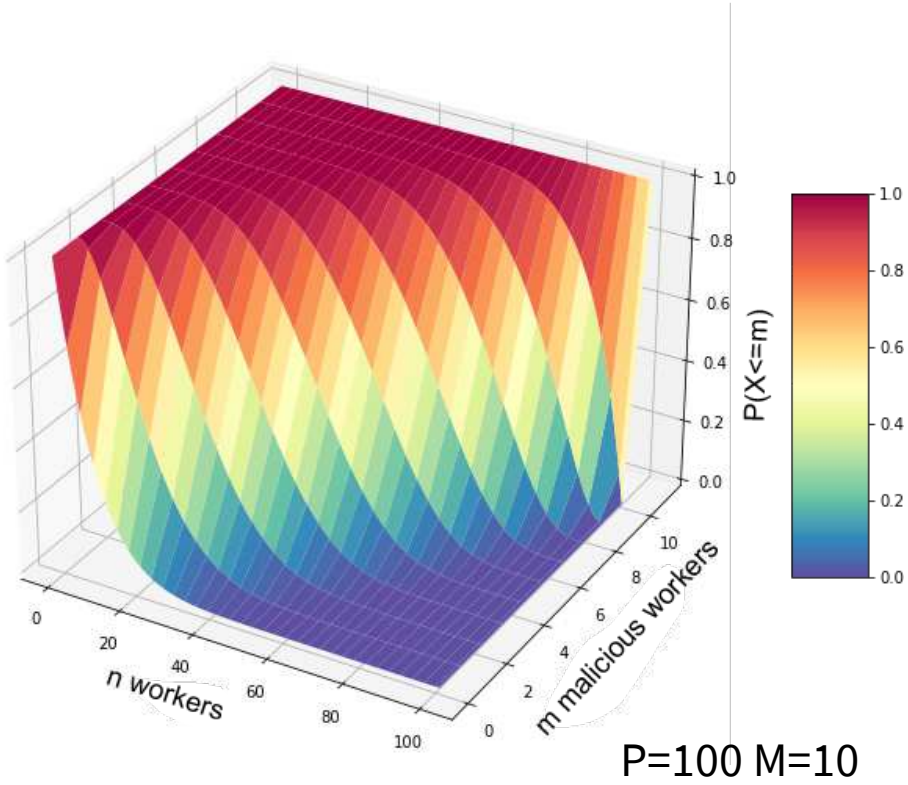


Proposal 1:
Probability Distribution

- **Hypergeometric Law**

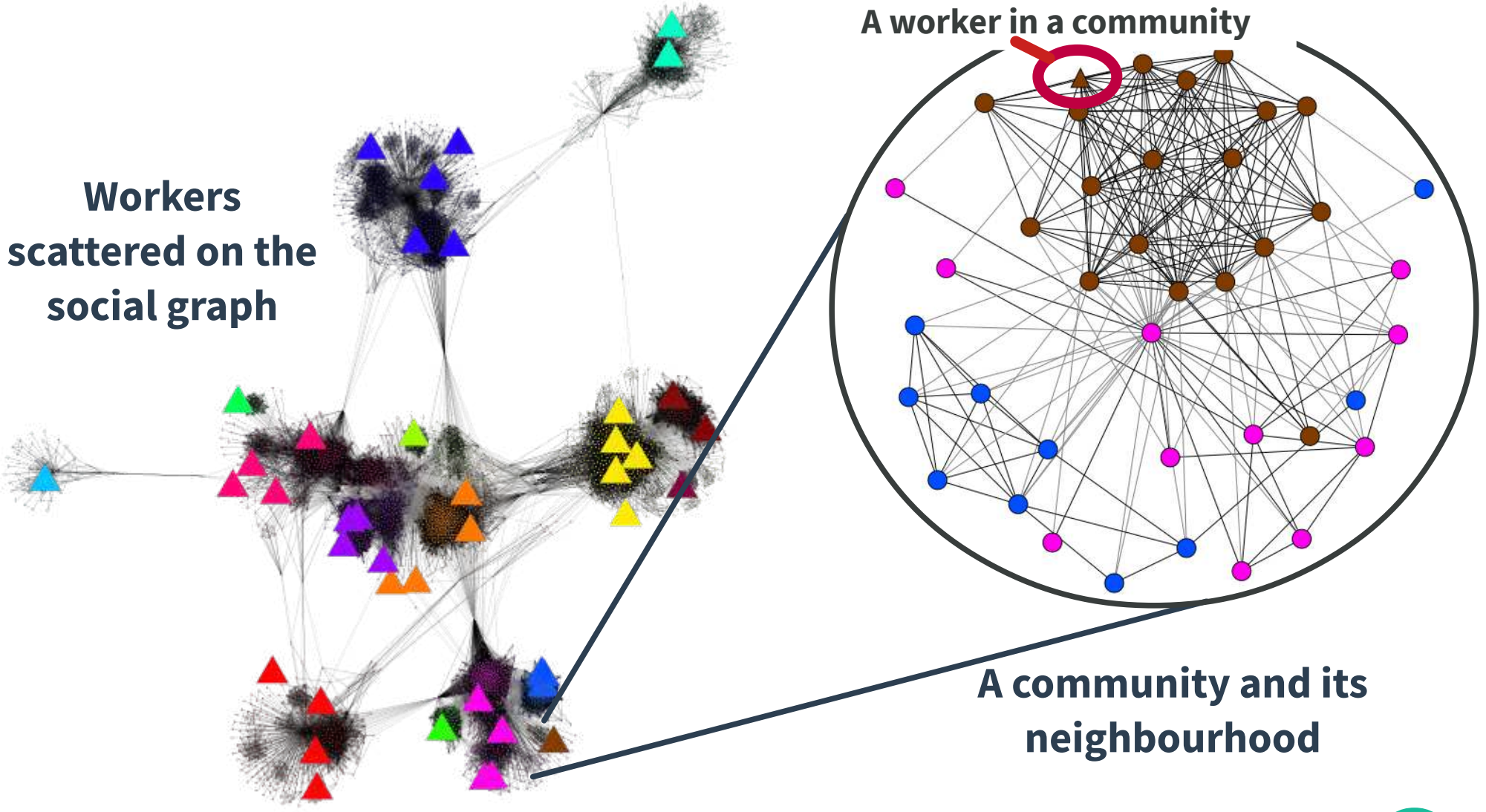
  H(n, M/P, P)

