Holography Accumulation

Carla Ràfols, joint work with Nikitas Paslis, Alexandros Zacharakis.

ZKPROOF7

OVERVIEW

01

INTRO TO SNARKS

& RESEARCH PROBLEM

02

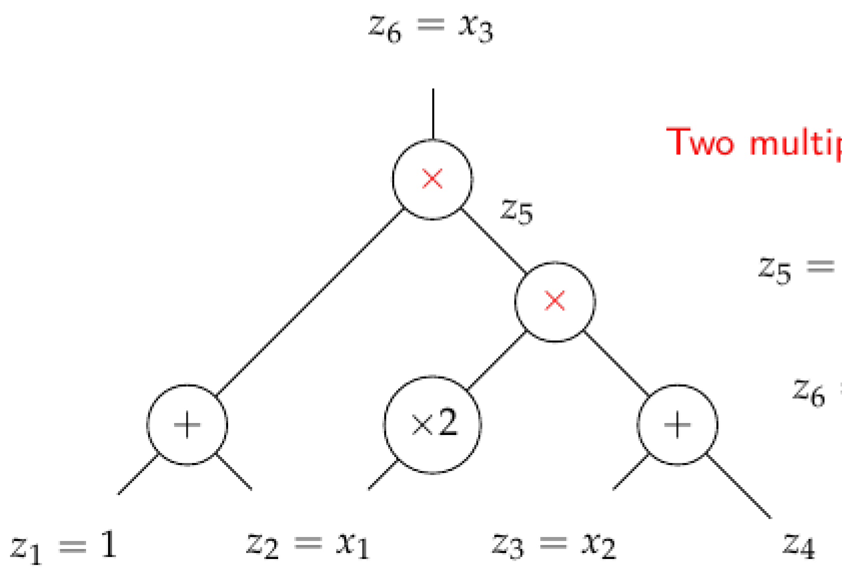
RECURSIVE PROOFS

03

PRIVACY PRESERVING DELEGATION OF SNARK COMPUTATION

Circuit Satisfiability - R1CS

Statement: $C(1, x_1, x_2, w) = x_3$ for some w, $\vec{x}$ public inputs.



Two multiplication gates g_1, g_2

$z_5 = (2z_2)(z_3 + z_4)$

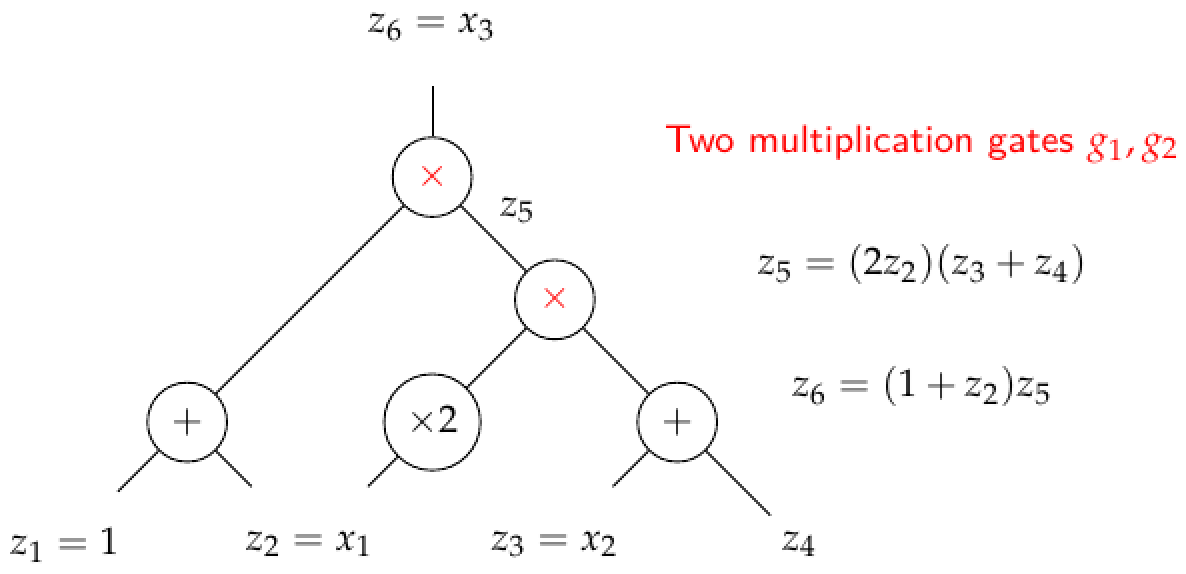
$z_6 = (1 + z_2)z_5$

$$\boxed{\mathbf{A}\vec{z} \circ \mathbf{B}\vec{z} = \mathbf{C}\vec{z}}$$

1. **Public Input Relations:**
   $\{z_1 = 1, z_2 = x_1, z_3 = x_2, z_6 = x_3\}$
2. **Hadamard Product Relation**:
   $\vec{a} \circ \vec{b} = \vec{c}$
3. **Linear Relations**:
   $\vec{a} = \mathbf{A}\vec{z}$, $\vec{b} = \mathbf{B}\vec{z}$, $\vec{c} = \mathbf{B}\vec{z}$.

Circuit Satisfiability - R1CS

Statement: $C(1, x_1, x_2, w) = x_3$ for some w, $\vec{x}$ public inputs.



$$\boxed{\mathbf{A}\vec{z} \circ \mathbf{B}\vec{z} = \mathbf{C}\vec{z}}$$

1. **Public Input Relations:**
   $\{z_1 = 1, z_2 = x_1, z_3 = x_2, z_6 = x_3\}$
2. **Hadamard Product Relation**:
   $\vec{a} \circ \vec{b} = \vec{c}$
3. **Linear Relations**:
   $\vec{a} = \mathbf{A}\vec{z}$, $\vec{b} = \mathbf{B}\vec{z}$, $\vec{c} = \mathbf{B}\vec{z}$.

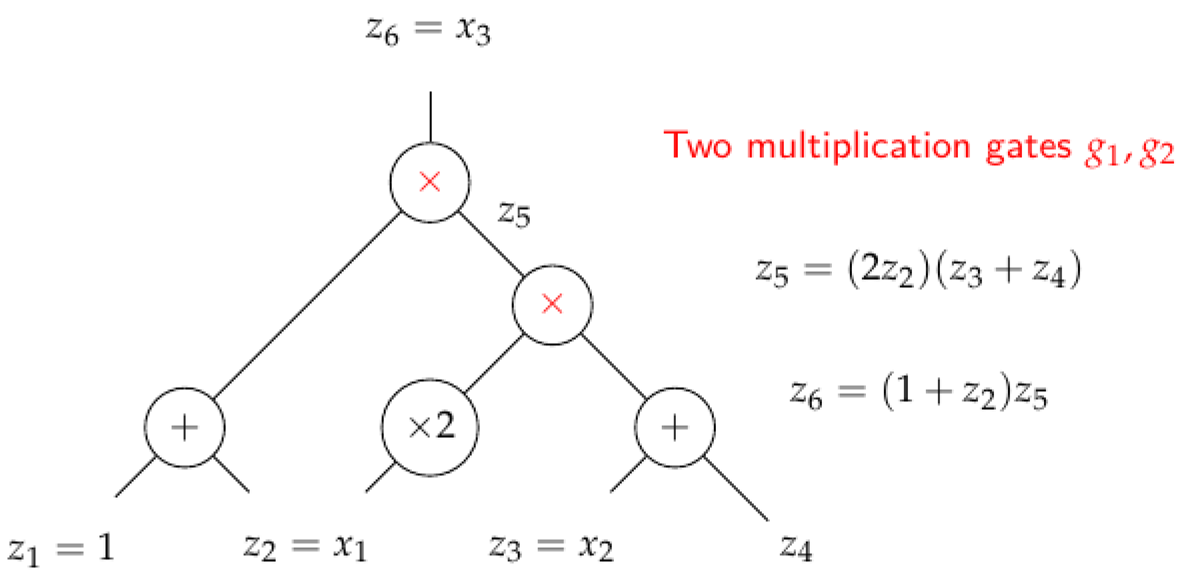
Matrices sparse, of size $|m.gates| \times |witness|$.

$$\mathbf{A}\vec{z} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \end{pmatrix} = \begin{pmatrix} 1 \\ z_2 \\ z_3 \\ z_4 \\ 2z_2 \\ 1 + z_2 \end{pmatrix}, \quad \mathbf{B}\vec{z} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ z_3 + z_4 \\ z_5 \end{pmatrix}$$

$$\mathbf{C}\vec{z} = \mathbf{I}\vec{z} = \vec{z}$$

Circuit Satisfiability - R1CS and CCS

Statement: $C(1, x_1, x_2, w) = x_3$ for some w, $\vec{x}$ public inputs.



$$\mathbf{A}\vec{z} \circ \mathbf{B}\vec{z} = \mathbf{C}\vec{z}$$

1. **Public Input Relations:**
   $\{z_1 = 1, z_2 = x_1, z_3 = x_2, z_6 = x_3\}$
2. **Hadamard Product Relation**:
   $\vec{a} \circ \vec{b} = \vec{c}$
3. **Linear Relations**:
   $\vec{a} = \mathbf{A}\vec{z}$, $\vec{b} = \mathbf{B}\vec{z}$, $\vec{c} = \mathbf{B}\vec{z}$.

---

1. **Public Input Relations:**
   $\vec{z} = (1, \vec{x}, \vec{w})$
2. **Hadamard Product Relation**:
   $\sum_{i=0}^{q-1} c_i \cdot \bigcirc_{j \in S_i} \vec{z}_{M_j} = \vec{0}$
3. **Linear Relations**:
   for all j, $\vec{z}_{M_j} = \mathbf{M}_j \vec{z}$.

$$\sum_{i=0}^{q-1} c_i \cdot \bigcirc_{j \in S_i} (\mathbf{M}_j \vec{z}) = \vec{0}$$

From Algebraic Relations to Polynomials

- $\mathbb{H} = \{h_0, \ldots, h_{n-1}\} \subset \mathbb{F}_p^*$, multiplicative subgroup

$$\lambda_i(X) = \prod_{j \neq i} \frac{(X - h_j)}{(h_i - h_j)}, \qquad v_{\mathbb{H}}(X) = \prod_j (X - h_j).$$

Algebraic Formulation	Polynomial Formulation
Vector $\vec{y} = (y_0, \ldots, y_{n-1})$	Polynomial $Y(X) = \sum_{i=0}^{n-1} y_i \lambda_i(X)$
Public Input: $\vec{z}, \vec{w}$ agree on l positions	$Z(X) - W(X)$ is divisible by $v_l(X)$
Hadamard Product $\vec{a} \circ \vec{b} = \vec{c}$	$A(X)B(X) - C(X)$ is divisible by $v_{\mathbb{H}}(X)$
Inner product $\sigma = \vec{f} \cdot \vec{g}$	[Ben-Sasson et al. 18] $\exists R(X)$, $deg\ R(X) \leq n - 2$ s.t $v_{\mathbb{H}}(X)$ divides $f(X)g(X) - n^{-1}\sigma - XR(X)$

We can immediately build a non-interactive IOP for any of these relations.

