Puzzling Sybil into Bankruptcy

Diksha Gupta

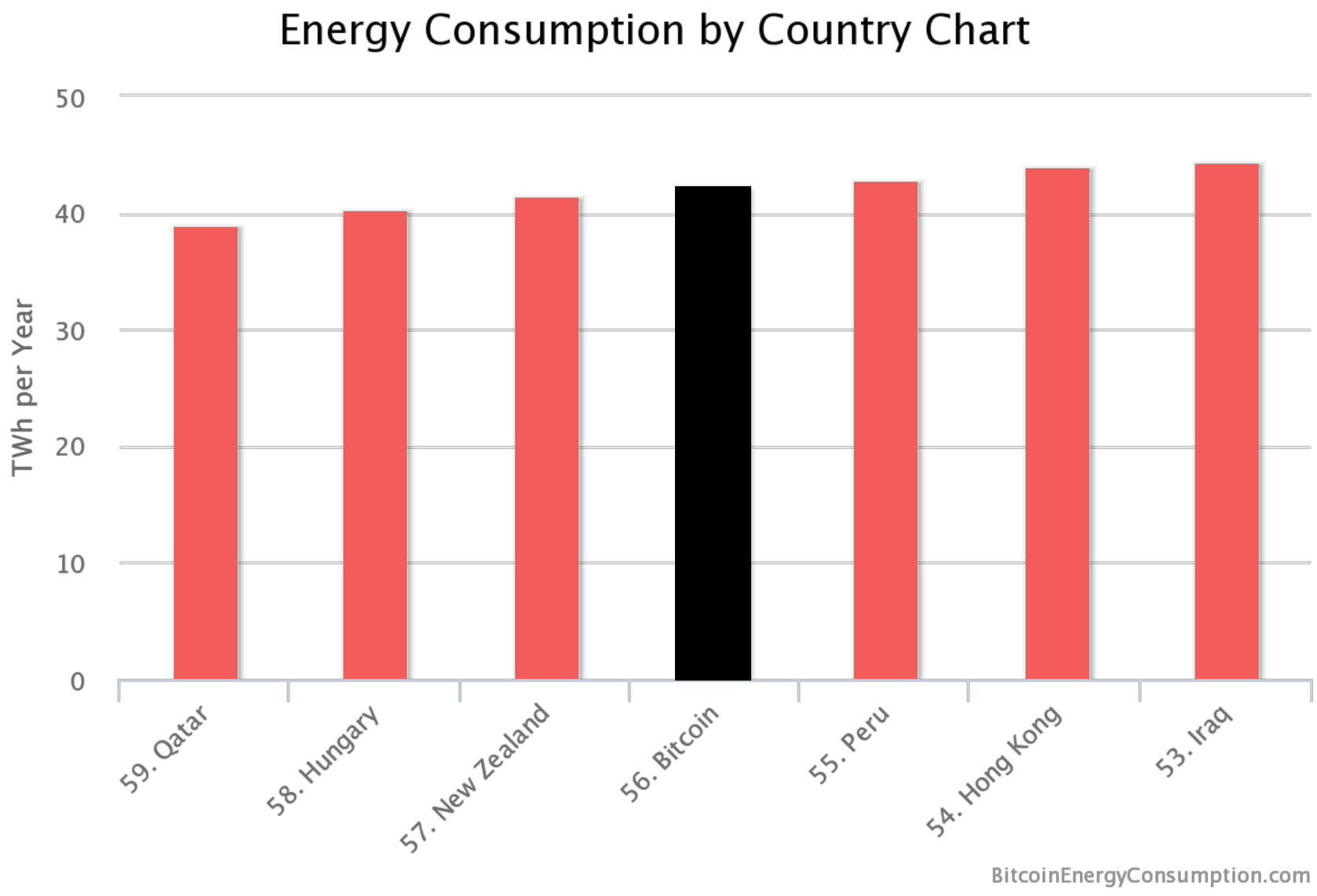
Joint work with Jared Saia and Maxwell Young