  - P participants
  - M ≤ P malicious participants
  - n workers
  - m malicious workers

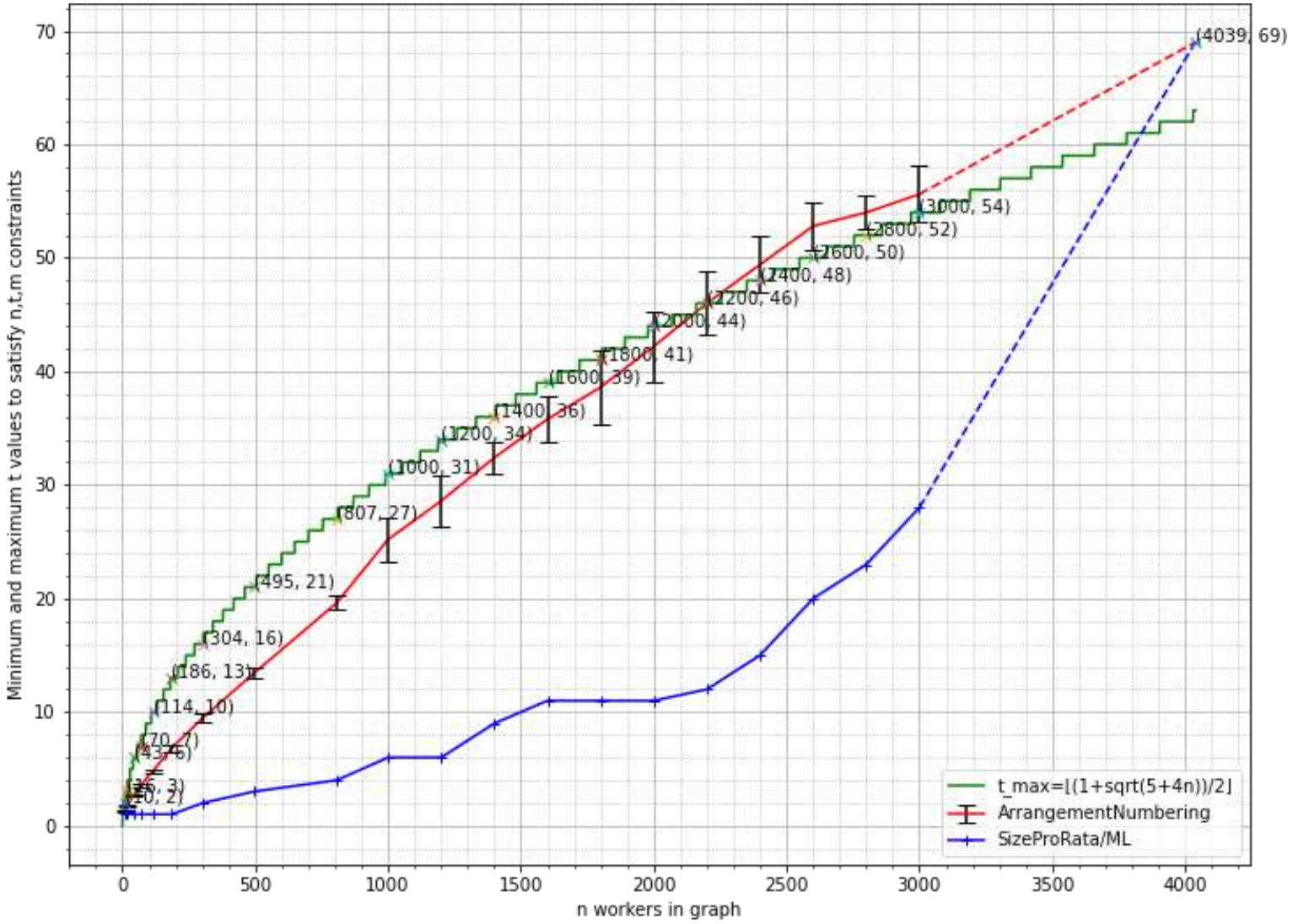


P=100 M=10

Proposal 2:
Worker distancing



Results:
Cliques of adjacent workers



Application to iExec's worker selection

- **Scheduler: trusted to distribute work fairly**

  → Proposed solutions can be used

- **Difference with Schiedermeier's:**
  - iExec workers perform multiple tasks
  - versus anonymized participants for each vote

Application to iExec's worker selection

- **Aral et al. (2020)**
  - Use of workers' task execution history
  - Clustering of workers that fail together

  → Algorithm to maximize success probability

RedChainLab

iExec - DRIM

Matthieu BETTINGER

RedChainLab

- **Joint laboratory**
  - Between the LIRIS-DRIM team and iExec
  - Coordinated by Sonia BEN MOKHTAR (BFT, Distributed Systems)

- **From 2022 to 2025**

  ~2 end-of-studies projects per year

Research fields

- Blockchain Scalability
- Trusted Execution Environments
- Confidentiality
- Genericity/Interoperability/Composability

Oracles between blockchains and the outside world

Matthieu BETTINGER

Summary

- Motivation
- Oracles: Theory and practice
- Standardization & Interoperability
- Data provisioning trade-offs

Motivation

- **Blockchains are closed systems**

  → they cannot read/write data of off-chain systems