Proving Linear Constraints

in Universal Preprocessing SNARKS

Statement: $\vec{y} = \mathbf{M}\vec{z}$.

Plonk, Hyperplonk, Plonky
Permutation-based arguments
$\mathbf{M}$ is a permutation

$$\prod(X + y_i) = \prod(X + z_i).$$

Private Computation

Marlin, Fractal, Spartan
Lincheck-Based Arguments:
Reduce many to one relation and use inner product

$$\vec{y} = \mathbf{M}\vec{z} \iff r^\top \cdot \vec{y} = (\vec{r}^\top \mathbf{M})\vec{z},$$

w.h.p. if $\vec{r}$ sufficiently random

Private and Public Computation

1) Private: $\vec{r}^\top \cdot \vec{y} = (\vec{r}^\top \mathbf{M})\vec{z}$
2) Public: $r^\top \mathbf{M}$ correct.

There are advantages in Lin-Check Based Arguments:

(1) **In Recursive Proof Composition.**

(1) **In Privacy Preserving Delegation of SNARK Provers**

LinCheck Based Arguments

e.g. Marlin

Commit

Commit to witness $\vec{z}$

$\eta_A, \eta_B, \eta_C, \alpha$

Outer sumcheck

Commit to terms to prove Hadamard, and $\vec{r}^\top(\mathbf{M}\vec{z}) = \vec{r}^\top\vec{y}$

β

Inner sumcheck

Prove $r^\top\mathbf{M}$ is correct

Open Polynomials

$$\mathbf{M} = \begin{pmatrix} \mathbf{A} \\ \mathbf{B} \\ \mathbf{C} \end{pmatrix}$$

$$\vec{r} = \begin{pmatrix} \eta_A \vec{\lambda}(\alpha) \\ \eta_B \vec{\lambda}(\alpha) \\ \eta_C \vec{\lambda}(\alpha) \end{pmatrix}.$$

$$\vec{r}^\top\mathbf{M} \leftrightarrow t(X) = \vec{r}^\top\mathbf{M}\vec{\lambda}(X)$$

$$\Pi = (\pi_{succ}, \pi_{PC}, \pi_{Lin})$$

$$b_{succ} \wedge b_{PC} \wedge b_{Lin} \leftarrow \mathcal{V}(x, SRS_{\mathcal{V}}, \Pi)$$

Overhead

Inner sumcheck

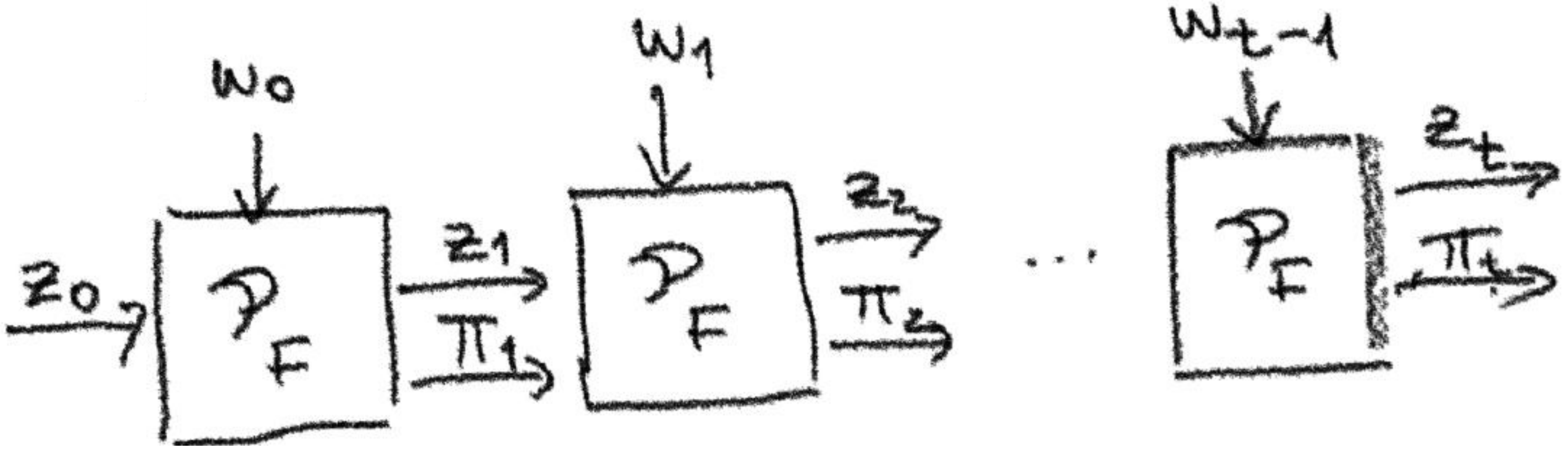
Prove $r^\top \mathbf{M}$ is correct

Cost of Inner Sumcheck: O(K) MSMs, O(K log K) Field

K = c |m.gates|, c small constant, e.g. c=1.5, 2, 3

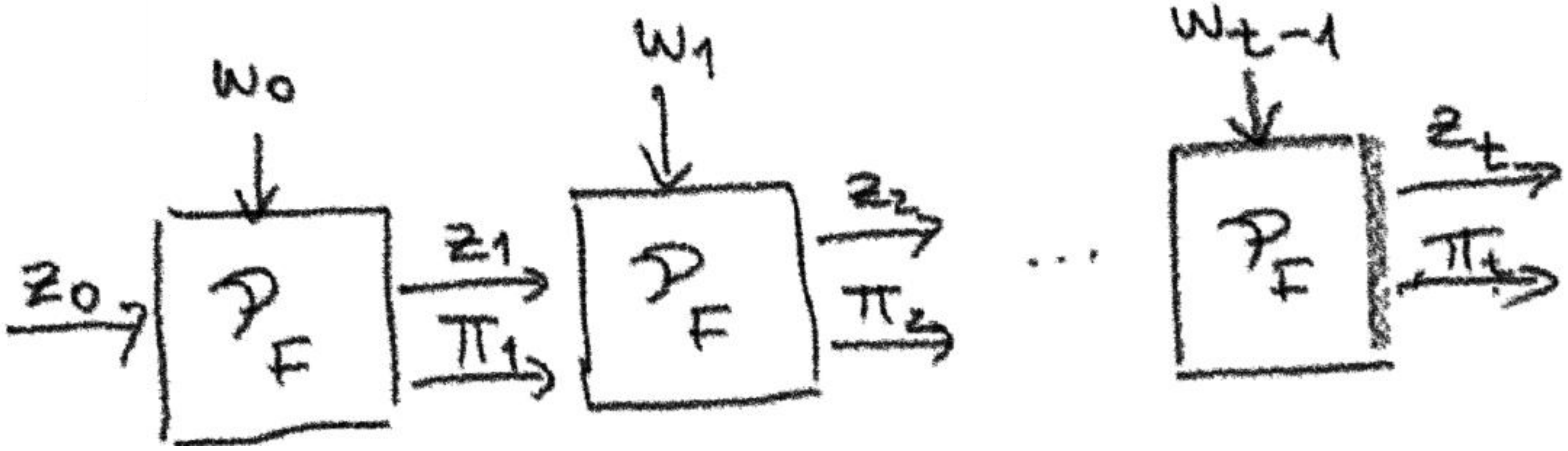
Recursive Proofs

Incrementally Verifiable Computation

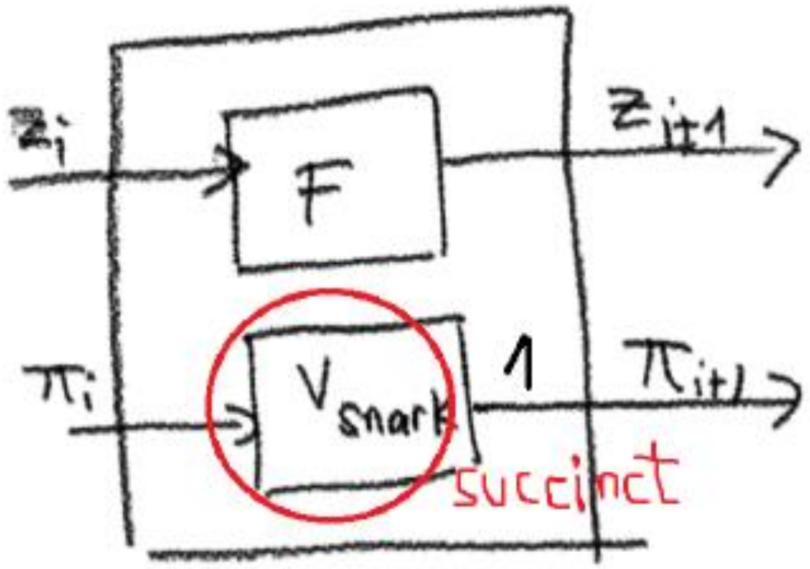


Checking a t-step non-deterministic computation, i.e. (F, z_0, z_t), check if $\exists w_0, \ldots, w_{t-1}, z_1, \ldots, z_{t-1}$ such that $\forall i = 0, \ldots, t-1,\ F(z_i, w_i) = z_{i+1}$.

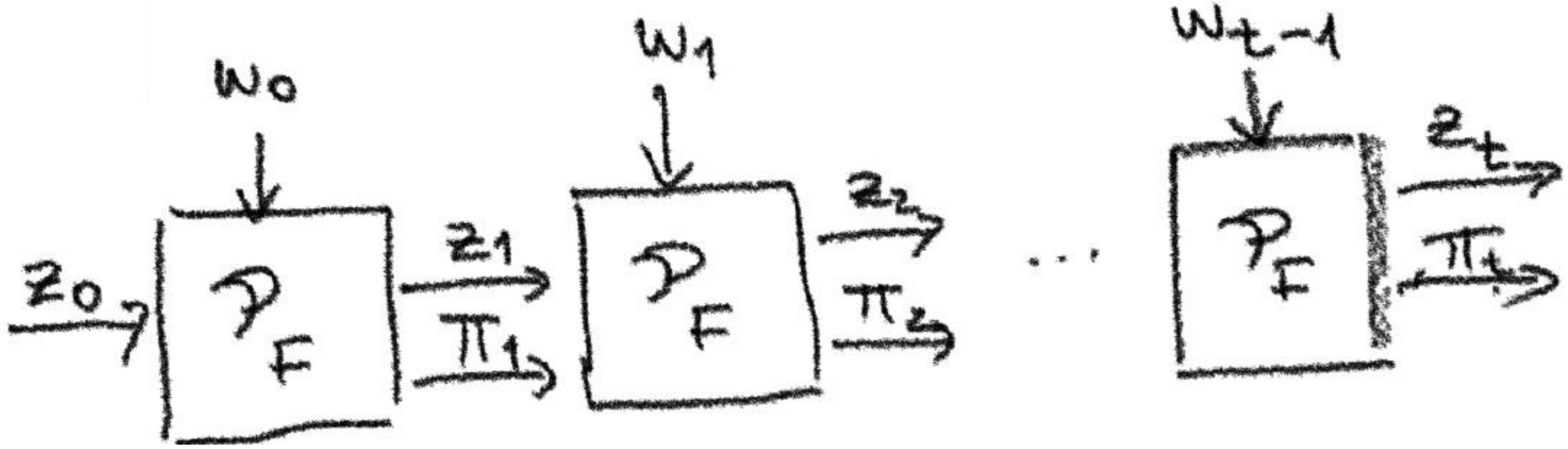
Incrementally Verifiable Computation



In each step, prove that computation is correct, and a proof that the proof of previous computations verifies.



