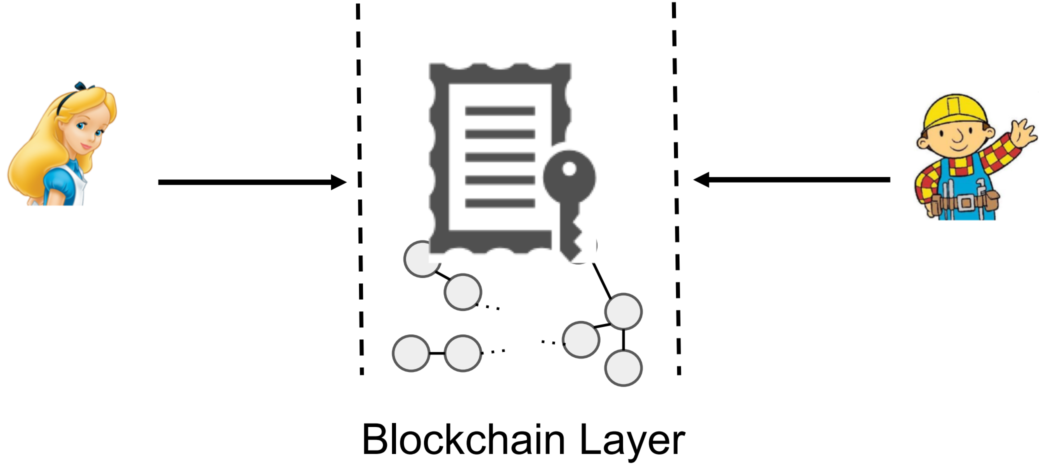
YODA: Enabling Computationally Intensive Contracts in Blockchains with Byzantine and Selfish nodes

Sourav Das, Vinay J. Ribeiro, and Abhijeet Anand
Indian Institute of Technology Delhi

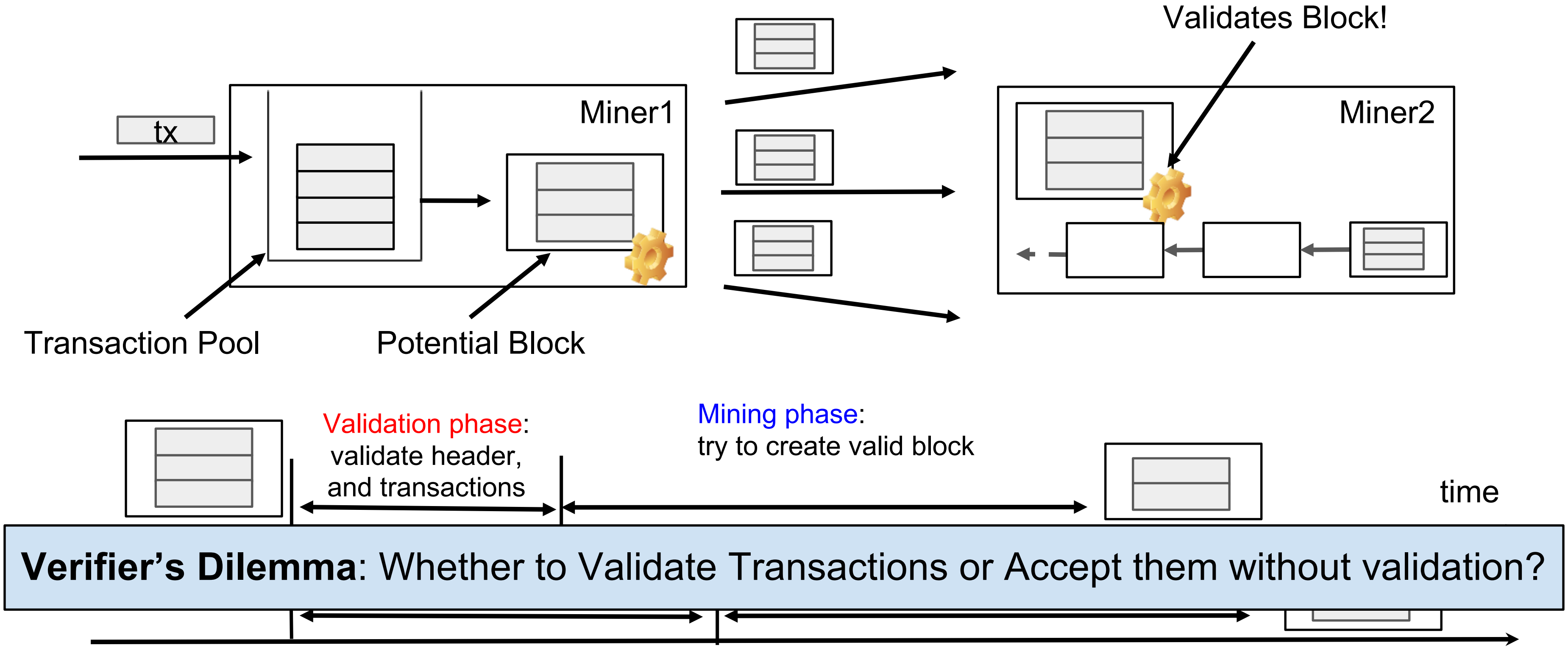
Smart Contracts



- Fair exchange of goods
- Fair public auctions

Correctness of Contract Execution

- Verification of Contract by Re-execution



Verifier's Dilemma: Whether to Validate Transactions or Accept them without validation?