  → off-chain entities provide data

  → Need for data provisioning systems

- **Use cases for smart contracts in IoT (Industry 4.0, Smart Cities,...), Education, Health and more**
  - Relying on critical off-chain data or events

    → Need for ***trustworthy*** data provisioning systems

Oracles

- **Link blockchains to off-chain data and events (even to other blockchains) and vice-versa**

- **Two components:**
  - Off-chain: data collection, processing, transmission wrt. requests
  - On-chain: interfaces requesting/accepting data + storage

- **The proportion of on/off-chain computation depends on the architecture**

Types of oracles (Mühlberger et al. 2020)

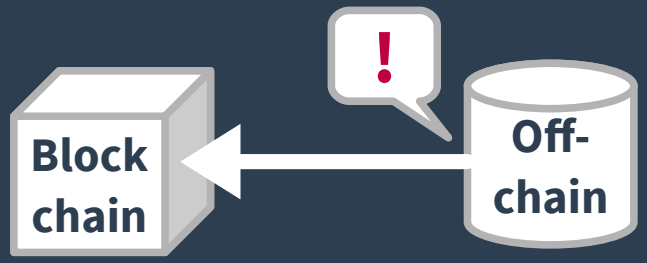
- **Is it the data recipient who requests the data transfer ?**
  - Yes → Pull-based
  - No → Push-based (another entity is responsible)

- **What is the data transfer path ?**
  - Blockchain to outside world → Outbound
  - Outside world to blockchain → Inbound

Example for showcasing oracle types

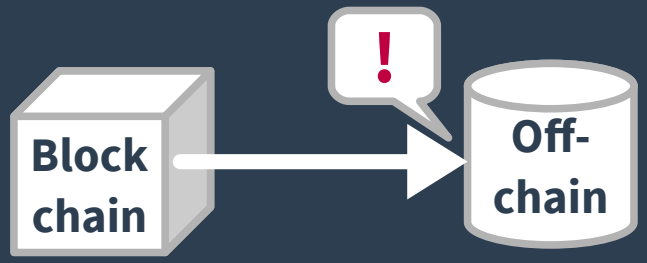
- **Blockchain-based betting platform**
  - Users populate events (e.g. sports matches) on-chain
  - Any interested user can lookup current betting opportunities and bet on them
  - The platform requests the results from a trustworthy oracle system
  - Once a result is known, the platform broadcasts winners of the corresponding bet