THE UNIVERSITY OF NEW MEXICO

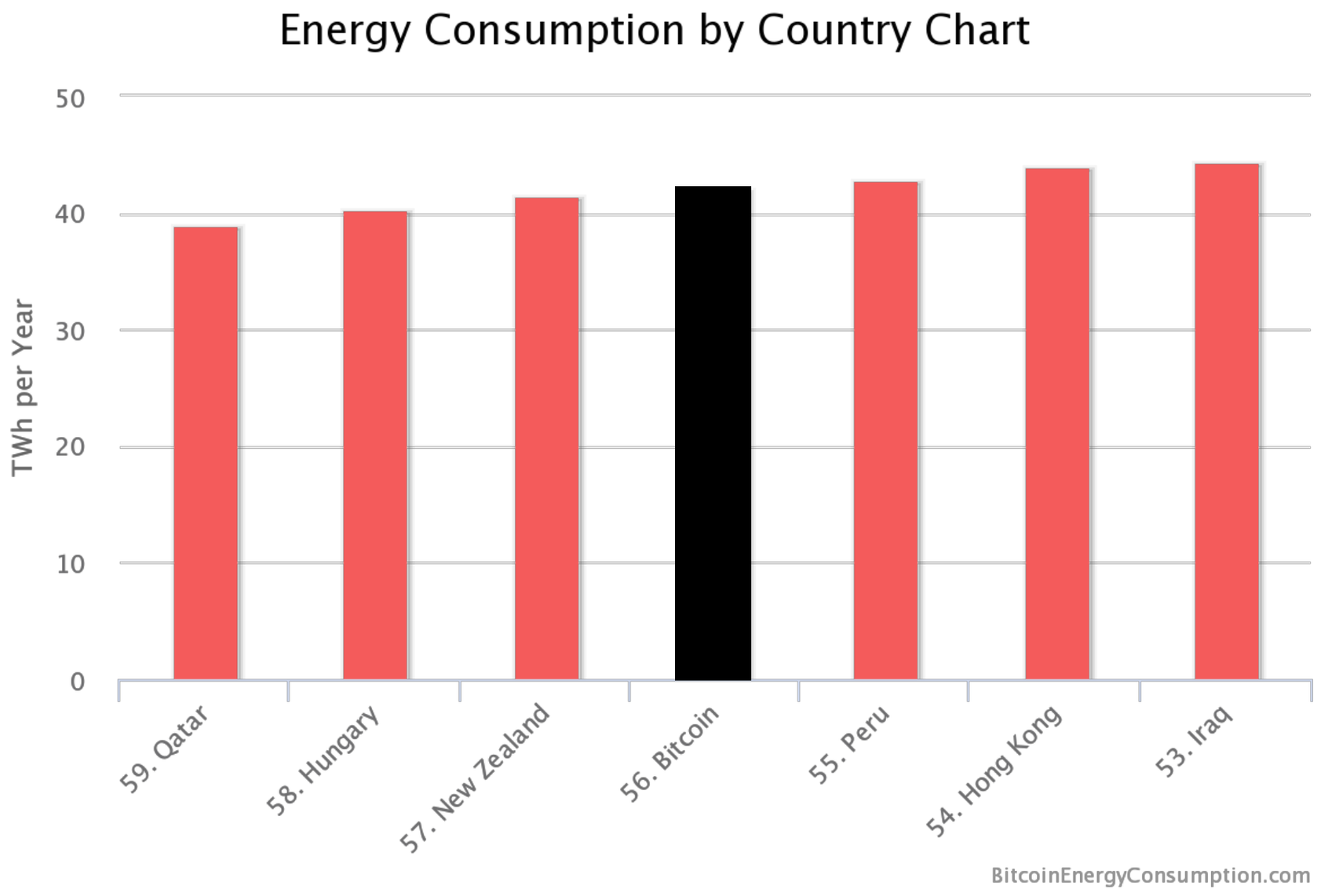
STATE
MISSISSIPPI STATE UNIVERSITY™

Some Motivation…

Bitcoin Energy Consumption w.r.t several countries



PoW is expensive!



Problem Statement

Design an algorithm that limits the number of sybil IDs in an open system such that it provides the following guarantees:

- **Correctness** - Majority of IDs in the system are honest (good).

- **Efficiency** - Keep computational Costs as low as possible.

Our Model

Computation

- **Random Oracle Assumption:** We have a function, h, and h(x) is uniformly random on (0, 1] the first time bit string x is input to h.
- **Computation Cost**: Computational cost is number of times h is called