Incrementally Verifiable Computation



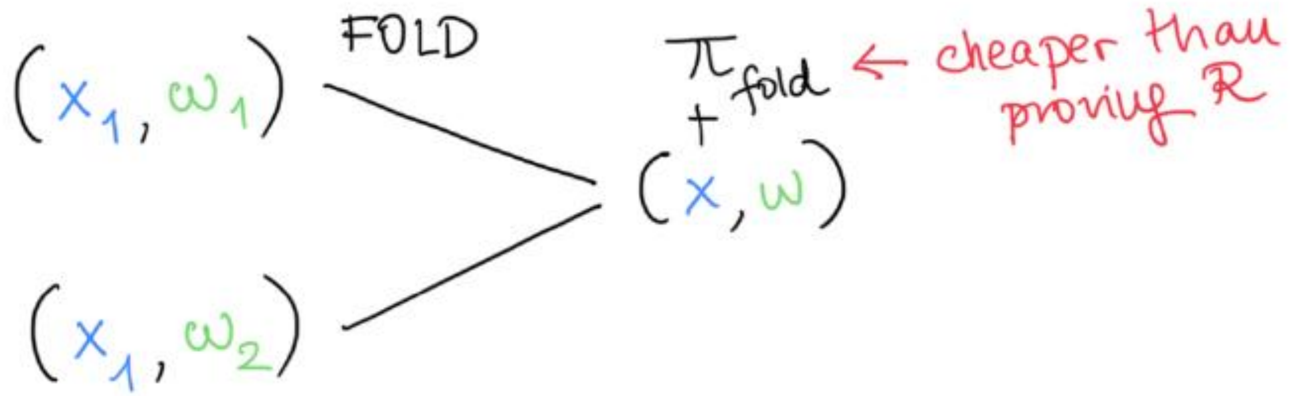
Checking a t-step non-deterministic computation, i.e. (F, z_0, z_t), check if $\exists w_0, \ldots, w_{t-1}, z_1, \ldots, z_{t-1}$ such that $\forall i = 0, \ldots, t-1,\ F(z_i, w_i) = z_{i+1}$.

Novel idea: defer part of the computation.

Folding / Split Accumulation

NP language $\mathcal{L}$ with corresponding relation $\mathcal{R}$.

- $\mathsf{Fold}(x_1, w_1, x_2, w_2) \rightarrow x, w, \pi_{\mathsf{Fold}}$

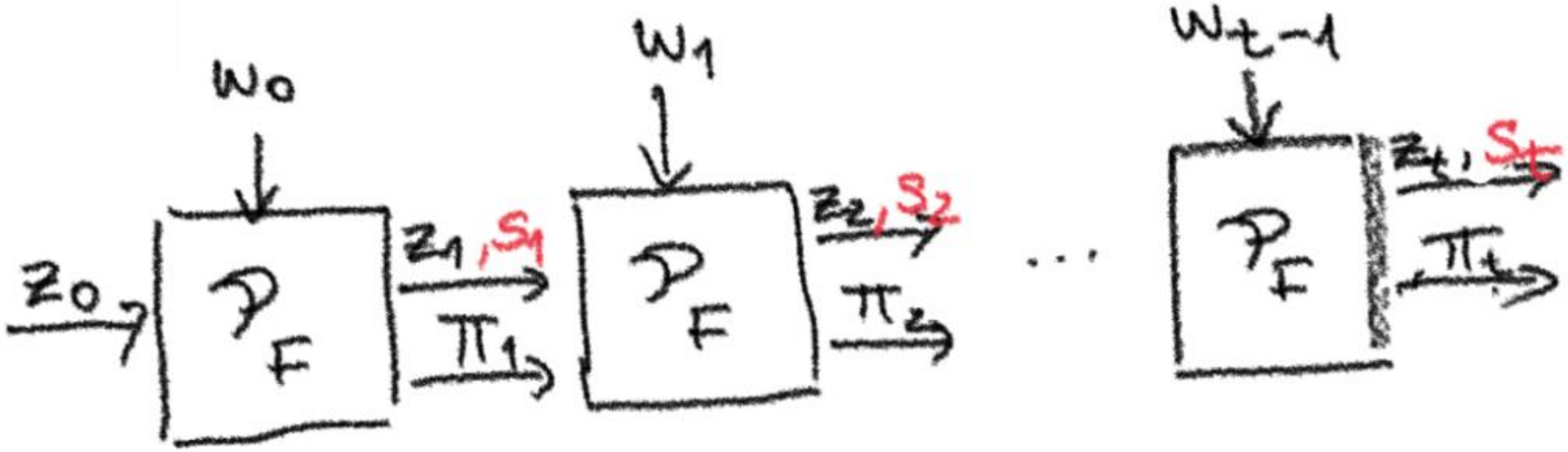


- (**Knowledge soundness**): If $\mathsf{FoldVrfy}(x_1, x_2, x, \pi_{\mathsf{Fold}}) \rightarrow 0/1$, then

$$(x, \omega) \in R \implies \begin{matrix} (x_1, \omega_1) \in R \\ \text{and} \\ (x_2, \omega_2) \in R \end{matrix}$$

Parts of the prover not executed, claims are accumulated into one final claim.

Incrementally Verifiable Computation



Checking a t-step non-deterministic computation, i.e. (F, z_0, z_t), check if $\exists w_0, \ldots, w_{t-1}, z_1, \ldots, z_{t-1}$ such that $\forall i = 0, \ldots, t-1,\ F(z_i, w_i) = z_{i+1}$.

Novel idea: defer part of the computation.

State-of-the-Art

$\boxed{b_{succ} \wedge b_{PC} \wedge b_{Lin} \leftarrow \mathcal{V}(x, SRS_{\mathcal{V}}, \Pi)}$

(1)**Full Recursion:**

- π_i SNARK proofs
- V verifies π_i
- Fractal,Plonky2

(2)**Atomic Accumulation:**

- π_i SNARK proofs
- V partially verifies π_i
- Halo

b_{PC} not fully checked.

(3)**Folding/Split Accumulation:**

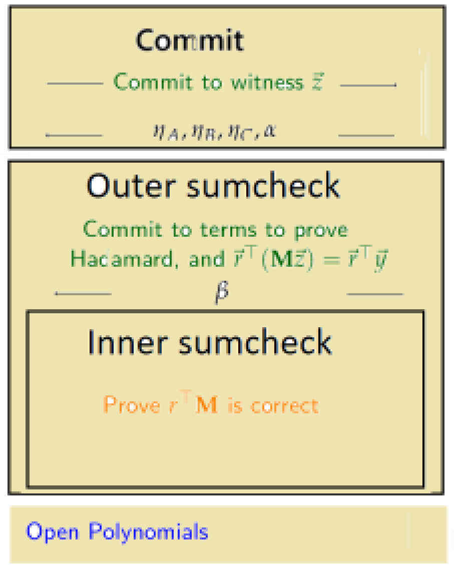
- π_i commitment to witness + state s_i
- V verifies correct folding, i.e. RLC of commitments $-- >$ V small
- Nova, ...

State-of-the-Art

$$b_{succ} \wedge b_{PC} \wedge b_{Lin} \leftarrow \mathcal{V}(x, SRS_{\mathcal{V}}, \Pi)$$

(1) Full Recursion:

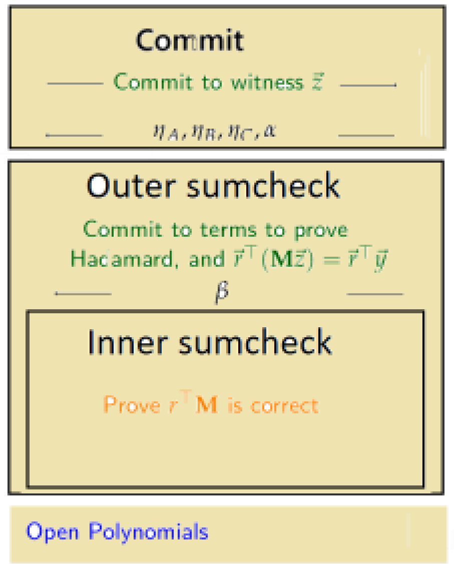
- π_i SNARK proofs
- V verifies π_i
- Fractal, Plonky2



(2) Atomic Accumulation:

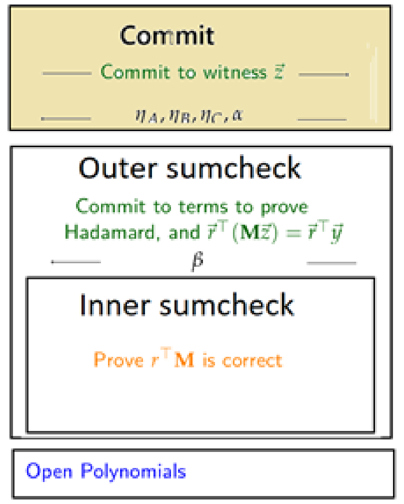
- π_i SNARK proofs
- V partially verifies π_i
- Halo

b_{PC} not fully checked.



(3) Folding/Split Accumulation:

- π_i commitment to witness + state s_i
- V verifies correct folding, i.e. RLC of commitments $-->$ V small
- Nova, ...



HOW MUCH OF SNARK PROVER IS EXECUTED

State-of-the-Art REVISITED

$$b_{succ} \wedge b_{PC} \wedge b_{Lin} \leftarrow \mathcal{V}(x, SRS_{\mathcal{V}}, \Pi)$$

(1) **Full Recursion:**

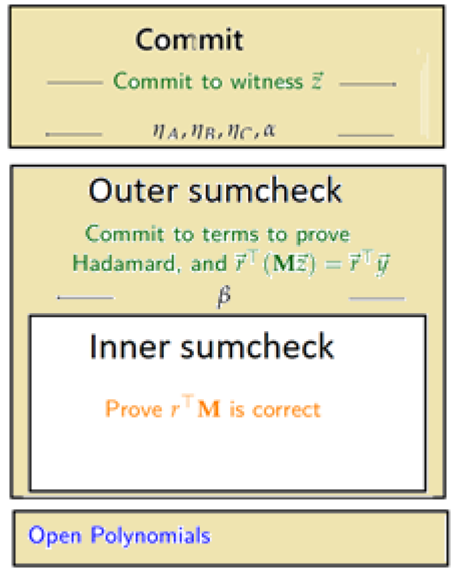
- π_i SNARK proofs
- V verifies π_i
- Fractal, Plonky2

(2) **Atomic Accumulation:**

- π_i SNARK proofs
- V partially verifies π_i
- Halo

b_{PC} not fully checked.

- Darlin: b_{Lin} not checked.



(3) **Folding/Split Accumulation:**

- π_i commitment to witness + state s_i
- V verifies correct folding, i.e. RLC of commitments $-- >$ V small
- Nova, ...