- How to deal with Verifier's Dilemma?
  - Limit maximum amount computation a transaction can invoke.
  - BlockGasLimit in Ethereum.
  - ~500k instructions per second

- Consequence of Re-executing contracts for Verification
  - Can't Execute large (Computationally Intensive) functions/ smart contracts

- Need of executing large functions in Blockchain
  - Playing Online Games.
  - Privacy preserving computation, e.g: FHE, Zero-Knowledge protocols
  - Machine Learning on Blockchain

YODA: Enabling Computationally intensive contracts in Blockchains with **Byzantine** and **Selfish** nodes.

Outline

- Introduction and Motivation
- System Model
- YODA Overview
- MiRACLE
- RICE
- Evaluation

System Model

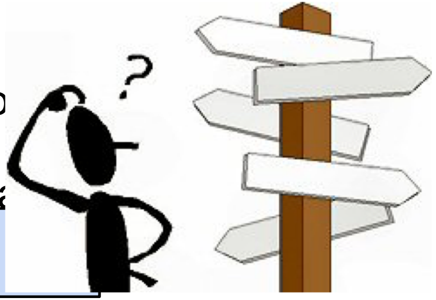
- Blockchain c ... no
- <50% of the nodes in the system a... yzantine.
- Underlying blockchain guarantees **Correctness** and **Availability**

Transactions are Executed Correctly

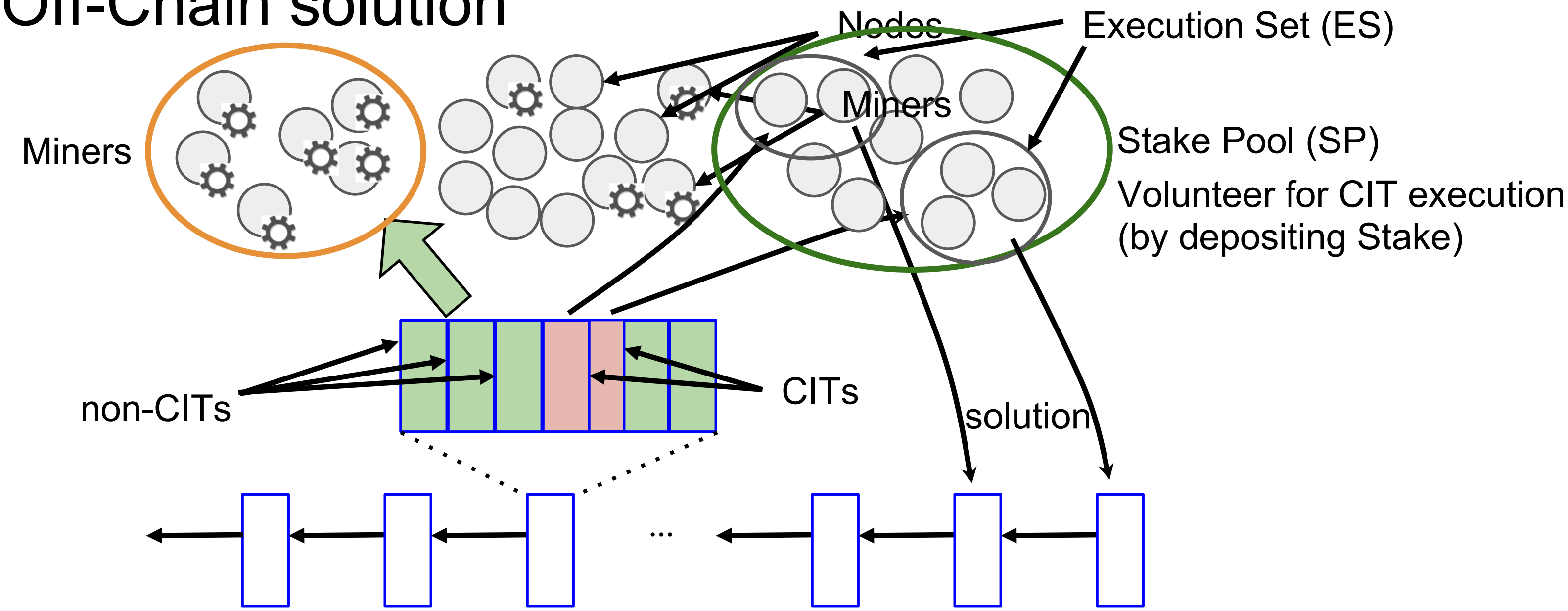
Transactions get Included within bounded delay

- Definitions
  - **Computationally Intensive Transactions** (CITs) invoke functions that are larger than Block Limit Threshold.
  - **non-CIT**: Transactions that are not CITs.
  - YODA executes CITs **off-chain**, i.e only by a subset of no
  - non-CITs are executed **on-chain** i.e by all miners. Identica

What does **off-chain** Execution mean?