Communication

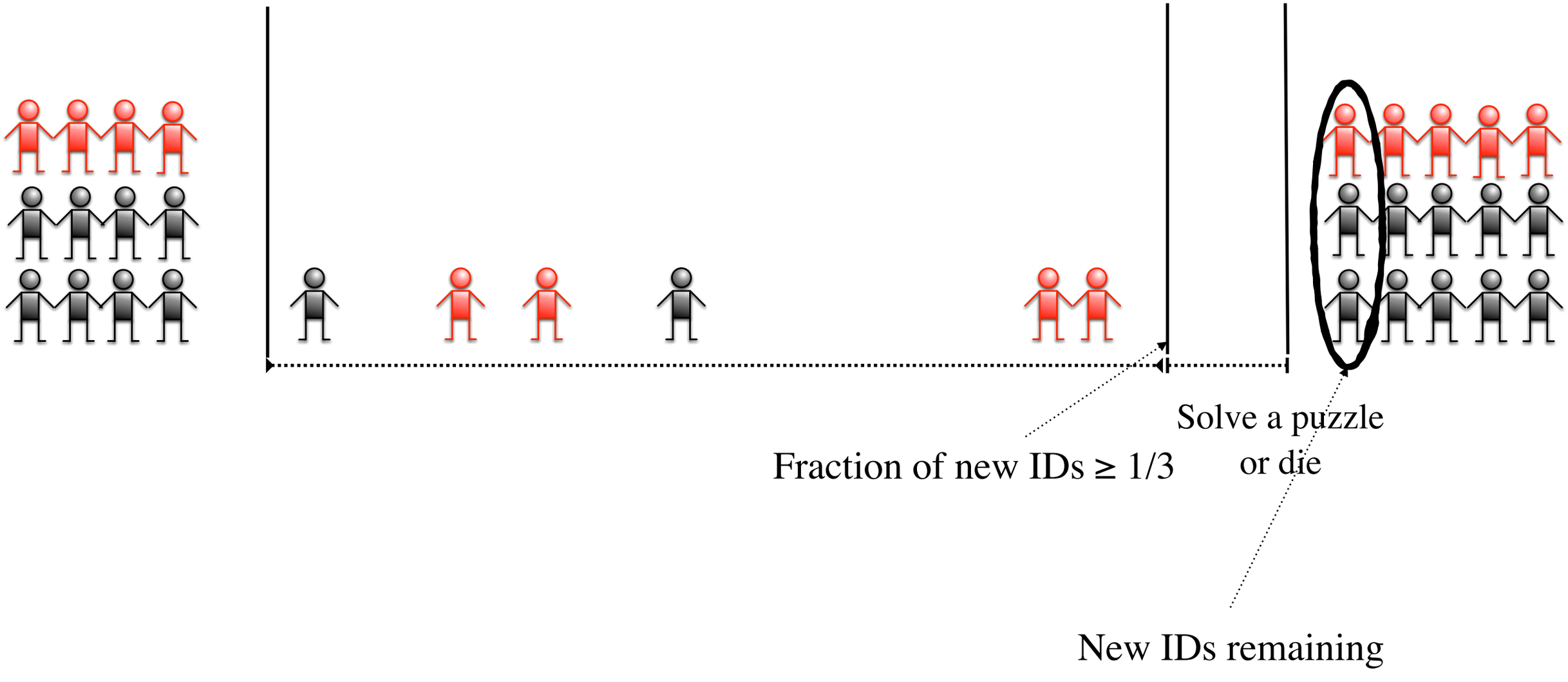
- *Diffuse* protocol:
  - Sends a message to all IDs
  - Never learn where a message came from
  - Communication time is negligible compared to computation time

Adversary

- Has a 1/3 fraction of the computational power of the network
- Knows our algorithms, not our random bits
- Cryptographically bounded