HOW MUCH OF SNARK PROVER IS EXECUTED

Holography Accumulation

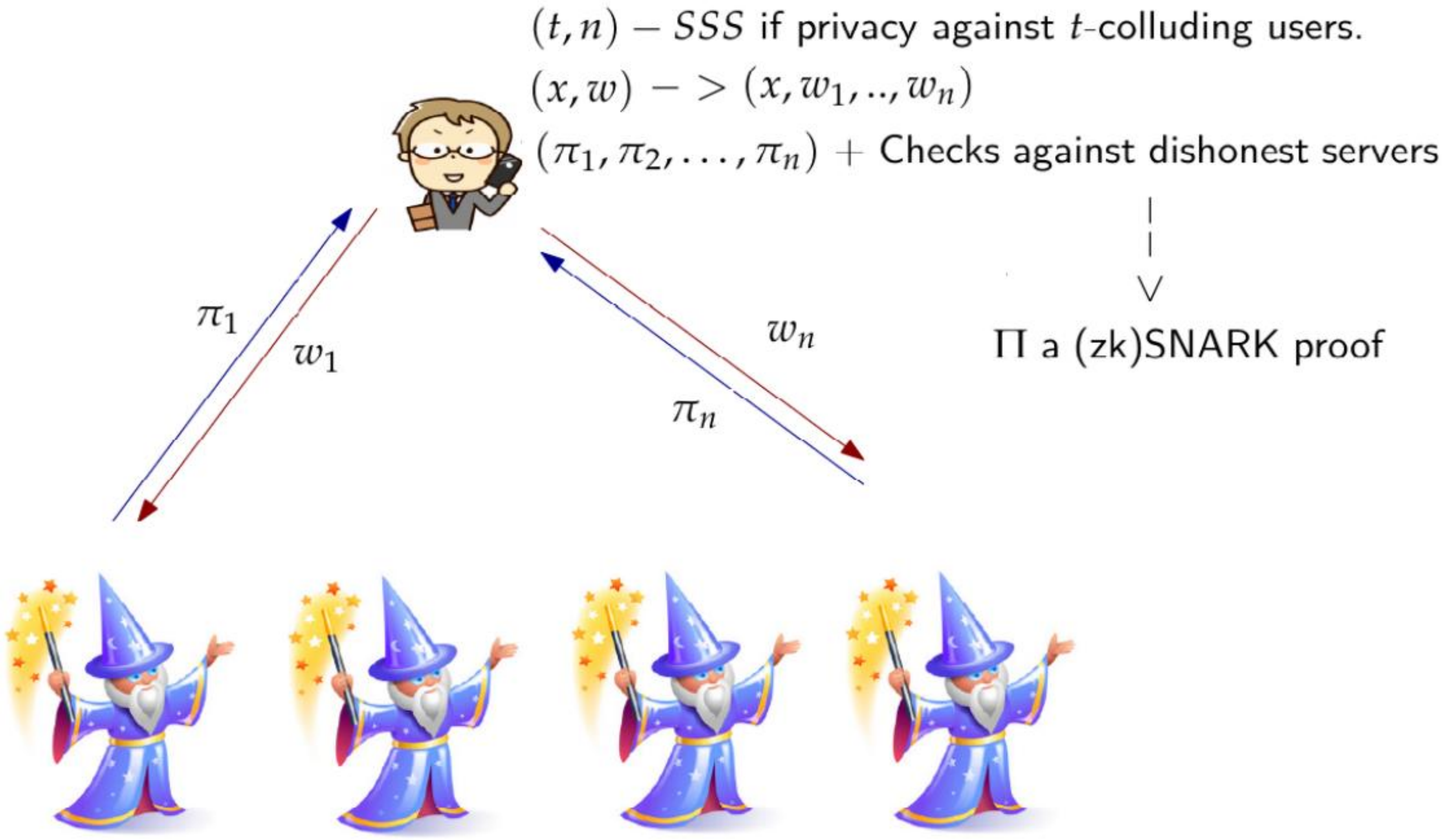
Results in Recursive Proof Composition

- Revisit Darlin: accumulate predicate b_{Lin}
- Generalize to CCS
- No witness information must be passed on to next computation stage;
- Right alternative to full recursion or just Halo-style atomic accumulation.
- In half-cycles with one pairing friendly curve, we show how to fold multiple relations building on two-tier (pairing based commitments), **avoid sparsity assumption!**

Privacy Preserving Delegation of Computation

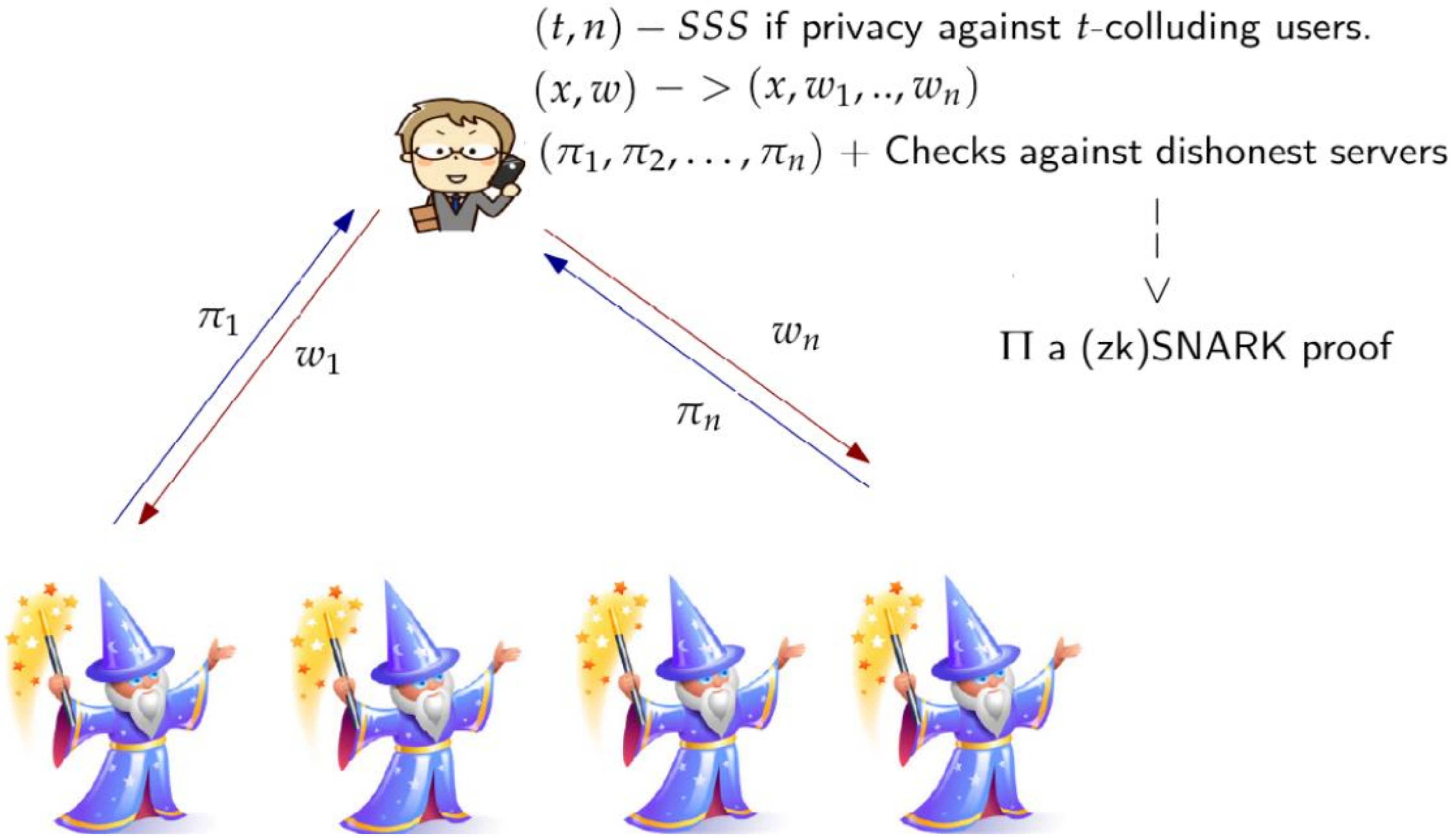
Privacy Preserving SNARK Proof Delegation

Blueprint



Privacy Preserving SNARK Proof Delegation

Research Question



Scenario: Servers do computation as a service for many users, amortize some of the work?

Privacy Preserving SNARK Proof Delegation

Mar-lin

Commit

Commit to witness $\vec{z}$ $\longrightarrow$

$\longleftarrow$ $\eta_A, \eta_B, \eta_C, \alpha$

Outer sumcheck

Commit to terms to prove Hadamard, and $\vec{r}^\top(\mathbf{M}\vec{z}) = \vec{r}^\top\vec{y}$