Push-inbound oracles



- **Goal: Feed relevant off-chain data to the smart contract**
- **Behaviour:**
  - Off-chain states/events are monitored
  - EOAs (Externally Owned Accounts) send transactions
  - Passive recipient smart contract
- **Example:**
  - Add a sports event on which to bet in the platform

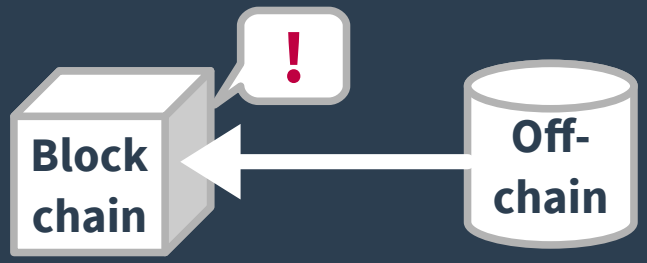
Pull-outbound oracles



- **Goal: Retrieve relevant data from the smart contract**
- **Behaviour:**
  - Queries: off-chain systems read the state of the contract
  - Passive data source smart contract

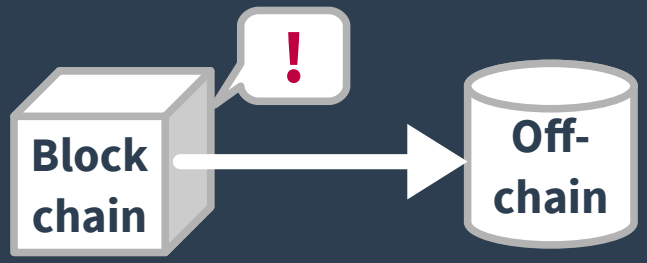
- **Example:**
  - Lookup current standings for a bet

Pull-inbound oracles



- **Goal: Receive relevant off-chain data**
- **Behaviour:**
  - Events (+ requests stored in the smart contract's state)
  - Active smart contract (still needs an EOA to initiate the request's creation)
- **Example:**
  - Request the result of a sports match (if available)

Push-outbound oracles



- **Goal: Signal relevant chain data to offchain systems**
- **Behaviour:**
  - Events (+ requests stored in the smart contract's state)
  - Active smart contract (still needs an EOA to initiate the request's creation)
- **Example:**
  - Broadcast winners of a bet

Real-world oracle systems

- **ChainLink V1: On-chain data aggregation**

  → Provides predetermined feeds of temporal data (e.g. ETH-USD exchange rate)

  - Off-chain nodes periodically fetch new values from an off-chain source
  - Each node pushes its findings on-chain
  - Values are aggregated on a given metric (avg, med, …)
  - Outlier values' nodes are punished

- **Pull-inbound oracle (for periodic updates)**
- **Push-inbound oracle (for value-adaptative updates, e.g. based on relative change)**

Event-based programming

- **In Solidity:**

  event NamedEvent(Type$_1$,T$_2$,T$_3$,...); → declaration in class' body

  emit NamedEvent(param$_1$,p$_2$,p$_3$,...); → call in a function's body

- **Event parameters (up to three) can be *indexed* for easier filtering**

- **In Java/web3j:**

  contractInstance.NamedEventFlowable(...).subscribe(event → { /*event handling*/ })

- **In JS/web3js:**

  var namedEvent = constractInstance.NamedEvent();

  namedEvent.watch(function(error, result){ /*event handling*/ });

Standardization

- **EIPs: Ethereum Improvement Proposals**
  - EIP-1154: push-inbound & pull-outbound oracles
    - Abandoned by the community (but still used by iExec)
  - EIP-2362: interface for pull-based price-feed oracles
    - By the Alliance of Decentralized Oracles

      
    - In reaction to the unpopularity of EIP-1154

Interoperability

- **Increasing difficulty**
  - Standard < Manual interop. < Automated interop.