Our Approach

Naive Approach



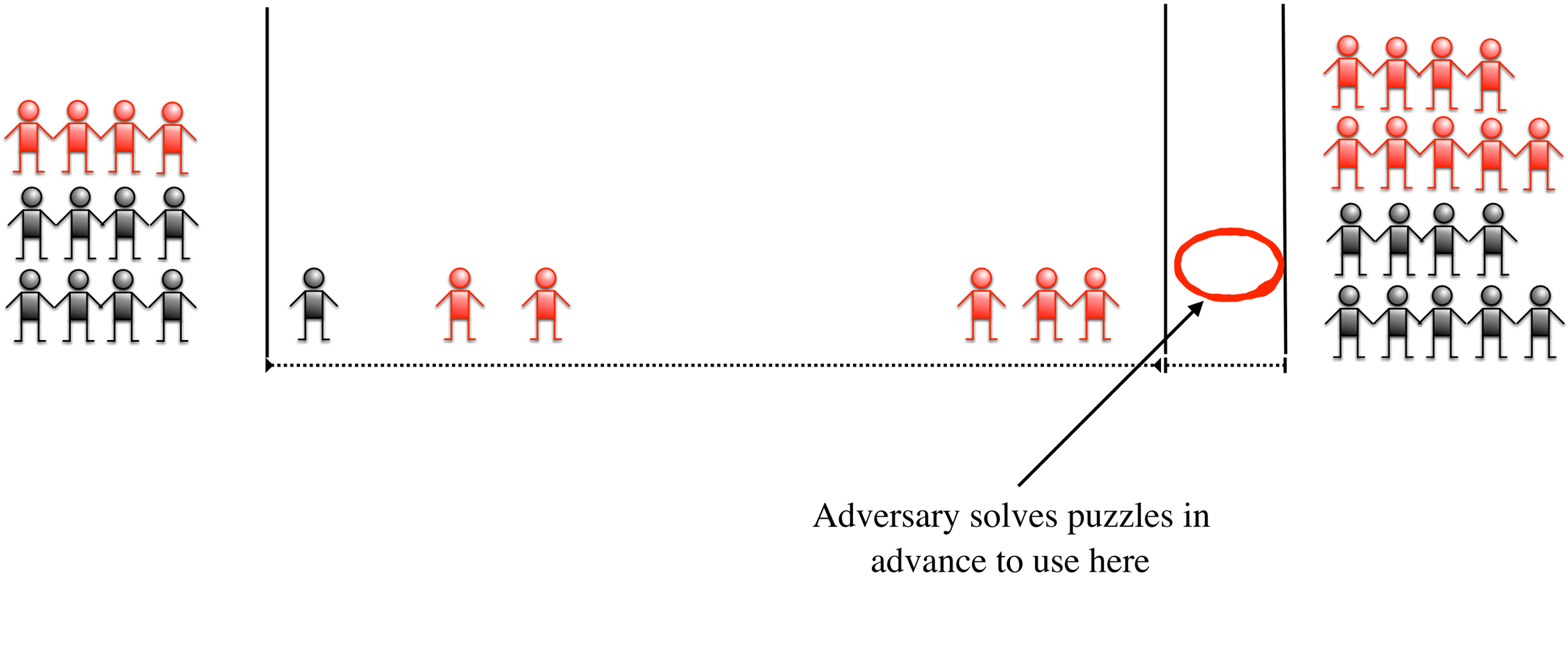
Problems

1. Pre-computation attack

2. Costs not commensurate

3. Handling joins and leaves

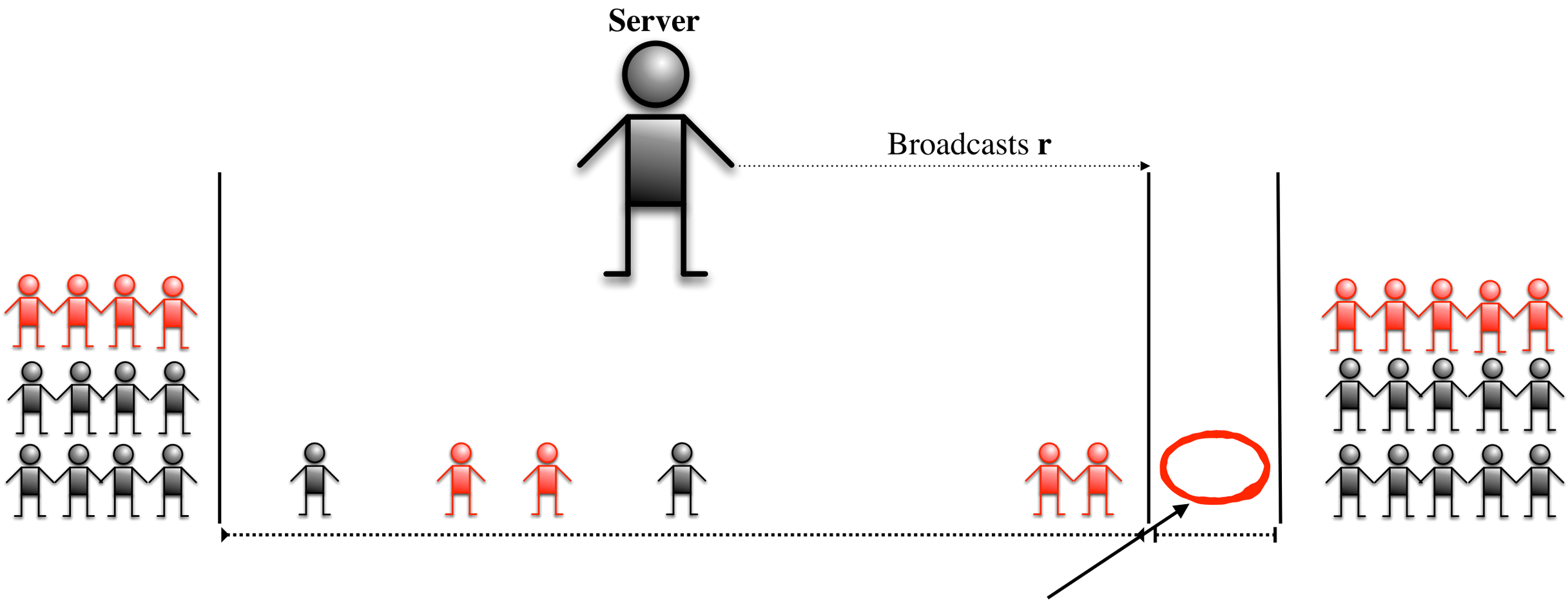
Problem 1: Pre-computation Attack



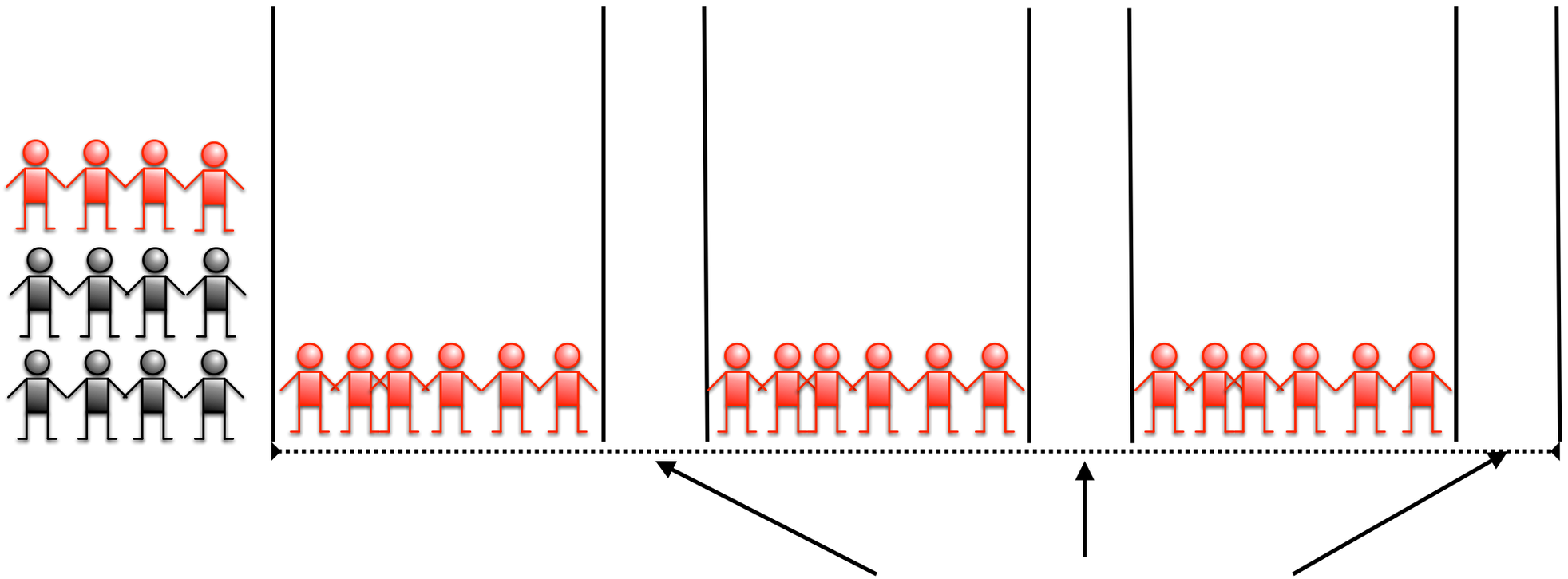
The Server