$\longleftarrow$ β

Inner sumcheck

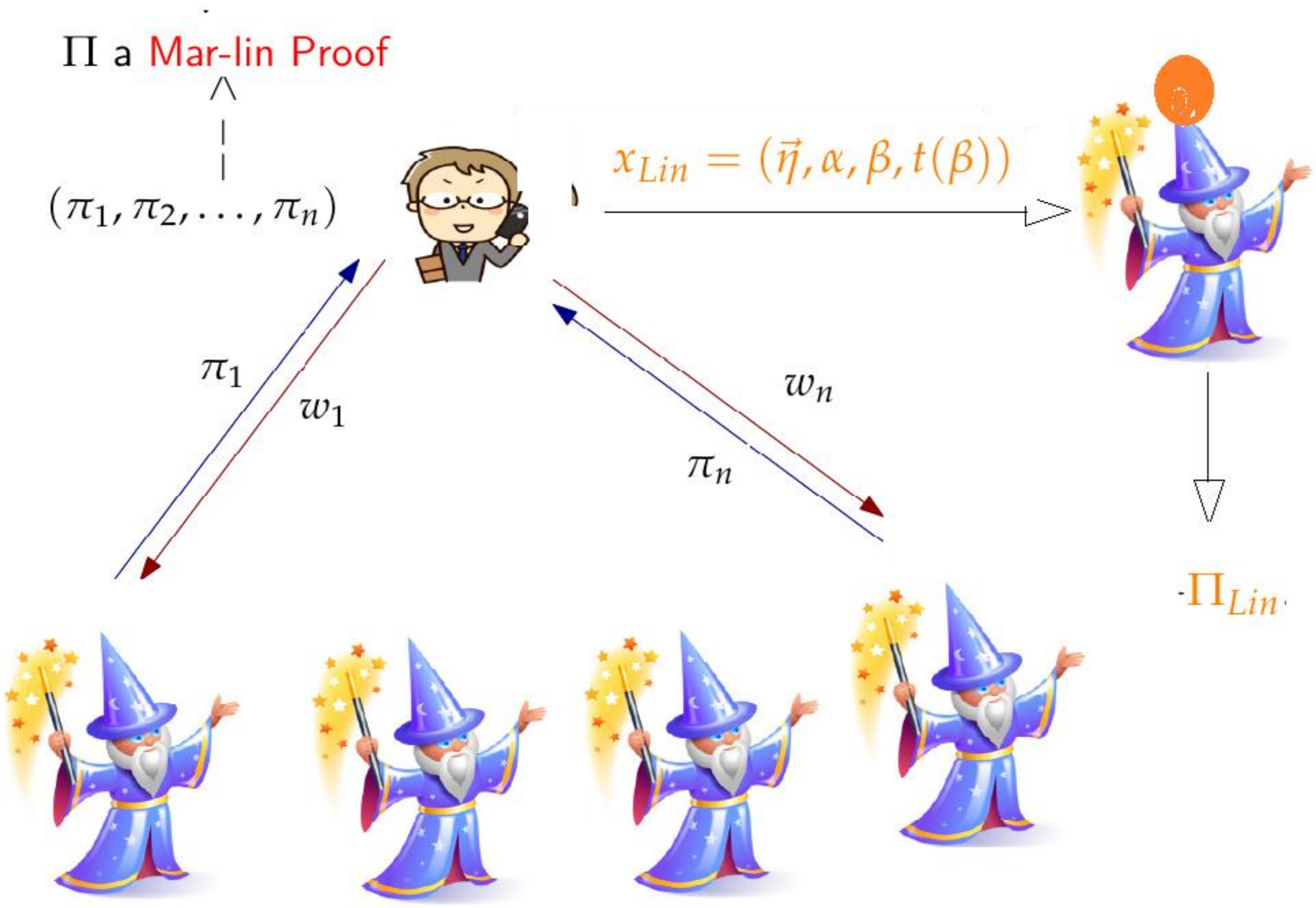
Prove $r^\top\mathbf{M}$ is correct

Open Polynomials

Witness Dependent Computation

Privacy Preserving SNARK Proof Delegation

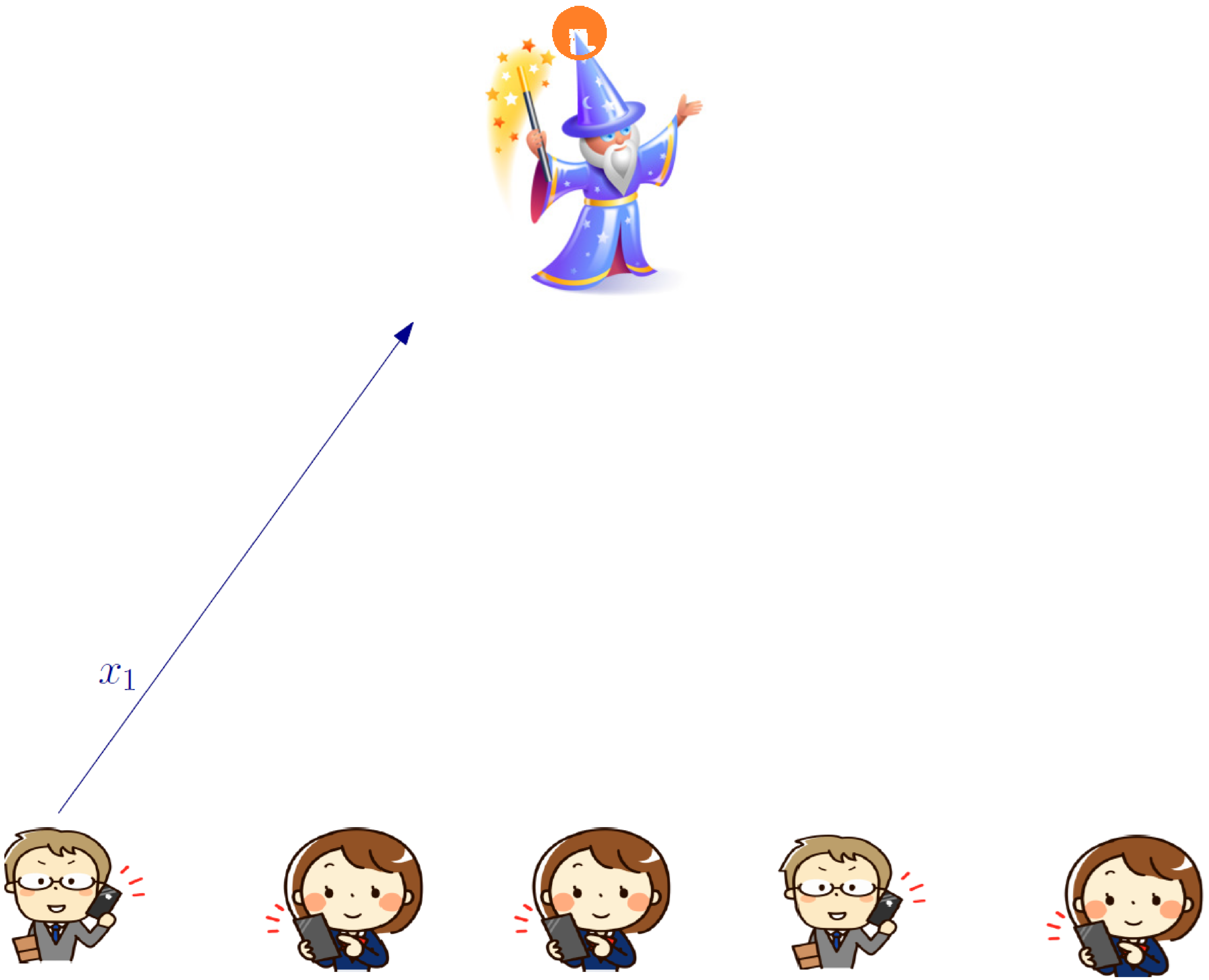
Revisited



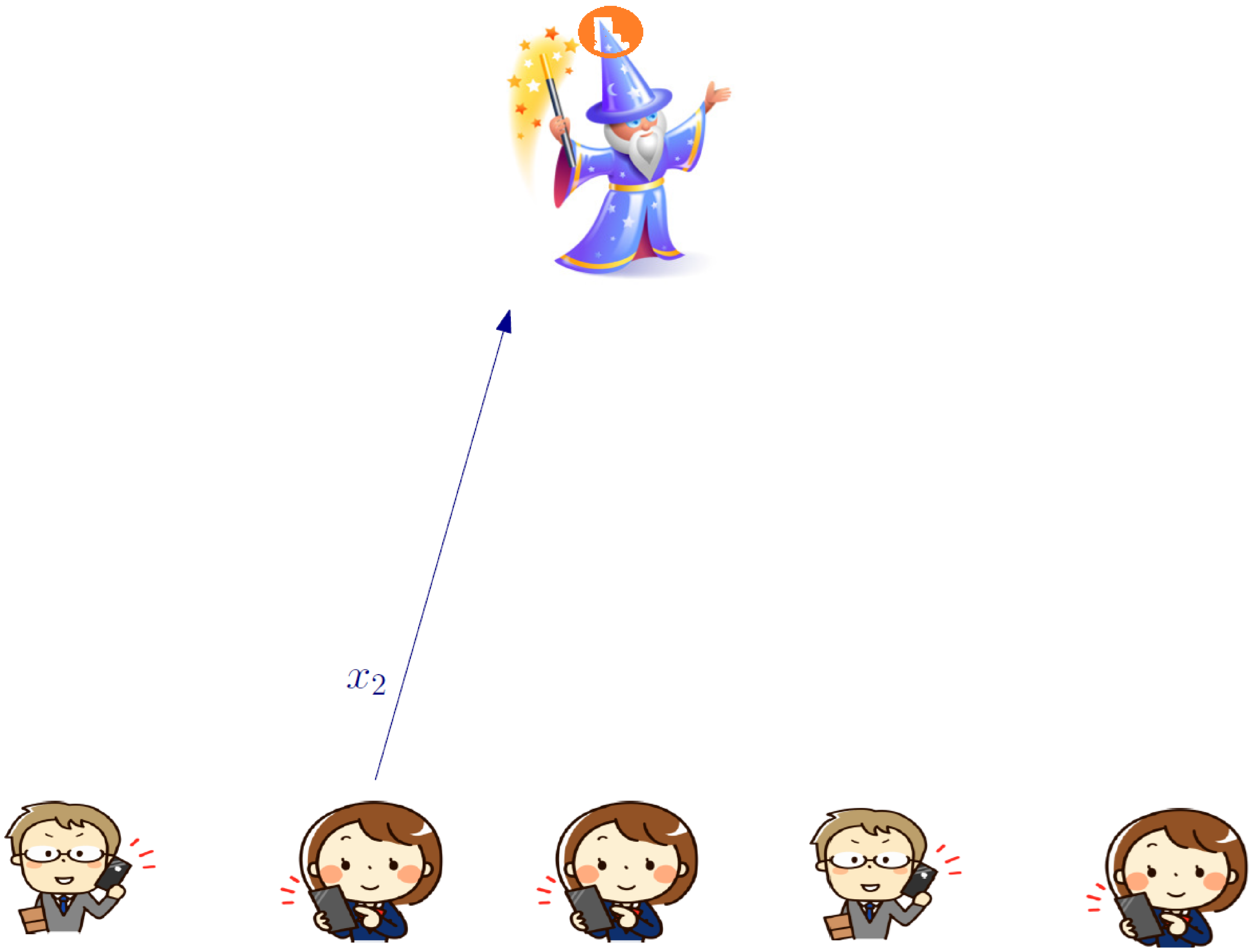
- Delegate public computation (INNER SUMCHECK) to a single powerful server.
- A Mar-lin proof can then be computed locally or delegated using privacy-preserving techniques.
- Verification checks $\Pi + \Pi_{Lin}$