- **Interoperating entities:**
  - smart contracts (SCs) in the same blockchain
  - SCs and oracle systems
  - SCs across distinct blockchains (through oracles)

On-chain VS Off-chain computations

- **Oracle systems have both on- & off-chain parts**

- **On-chain:**
  - Greater security/decentralization → Greater costs/delays
- **Off-chain:**
  - Less costs but loss of blockchain properties

→ Balance of costs and security to be found

On-chain VS Off-chain computations

- **ChainLink V2: Off-Chain Reporting**
  - Rounds of an off-chain BFT algorithm
  - A selected leader generates the new value to report
  - Other nodes agree by signing the report
  - The agreed-upon report is pushed on-chain

Other data acceptance mechanisms

- **Astrea: Voting**
  - A proposition is submitted on-chain
  - Participants check whether the proposition is true or false
  - Participants vote whether the proposition is true or false
  - After a timeout or sufficient votes, voting is closed
  - The minority loses its staked assets

Other data acceptance mechanisms

- **DOS (Decentralized Oracle System): Random selection**
  - **N** participants are randomly dispatched in small groups of size **n**
  - All nodes in a group compute the result of the same task
  - **t** or more out of **n** must sign the same result to attest it
  - This signed result is pushed on-chain
  - Only signers are rewarded

Blockchain interoperability

—

Cross-chain oracles

with

Decentralized Computing

Matthieu BETTINGER

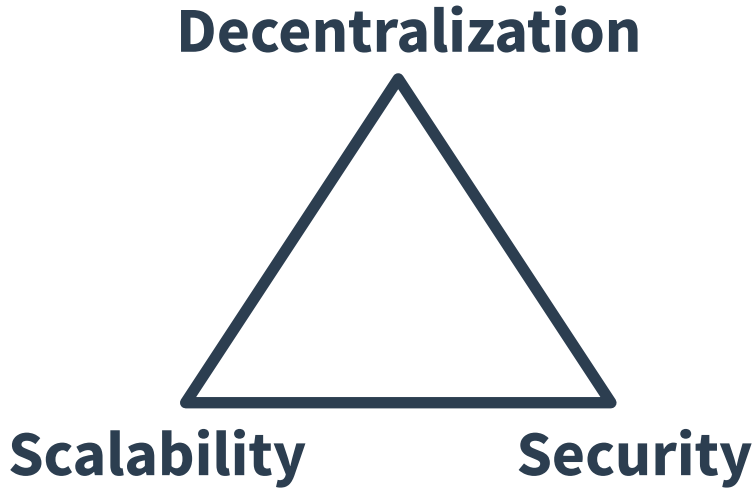
Summary

- Context & Motivation
- State of the Art
- Proposed solutions
- Results & Evaluation
- Conclusion & Future work

Context & Motivation

Blockchain trilemma:

- No "silver bullet"



- **Complex user needs**
  - → Integration of distinct blockchain solutions
  - – Oracles linking blockchains
- **User needs apply on oracle specs as well**
  - → Need flexible oracle architectures

Context & Motivation

- **Generic oracle use-cases**
  - Data queries (request-handling)
  - Triggers (event-handling)

  } → Cross-chain oracles

  - Cross-chain calls → Cross-chain smart contracts
- **Non-functional properties**
  - Data integrity & availability
  - On/Off-chain verifiability
  - Scalability (throughput, costs)

Context & Motivation

- **Variety of use-cases and non-functional requirements by end-users**
  - For blockchain systems
  - For oracle systems

  → Use of decentralized cloud computing infrastructures for flexibility

State of the Art

- **Datafeeds:**

  → Feed data to blockchains

+ Support all oracle use cases

– Rigid architectures

State of the Art

- **Cross-chain bridges**

  → Relay information between chains

+ Scalability, data integrity & availability

– Specialized in requested generic data transfers, no user-defined logic

– (usually) Single data acceptance mechanism

State of the Art

- **Decentralized cloud computing infrastructures:**

  → Execute arbitrary provided logic: a decentralized app

  + Sandboxed architectural freedom

  – Administrative & Genericity-related overheads

- **Objective:**

  → Enable flexible datafeeds & bridges with generic DCCI

Context & Motivation

- **Generic oracle use-cases**
  - Data queries (request-handling)
  - Triggers (event-handling)

  } → Cross-chain oracles

  - Cross-chain calls → Cross-chain smart contracts
- **Non-functional properties**
  - Data integrity & availability
  - On/Off-chain verifiability
  - Scalability (throughput, costs)

Research questions

- **Secure data transfers using DCCIs**
  - data integrity, on/off-chain verifiability
- **Flexibility of oracle system properties**
  - data acceptance, scalability, decentralization
- **Cost reduction strategies**

Proposal 1: Proxy tasks

+ ***Cost reduction:***

  → *Blockchain trust algorithm (PoCo) delegated on a fast and cheap chain*

  → *Storage reduction: Only results and traceability metadata stored on the costly chain*

+ ***Code & Data integrity*: TEE to securely generate & transfer results**

Inherited from iExec infrastructure:

– ***Usability; Storage costs*: Inefficient untyped blob storage for results**

– ***Reactivity; Throughput*: Single result per task**

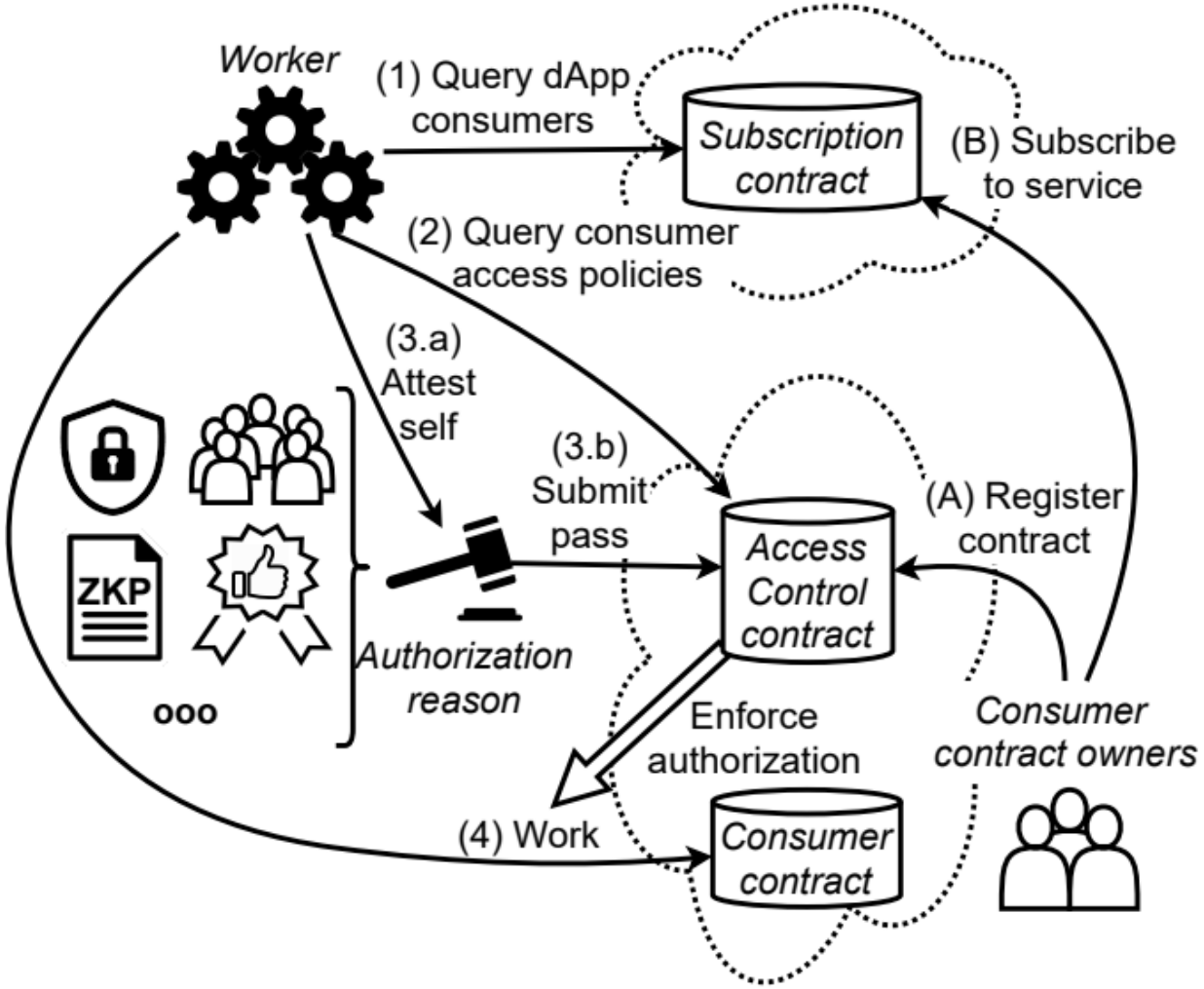
Proposal 2: Work passes
Basics

Access control lists (RBAC):

- A pass links:
  - 1 off-chain entity (e.g. an iExec worker)
  - 1 contract (or a subset of methods)
  - 1 scope of authorization
    - OWN > DELEGATE > WORK > NONE
    - For a limited time

Proposal 2: Work passes Workflow

- **Authorization reason**
  - Attested TEE
  - Committee consensus
  - (ZK) Proof
  - Trusted identity
  - …