- Initially we assume a trusted server
- Later we remove this assumption

Solution 1: Broadcast random seed **r**



Problem 2: Costs not commensurate

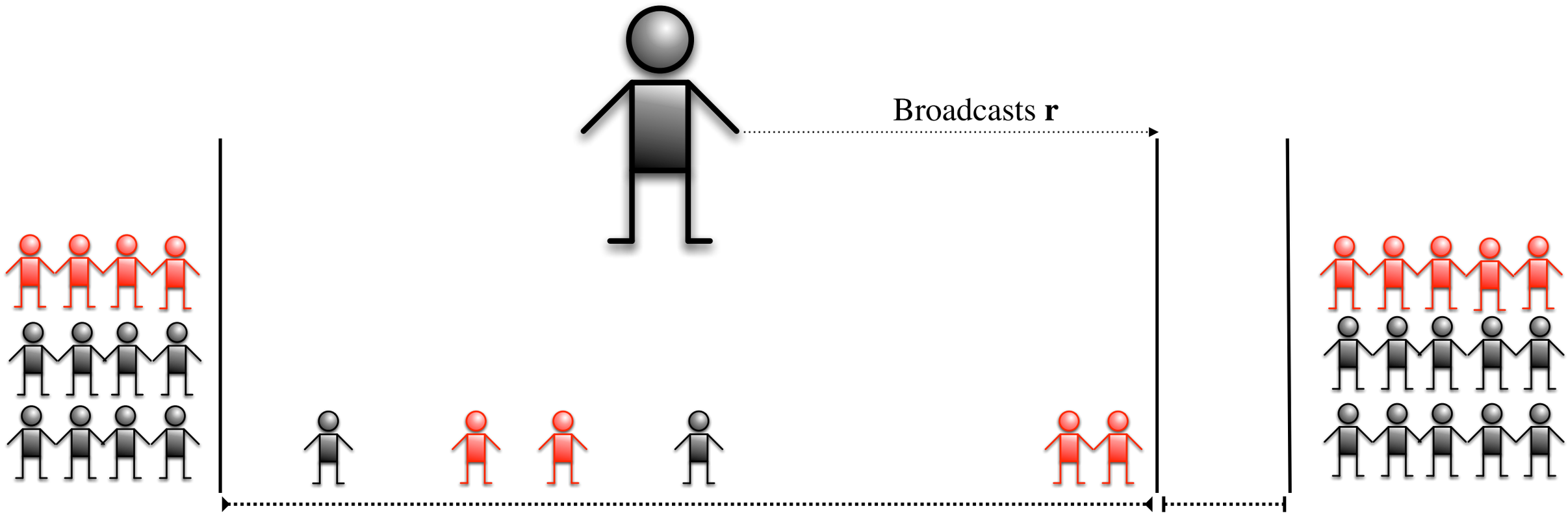


Good IDs solve puzzles here

Bad IDs do not - they drop out

Good IDs pay high cost; Adversary does not.