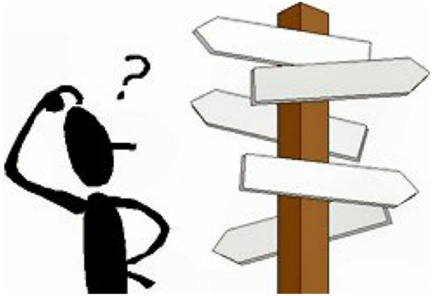


YODA’s Off-Chain solution



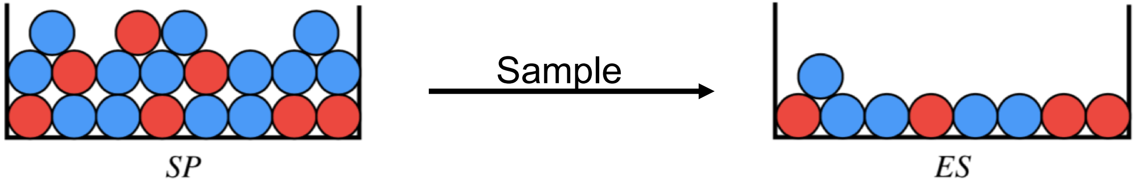
- Requirements from YODA
  - Small ES
  - 50% adversaria
  - β error Probabili

How does YODA meet these requirements?

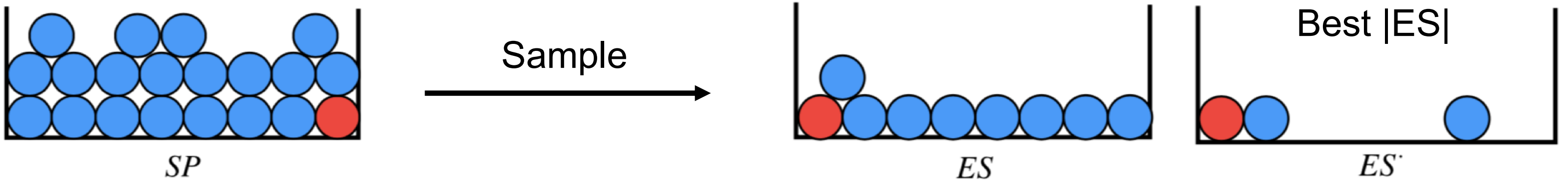


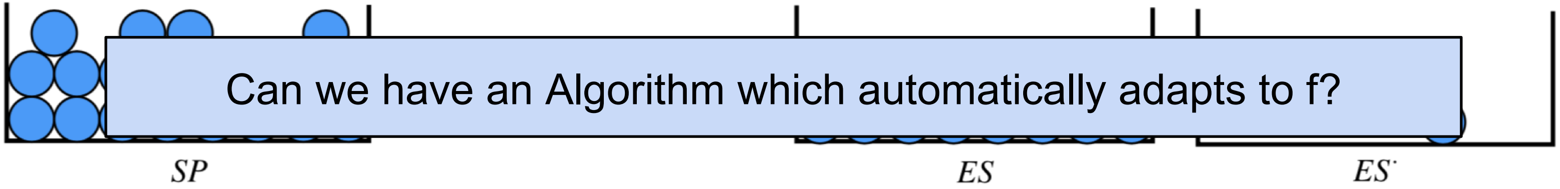
Byzantine and ~~Selfish~~ Honest nodes in SP

- **Honest** nodes: always submit correct execution result of CITs
- Consider a **Naive Solution** using sampling