IDEA: Accumulate INNER SUMCHECK to reduce computation per proof

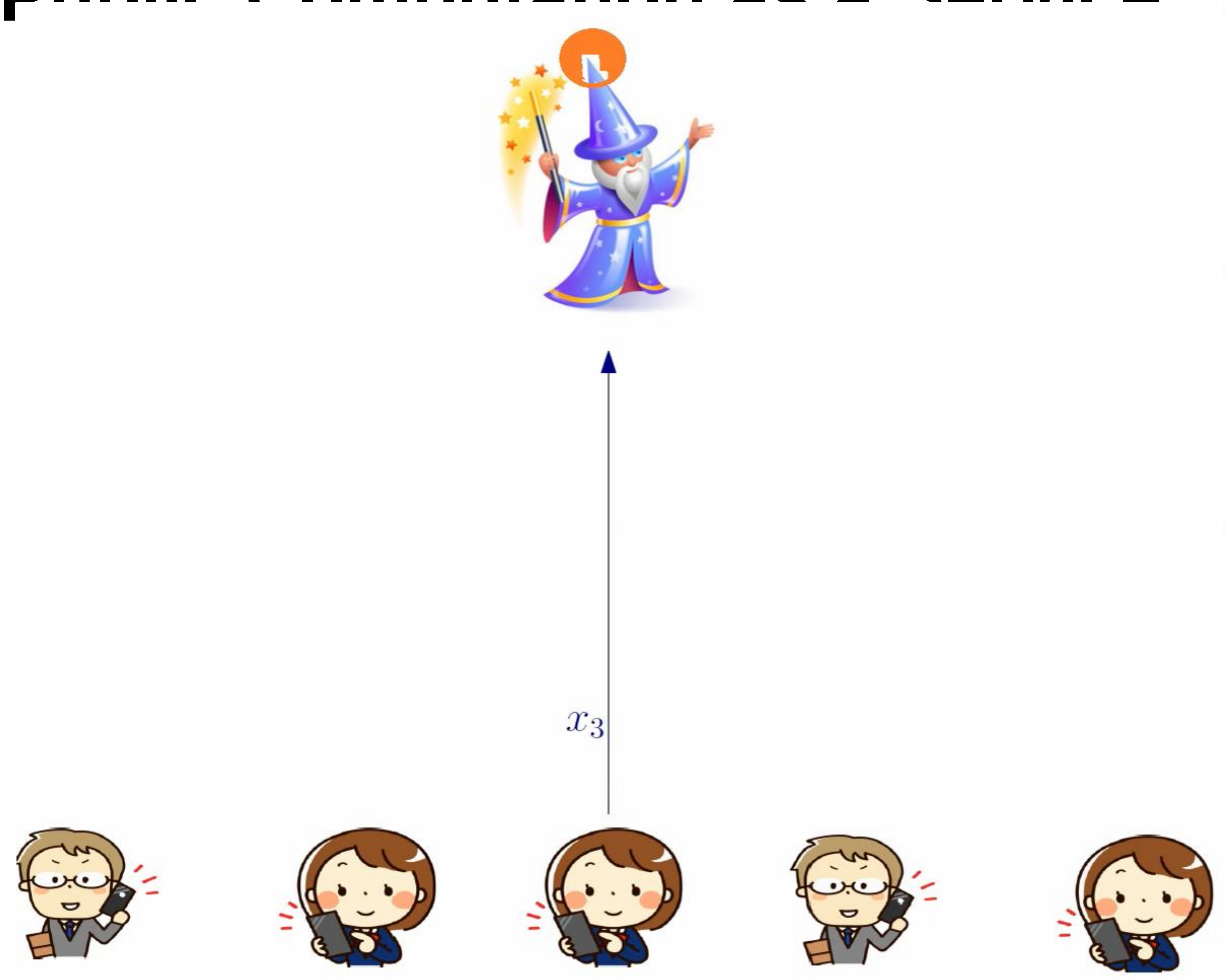
Public Computation as a Service



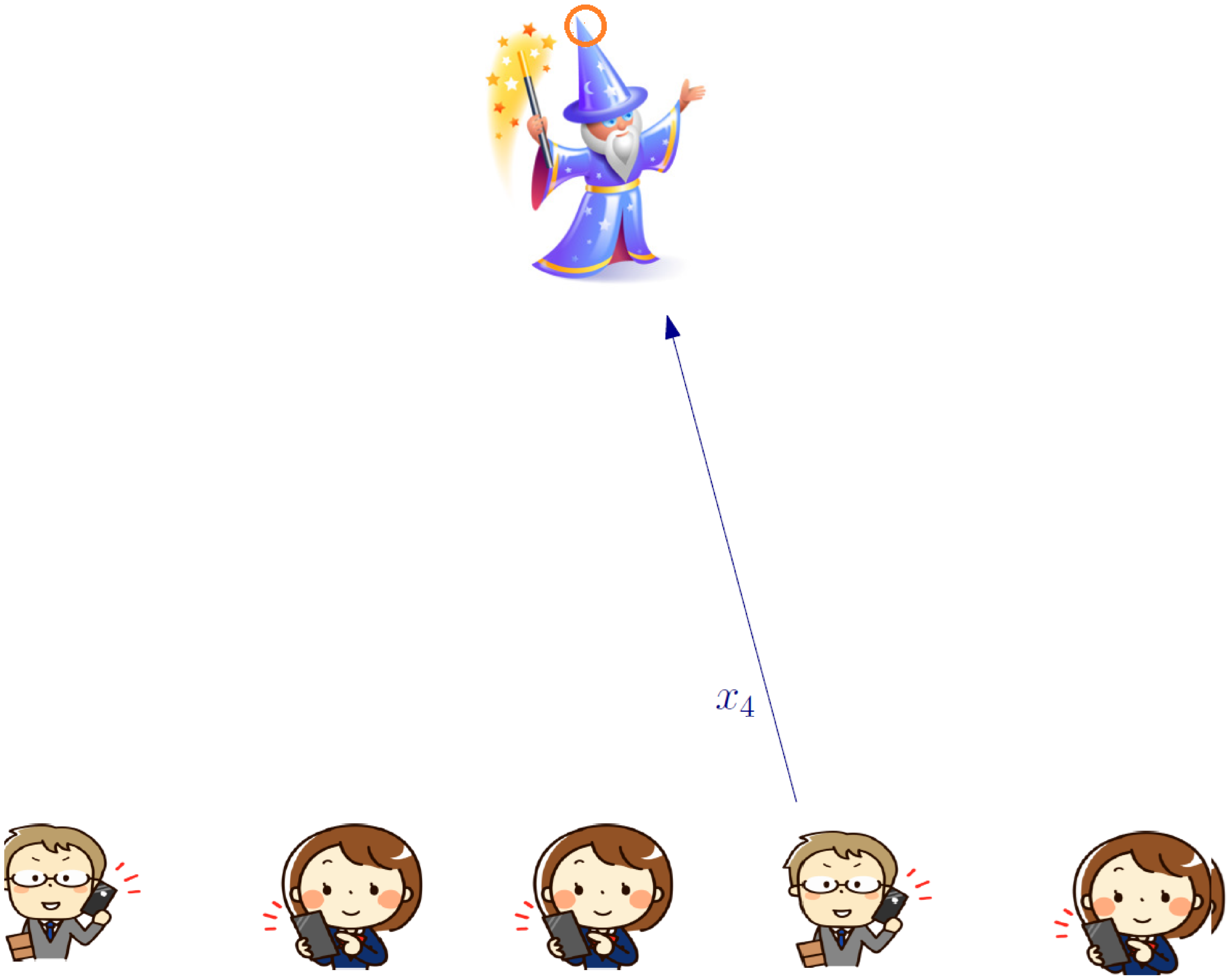
Public Computation as a Service



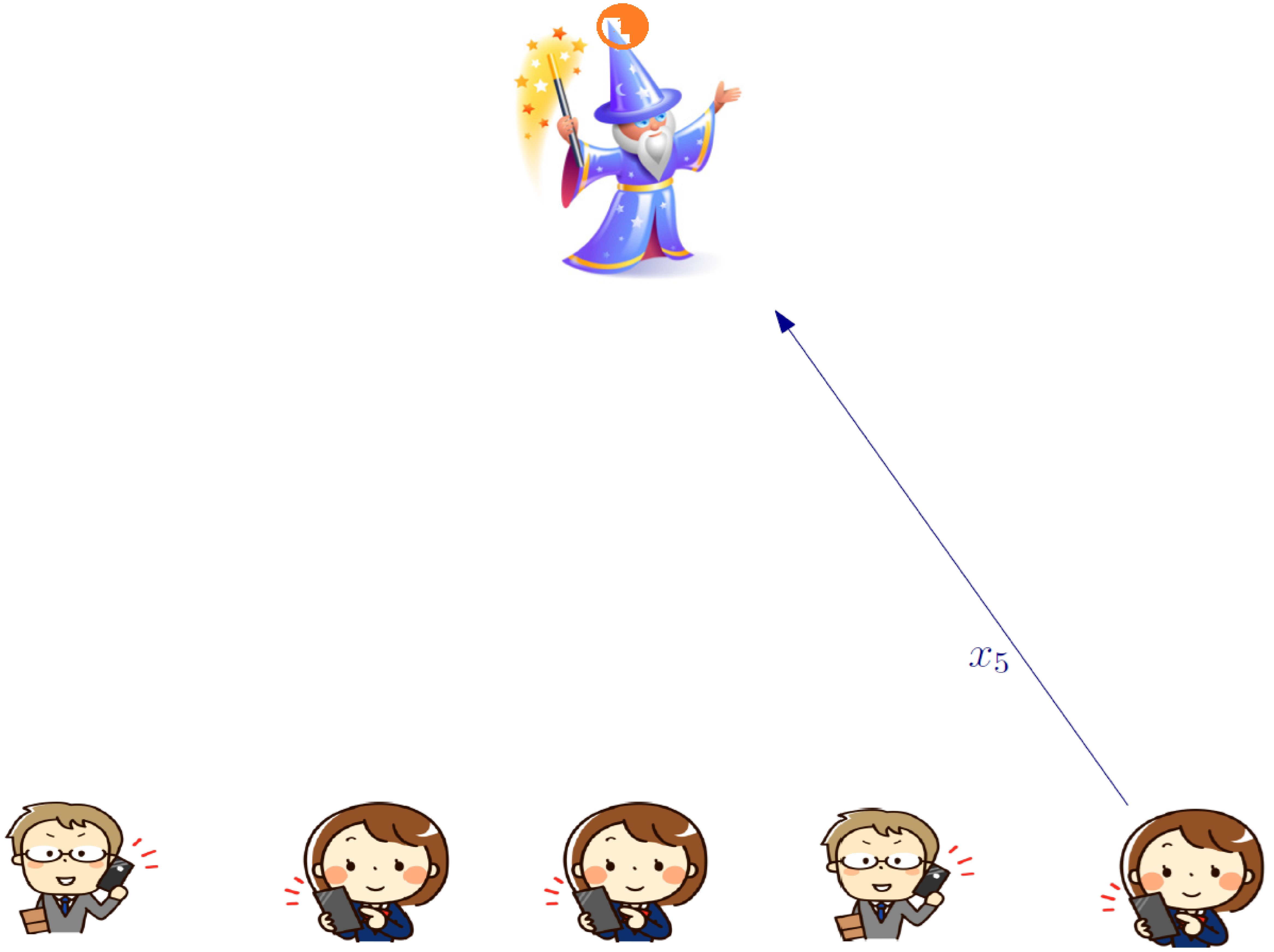
Public Computation as a Service



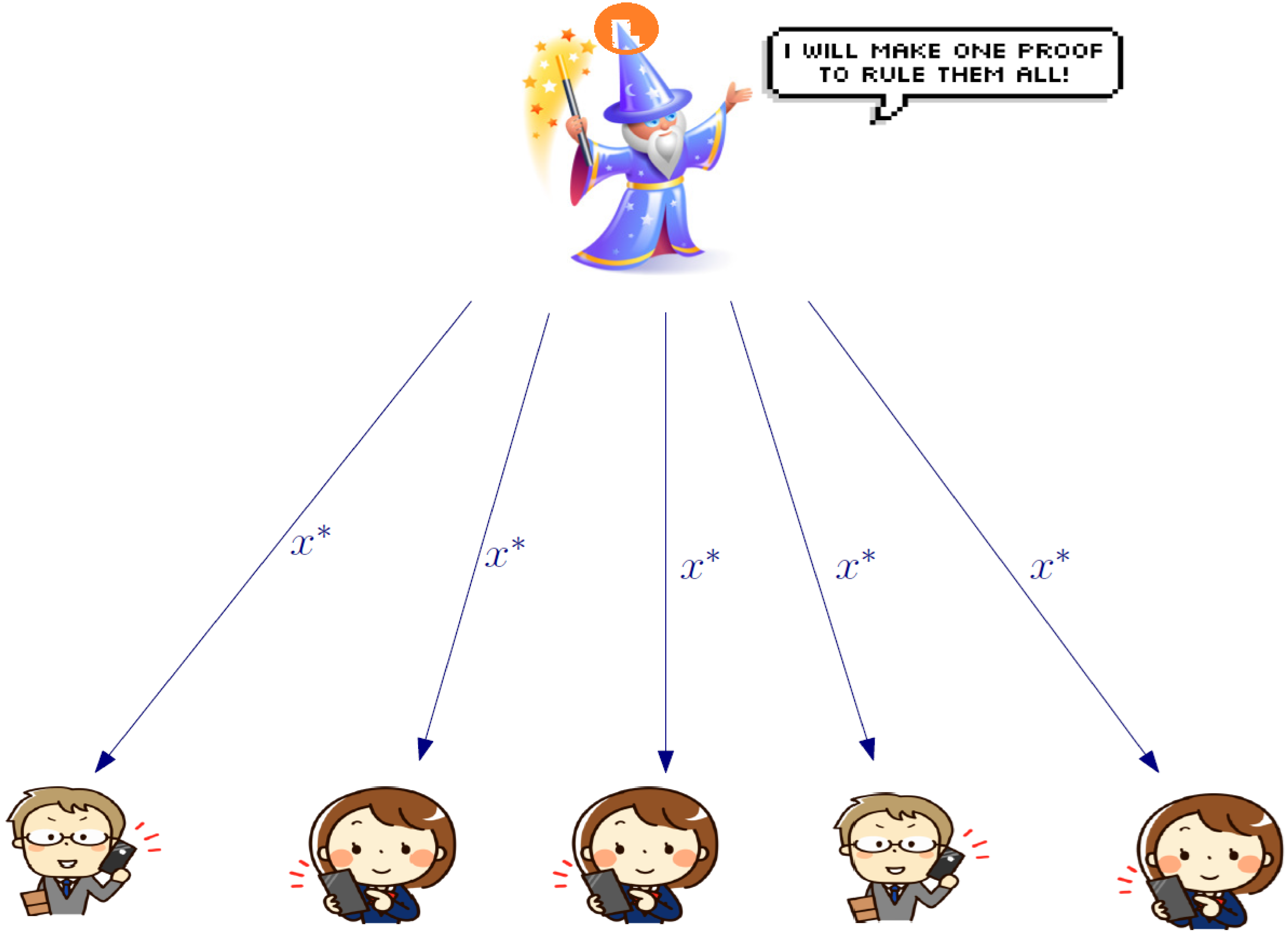