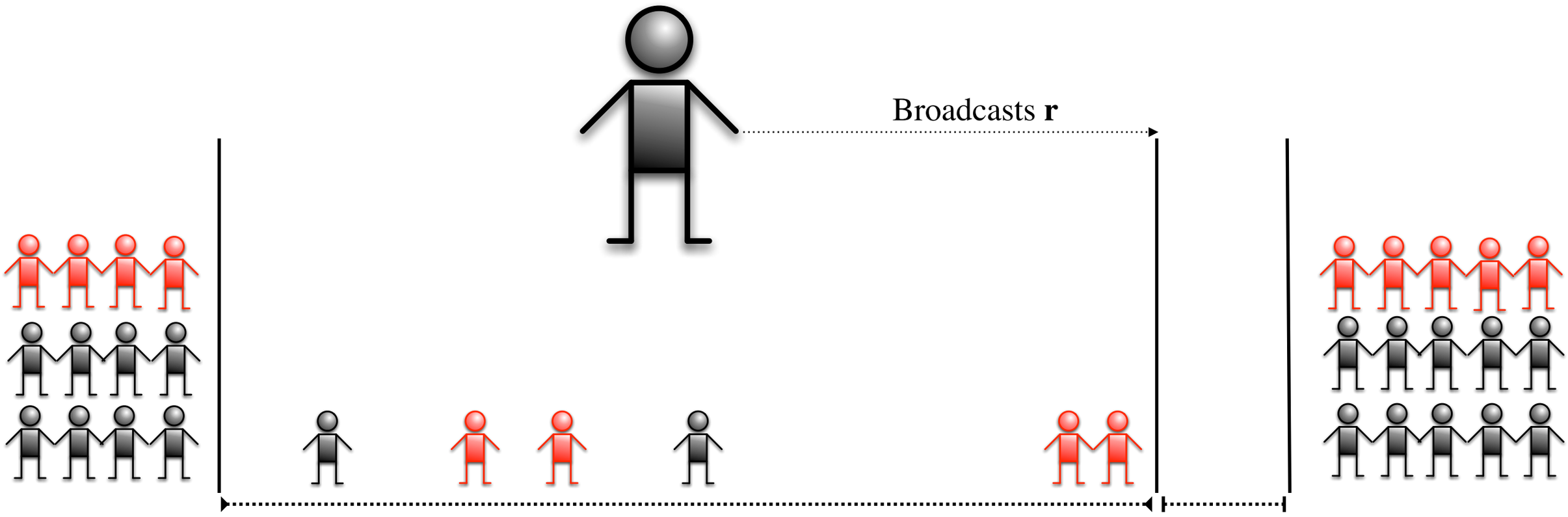
Solution 2: Entrance Puzzles



New IDs solve
Entrance Puzzle

All IDs solve
Purge Puzzle

Solution 2: Entrance Puzzles

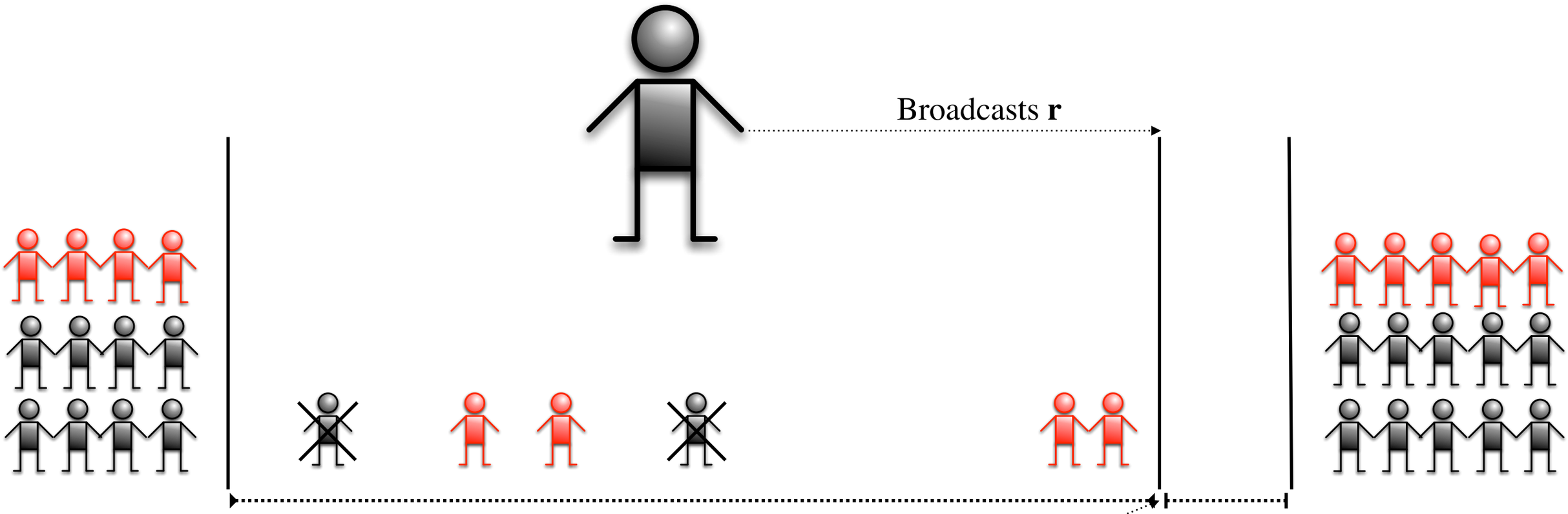


Purge Puzzle: find bit string x such that $h(\mathbf{r} \| x) \leq \tau$ for some small τ

Entrance Puzzle: find bit string x such that $h(t \| x) \leq \tau$ for some small τ, and current time t

Note: Adversary can pre-compute the purge puzzles. Their purpose is just to ensure Adversary pays (at some point) if good IDs pay