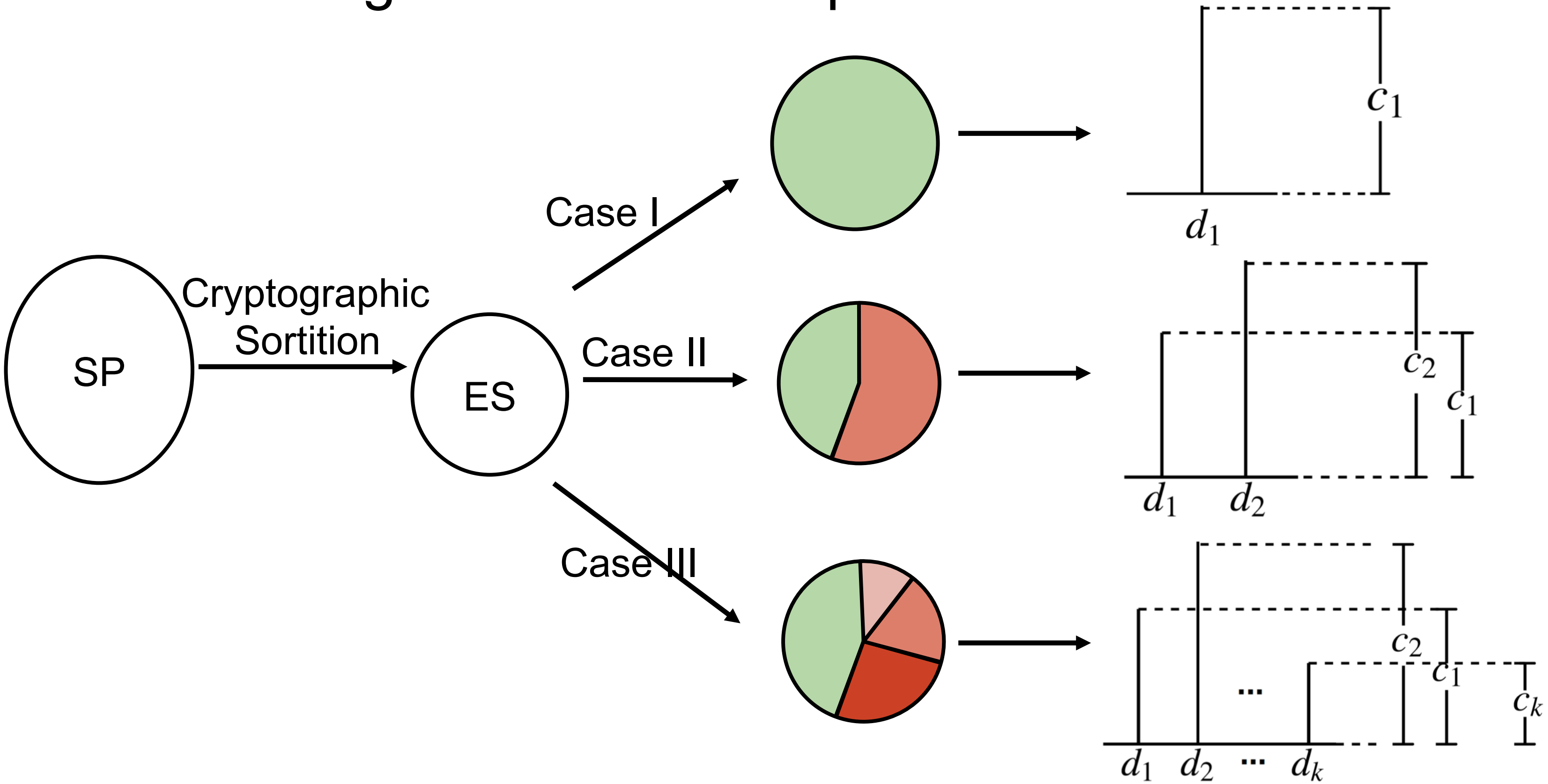
For $|SP| = 1600$ and $f_{max} = 0.35$, $|ES| \approx 900$. $\beta = 10^{-20}$





Can we have an Algorithm which automatically adapts to f?

MiRACLE Algorithm via Examples



Case II: MiRACLE has to identify correct solution from d_1 and d_2 .

MiRACLE: Case II

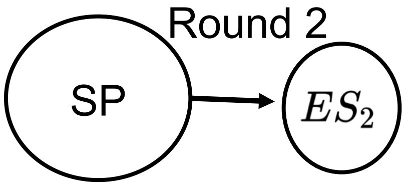
(d_1, c_1) (d_2, c_2)

Two Hypothesis: $H_1 : d_1$ is correct solution $H_2 : d_2$ is correct solution

Compute Likelihood: $\boldsymbol{L_{1,1}} = (c_1^2 - c_2^2)$ $\boldsymbol{L_{2,1}} = (c_2^2 - c_1^2)$

If $L_i > \mathbb{T}$ for $\mathbb{T} = \ln\left(\frac{\beta}{1-\beta}\right) \frac{2q(1-q)Mf_{max}(1-f_{max})}{(1-f_{max})-f_{max}}$ YES → d_i

NO ↓



(d_1, δ_1) (d_2, δ_2)