Public Computation as a Service



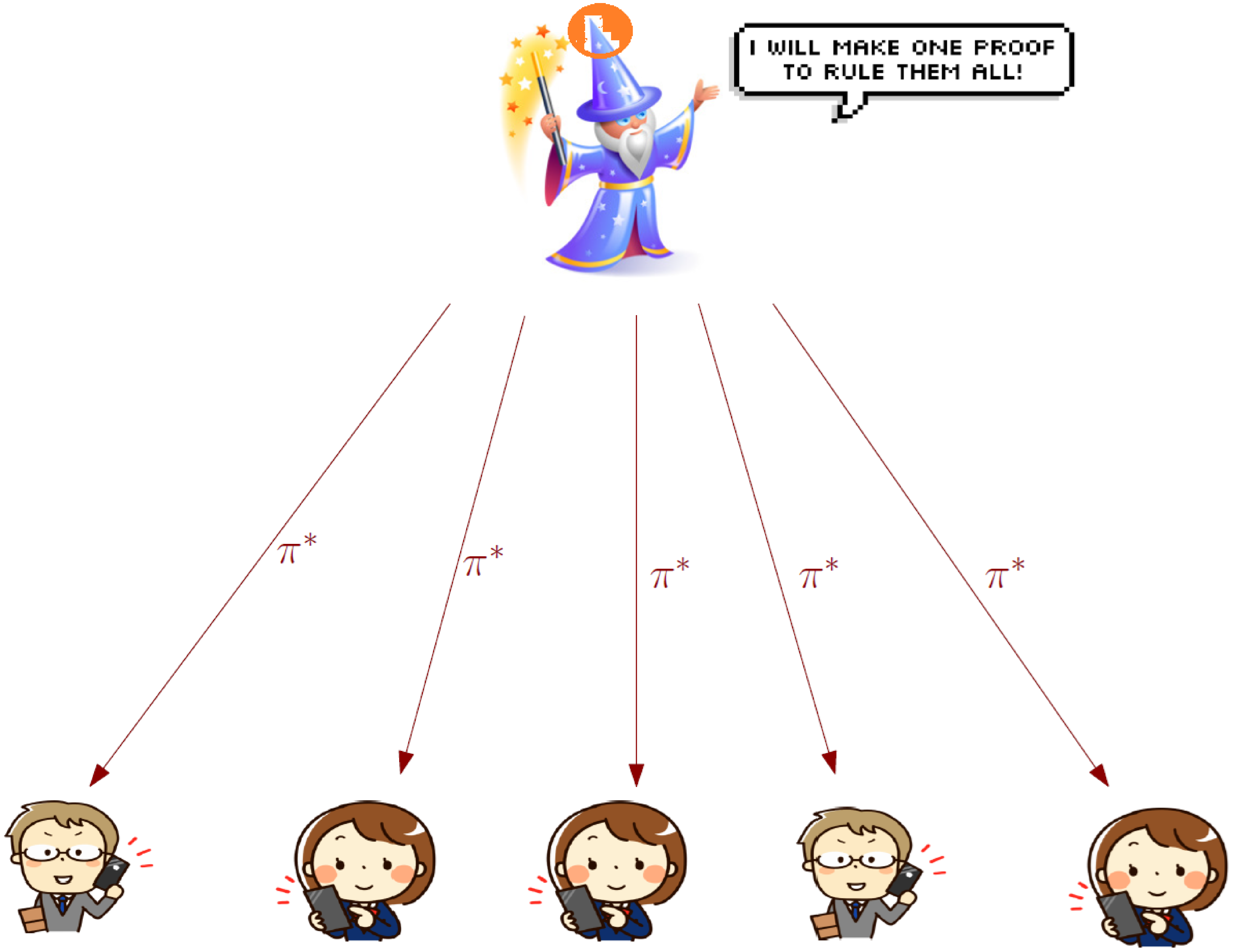
Public Computation as a Service



Public Computation as a Service

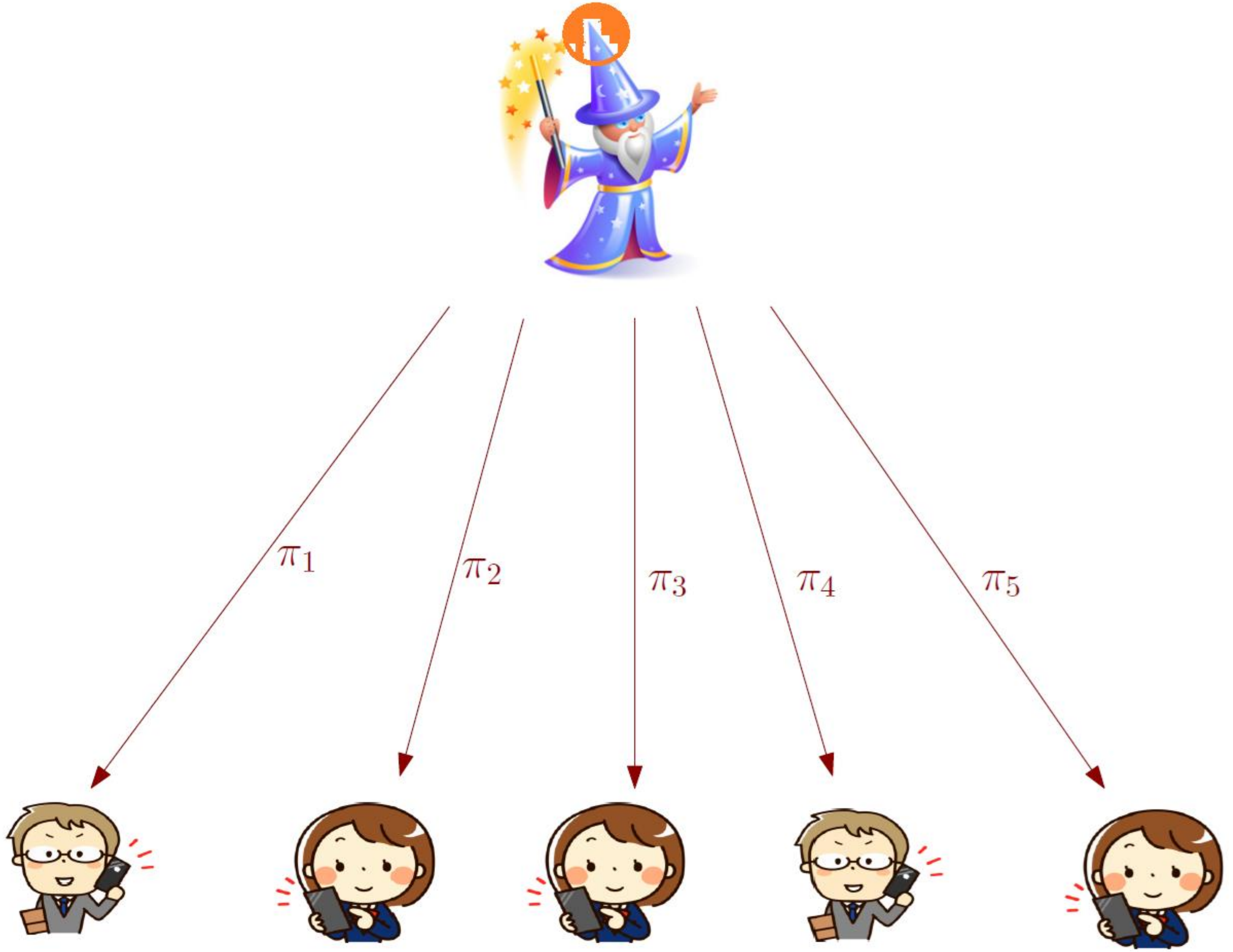


Public Computation as a Service



Users need to receive proof that final statement was derived from their individual statements.