Proposal 2: Work passes
Protocol properties

+ *Reactivity; Throughput*: **Multiple results per dApp execution**

+ *Flexible security*: **Consumer-defined data acceptance logic**

+ *Usability; Storage costs* : **Application-specific typed storage**

– *Security risk*: **Direct access to consumer methods**

→ Intermediate dApp-specific storage as buffer

– *Storage costs*: **A pass per consumer contract**

Proposal 2: Work passes
Protocol optimizations

- **One-time passes**

  + No persistent storage costs

  – Repeated authorization verification

- **Per-dApp passes**

  → 2 passes on-chain : Contract-Service & Service-Worker

  + Only one pass per worker per dApp per chain

  – Additional storage costs + pass linking costs

Results:
General Metrics

- **Full PoCo task execution with TEE worker :**

  → ~5 minutes of non-productive processing

  Proxy tasks VS current iExec tasks :

  → –64 % gas costs

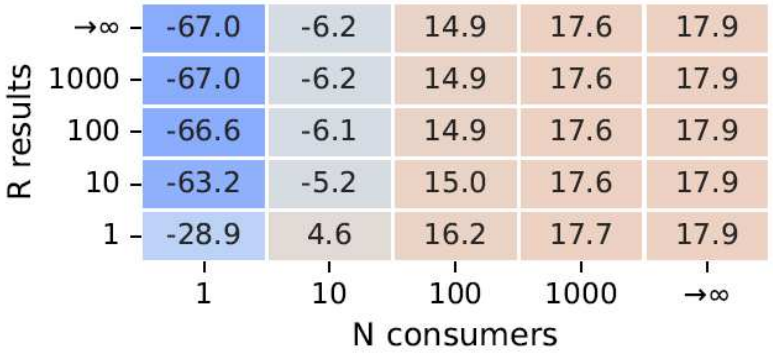
Results :
Algorithmic complexities

Complexity \ Method	Proxy tasks	Time-bounded work passes	One-time work passes	Per-service work passes
On-chain	$O(R * C)$	$O(C * N)$	$O(C * N)$	$O(C)$
Off-chain	$O(R)$	$O(C * N)$	$O(R * C * N)$	$O(1)$

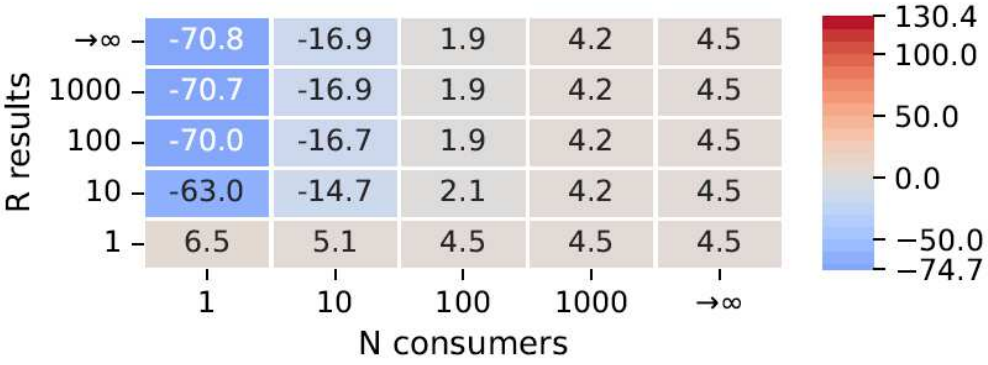
Table 6.2: Administrative cost complexities
Complexities are relative to the number C of target chains, N consumer contracts per chain (on average), and R results to transfer.

Results: Scaling costs

(a) *Time-bounded work passes*



(b) *One-time work passes*



(c) *Per-service work passes*

$$\lim_{N\to\infty,R\to\infty}\left(\frac{G(N,R)_X - G(N,R)_{PT}}{G(N,R)_{PT}}\right) = \frac{G_{receiveResultX} - G_{receiveResultPT}}{G_{receiveResultPT}}$$

Table 6.3: Relative costs (%) of *work pass* methods compared to *proxy tasks*, depending on the number N of consumers and R results to transfer