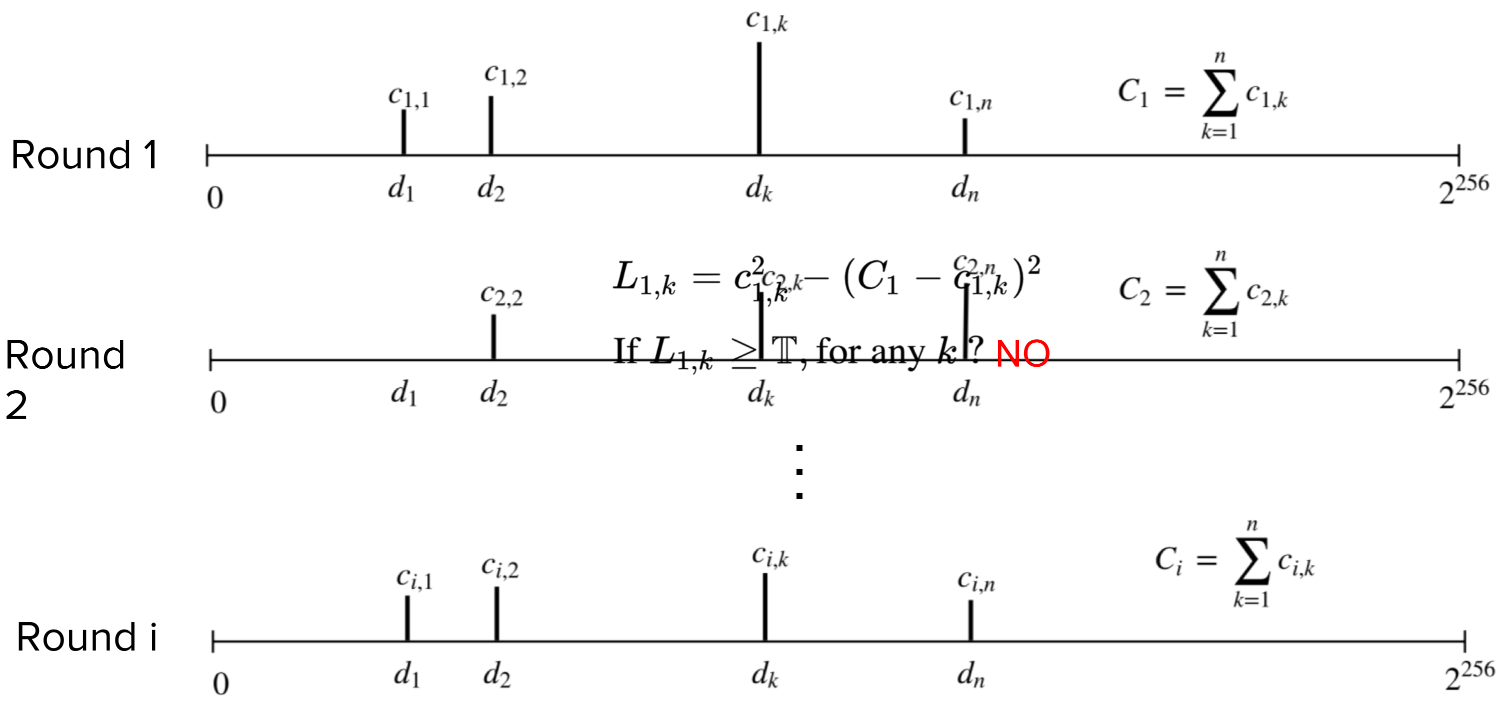
Compute Likelihood: $L_{1,2} = L_{1,1} + (\delta_1^2 - \delta_2^2)$ $L_{2,2} = L_{2,1} + (\delta_2^2 - \delta_1^2)$

$L_{i,2} > \mathbb{T}$? YES → d_i

NO ↓

MiRACLE terminates in expected $\frac{(1-\beta)\ln(\frac{\beta}{1-\beta})+\beta\ln(\frac{1-\beta}{\beta})}{A(f_{max},|SP|,|ES|)}$ rounds

MiRACLE: Case III

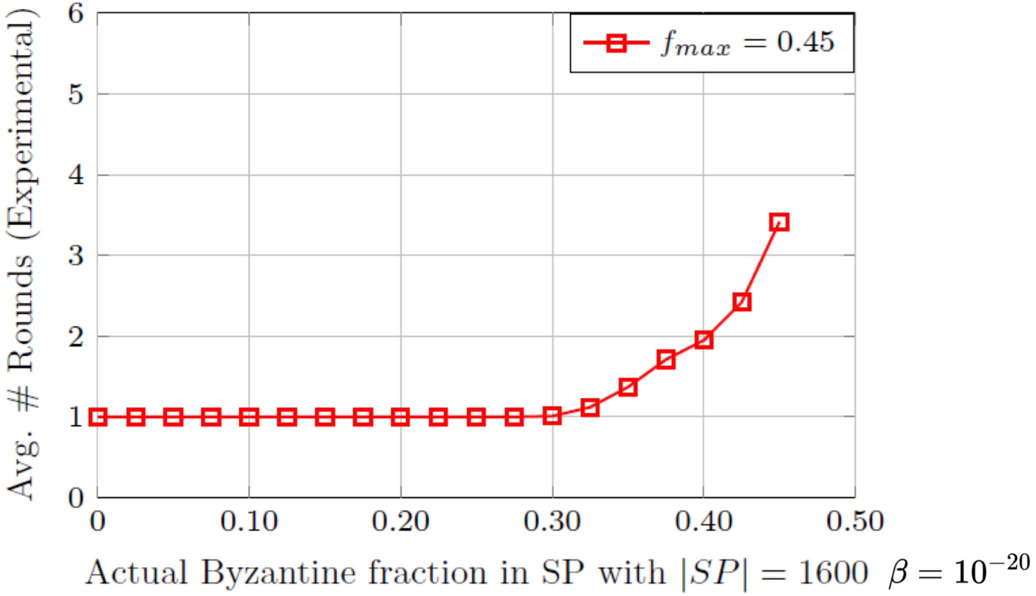


$$L_{i,k} = \sum_{j=1}^{i} \left(c_{j,k}^2 - (C_j - c_{j,k})^2\right)$$

If $L_{i,k} \geq \mathbb{T}$, for any k ? YES Declare d_k as correct digest

MiRACLE: Theoretical Results

- Terminates with correct solution with Probability $1 - \beta$
- Terminates with only single solution
- Best strategy for adversary is to submit single solution