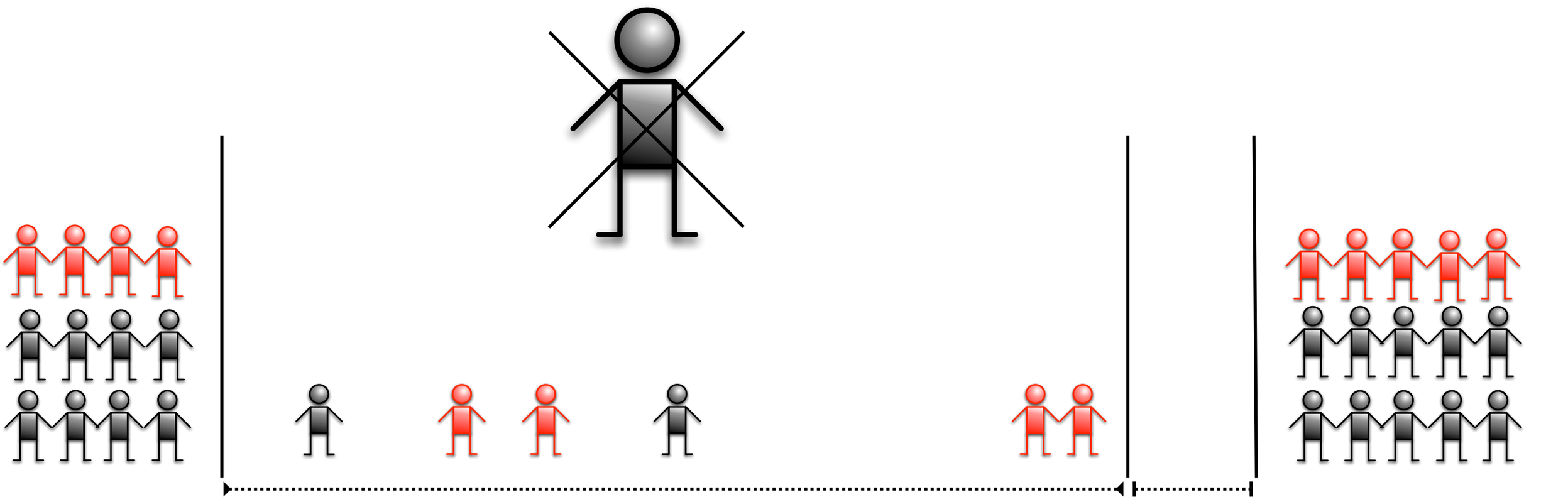
Problem 3: Handling Joins and Leaves



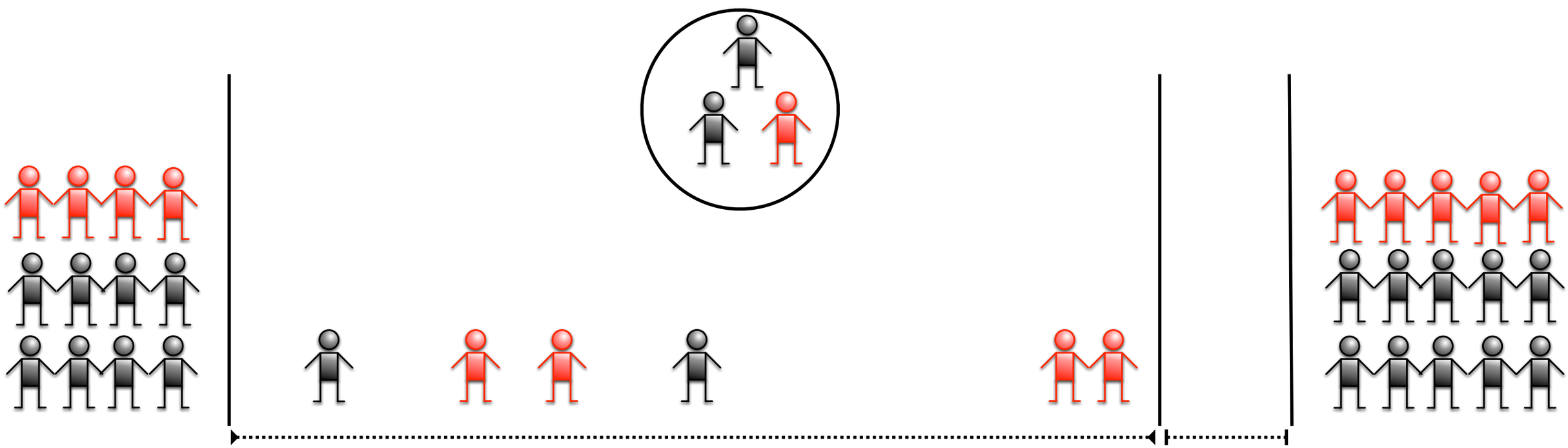
Solution: Check symmetric difference between set at beginning of epoch and current set. Purge IDs if:

$|(S_t \cup S_0) - (S_t \cap S_0)| \geq |S_0|/4$

Removing the server



Replace with a committee



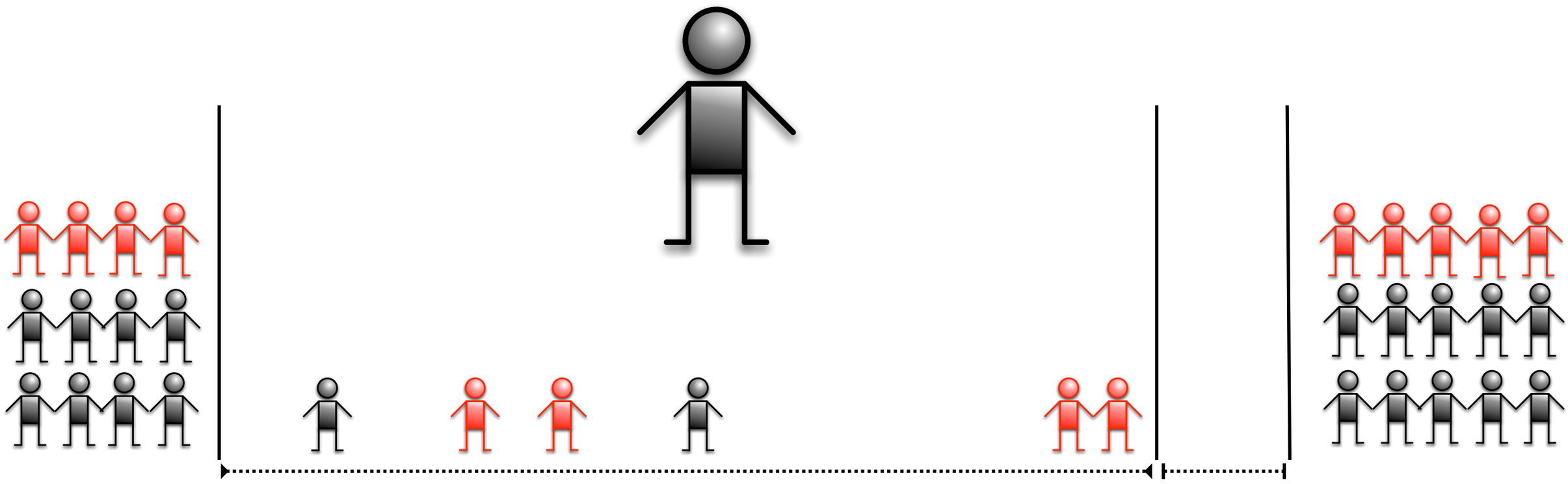
Theorem

Let **g** be the total number of good IDs that join over the lifetime of the system and **T** be the computational cost to the adversary. Then, w.h.p. our algorithm has the following properties:

- **Correctness** - Majority of IDs in the system are honest.

- **Efficiency** - In absence of an attack, the computational cost to good IDs is O(g). When under attack, the computational cost is O(g+T).

Reducing Costs Further



Tune difficulty of
Entrance Puzzle

Entrance Puzzle: For an ID entering at time t, difficulty of entrance puzzle = join rate of IDs until time t in current epoch/ average rate of joins in the previous epoch

Theorem

Let **J** be the average rate of join of good IDs over the lifetime of the system and **T** be the average adversarial spending rate. Then, w.h.p. our algorithm has the following properties:

- **Correctness** - Majority of IDs in the system are honest.

- **Efficiency** - Average computational spending rate of good IDs is $O(J+\sqrt{(JT)})$.

Thank You

Questions?