Results:
Impact of N-Batching

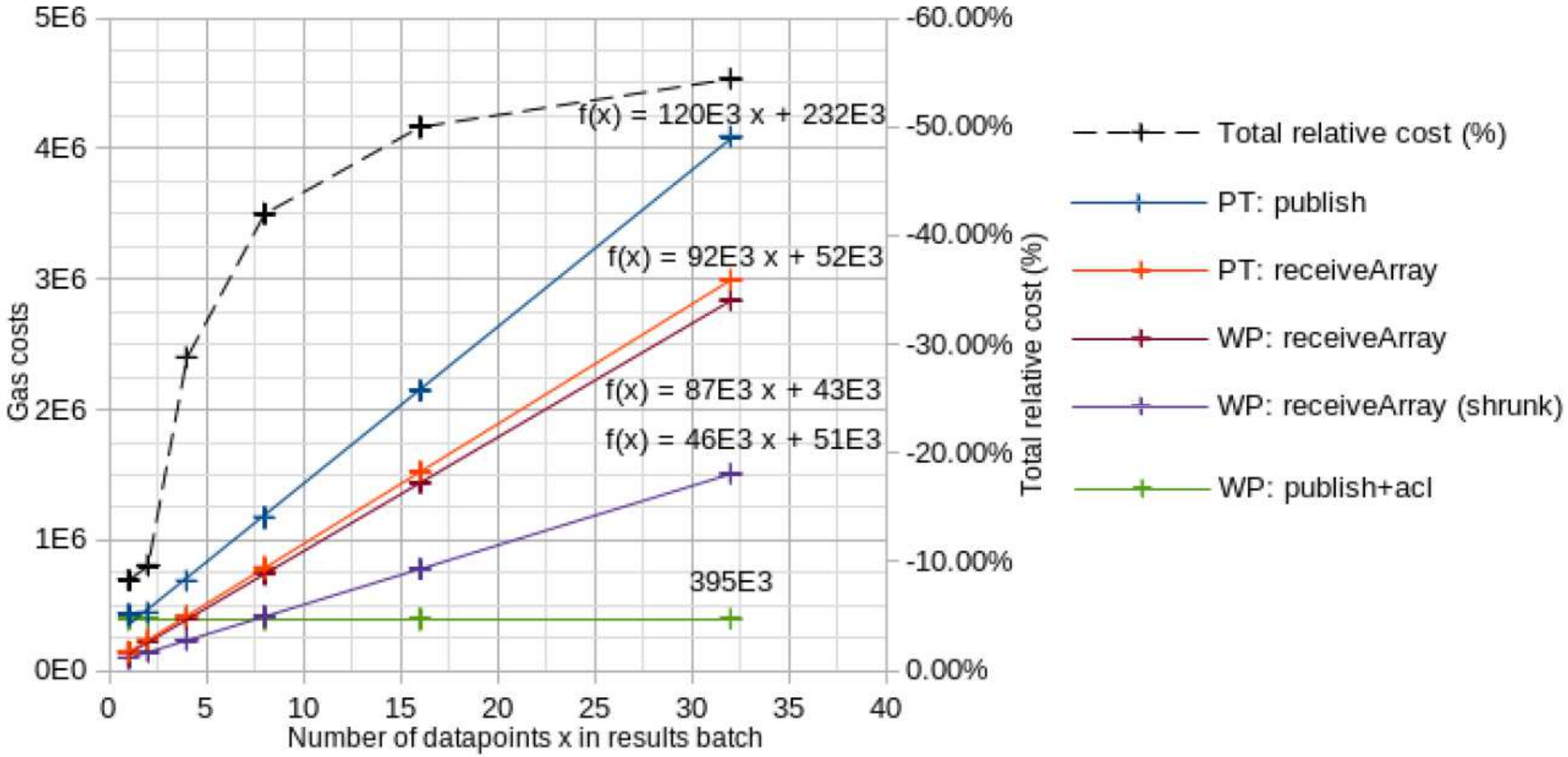


Figure 6.4: Gas costs to transfer n data points at once, using *Proxy tasks* (Proxy) or *Work Passes* (WP)

Conclusion

- **Reuse of trusted security mechanisms**
- **Less storage and processing on costly blockchains**
- **More genericity:**
  - Data acceptance mechanisms
  - Result size and number

Future work

- **Scalability stress tests**
- **dApp modularity/composability**
- **New data acceptance mechanisms**
  - Consensus for streamed results

Trustworthy oracles

Practical work

Matthieu BETTINGER

Goal

- Either:
  - Apply trust- and/or event-based logic to your own project
  - Experiment with concepts on a standalone project
    - Trustworthy datafeed mechanisms
      - Request-response infrastructure
      - Staking
      - Consensus:
        - Aggregation (avg, med,…)
        - Voting
        - Random selection
        - …

Resources

- Web3J Event listeners
  - https://kauri.io/communities/Java%20Ethereum/listening-for-ethereum-smart-contract-events-in-j/
- Web3JS Event Listeners
  - https://www.pauric.blog/How-to-Query-and-Monitor-Ethereum-Contract-Events-with-Web3/
- Solidity events
  - https://www.tutorialspoint.com/solidity/solidity_events.htm