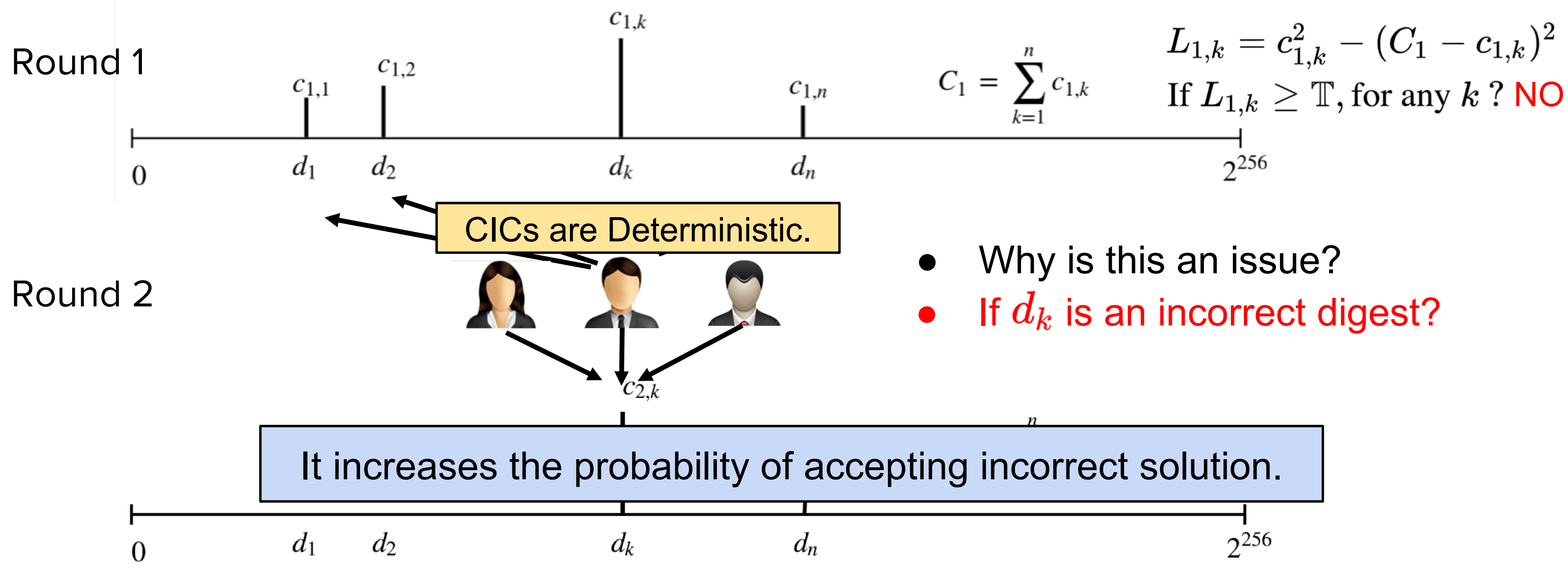


Outline

- Introduction and Motivation
- System Model and Related Work
- YODA Overview
- MiRACLE
- Selfish nodes in SP (RICE)
- Evaluation

Byzantine and Selfish ~~Honest~~ nodes in SP

- **Selfish** Node: seeks to maximize (reward for computation - cost of computation)
- Skip computation if they can guess the result beforehand
    - Using information available in the Blockchain
    - Colluding with other ES nodes.
- MiRACLE is not sufficient with Selfish nodes

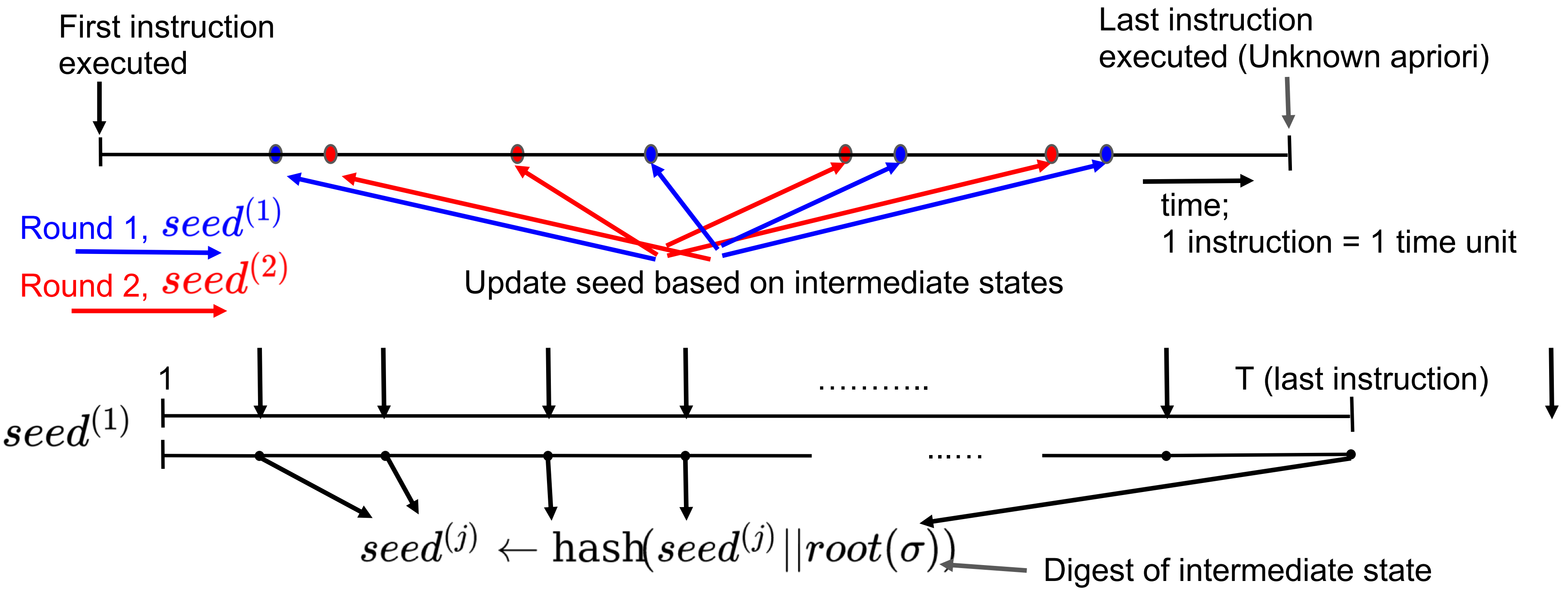


- Why is this an issue?
- If d_k is an incorrect digest?

RICE: Randomness Inserted Contract Execution

- ES nodes submit a solution $d \,||\, seed^{(j)}$ ← round number

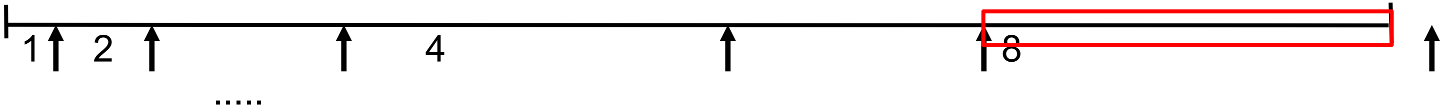
Solution of CIT — Round-dependent pseudorandom value



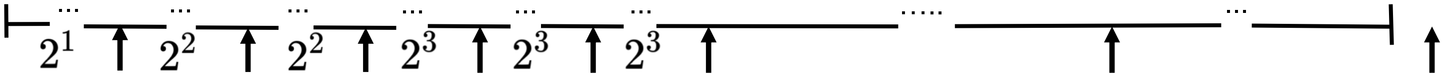
Choosing Indices in RICE

- Keep the number of updates small.

- Fixed size interval (k), **O(T)** updates



- Interval doubling after every step, $\mathbf{O(\log_2 T)}$ updates
- Gap between last update and T could be $\mathbf{\frac{T}{2}}$

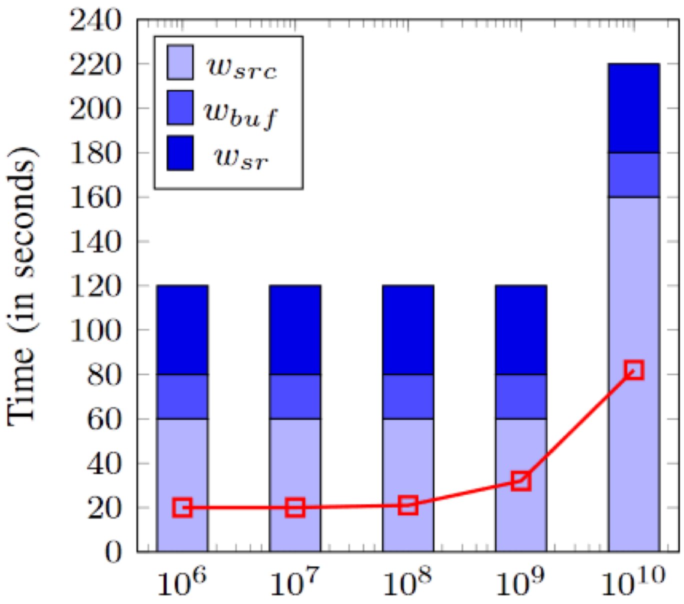


- YODA considers i interval of size 2^i
- $\mathbf{O(\log_2^2(T))}$ updates, last gap $\mathbf{\frac{T}{\log_2 T}}$

Outline

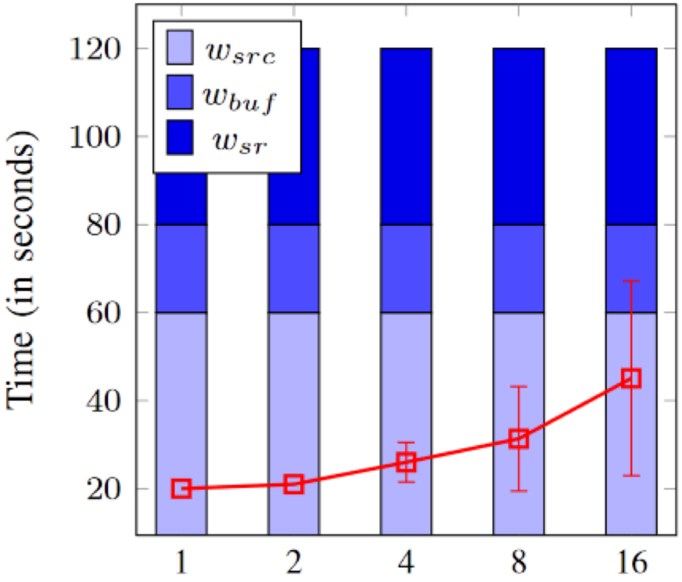
- Introduction and Motivation
- System Model and Related Work
- YODA Overview
- MiRACLE
- Selfish nodes in SP (RICE)
- Evaluation

CICs in YODA



Gas Usage (multiples of 5.3), $|ES| = 40$

Figure 3: Measured CIC execution time with varying gas usage.



Parallel CICs, $|ES| = 40$

Figure 4: Average digest commit time with increasing number of parallel ITs.

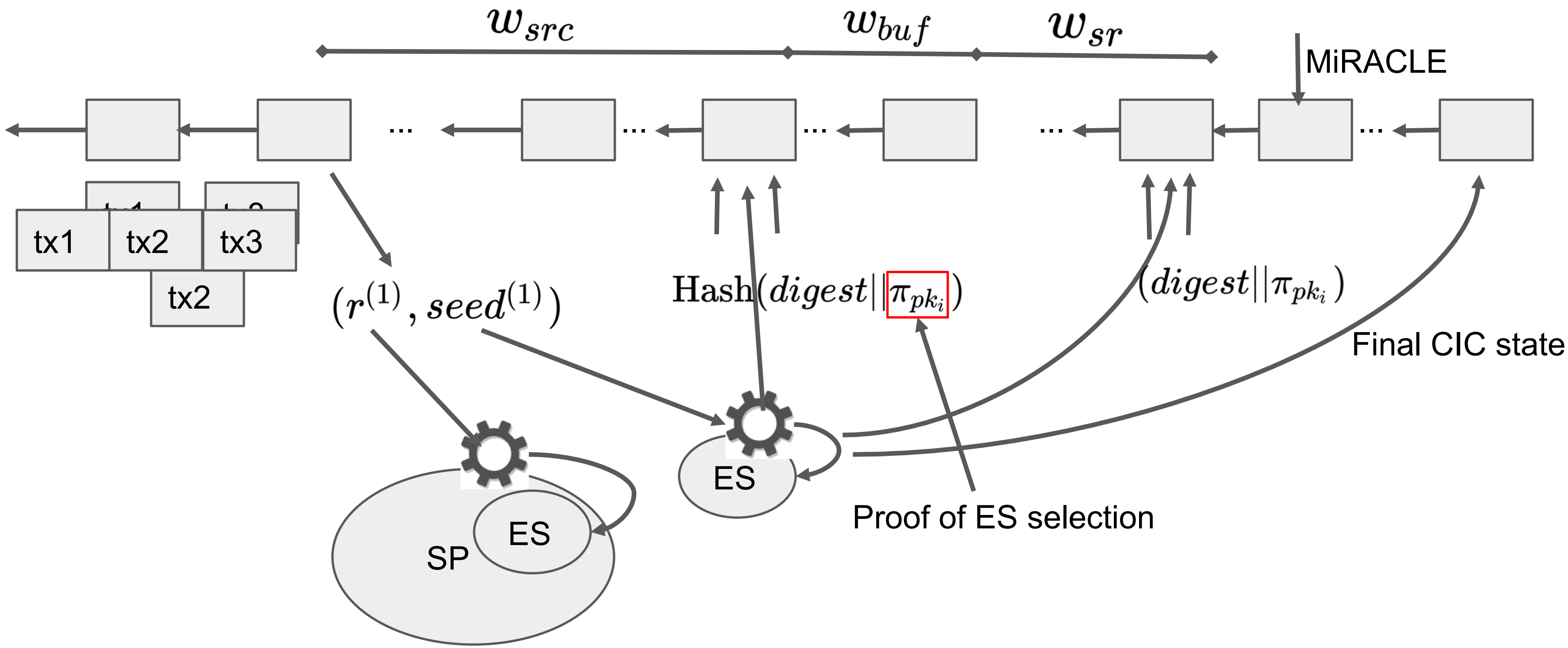
Additionally in paper.

- Security properties on MiRACLE and RICE
- Collusion among ES nodes from same rounds
- Fair reward mechanism
- Incentive compatibility of YODA (ϵ-Nash equilibrium)
- Implementation and Evaluation details

Thank You

souravdas1547@gmail.com

Putting it all together on a blockchain



MiRACLE (Pseudocode)

Algorithm 1 MIRACLE

$i \leftarrow 1$
while $L_{k,i} \leq \mathbb{T} \; \forall k$ **do**
    $i \leftarrow i + 1$
    Pick next ES to execute $\Psi(x)$
end while
declare $d_{k'}$ to be correct where $L_{k',i} > \mathbb{T}$