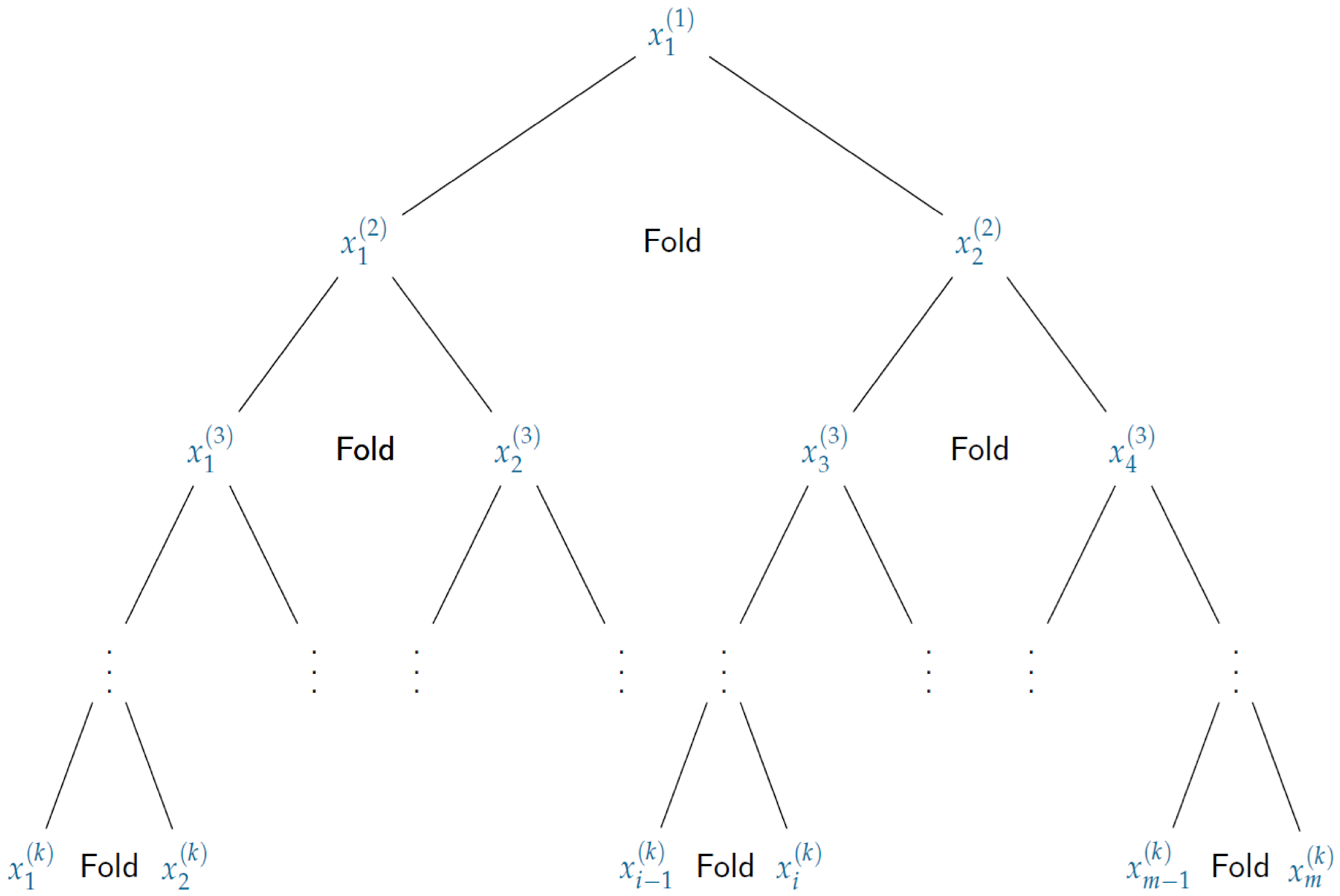
Public Computation as a Service



Naive solution:

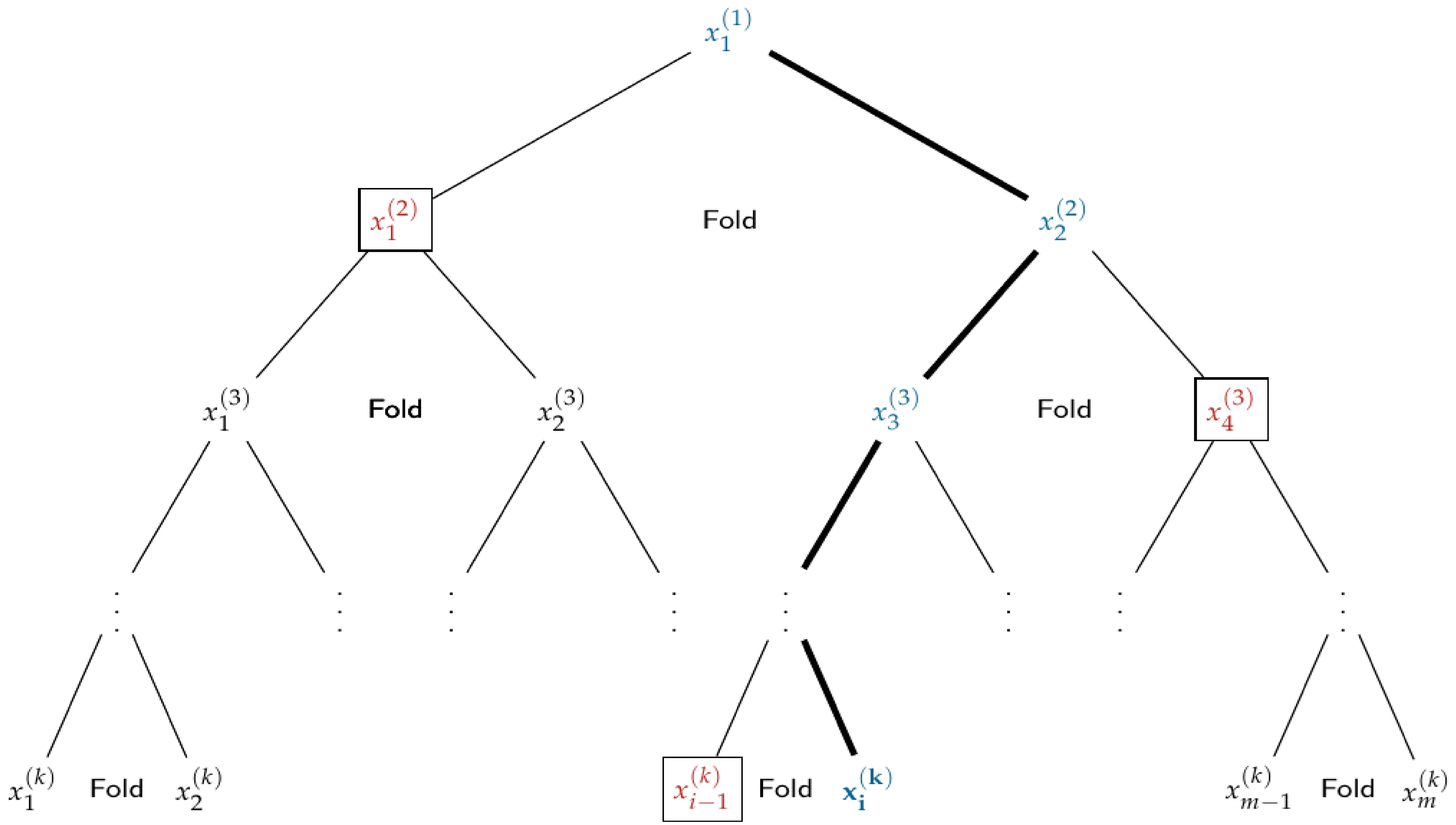
Proof that asserts that x_i is included in final statement requires knowing all of user's statements

Folding Schemes with Local Verification



x_i

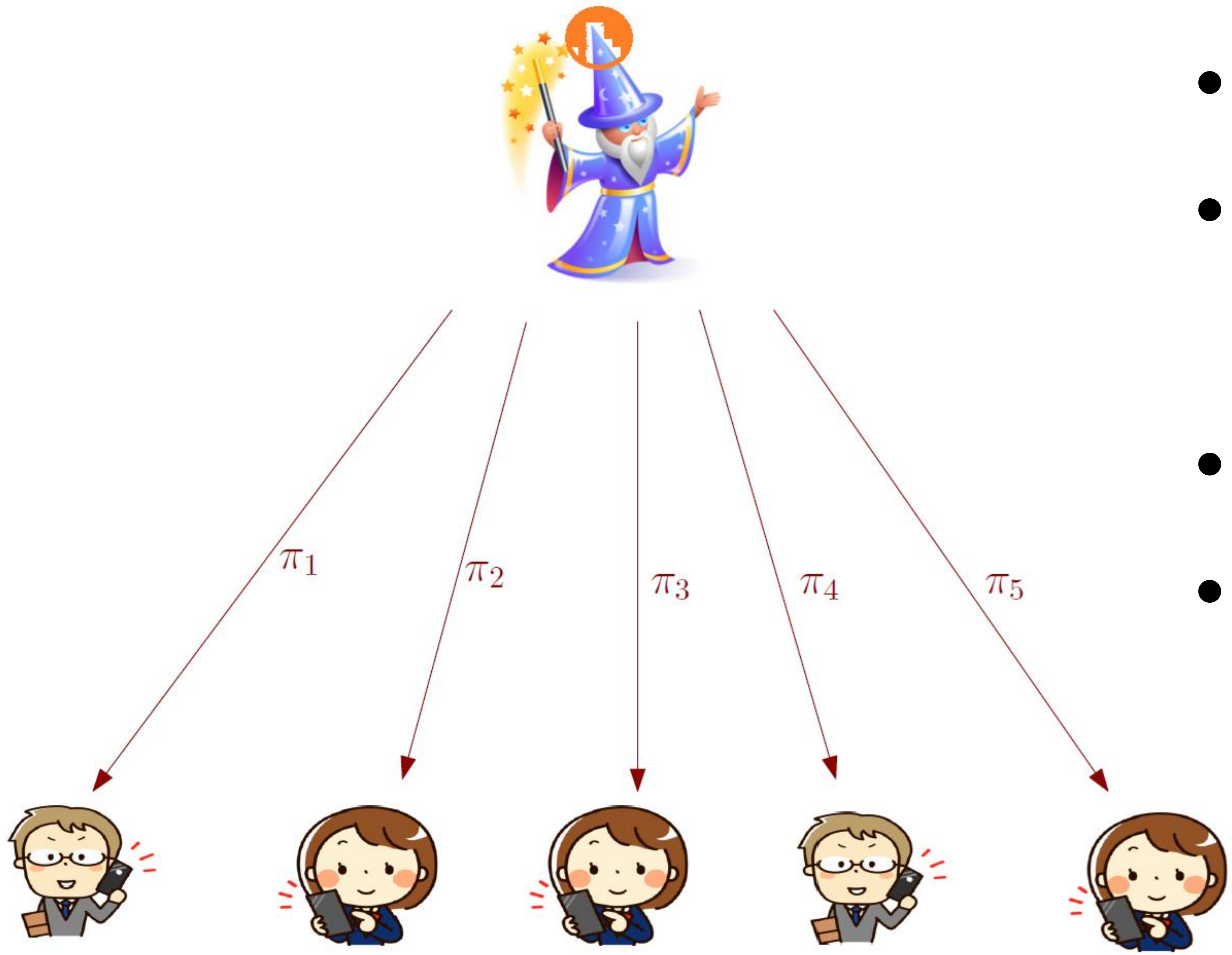
Folding Schemes with Local Verification



Give as *proof* the sibling statements & 2-folding proofs AND Prove only root statement.

Prover: $2m$ foldings + proof root./ Verifier: verify $\pi_i = O(\log m)$ + one proof.

Public Computation aas with FS with Local Verification



- Tradeoff cost Server - Verifier Π_{Lin}

x_i

- Local/Private part: closer to Groth16 (dominated by 5 MSMs of size |m.gates|)
- Verifier checks $\Pi + \Pi^*_{Lin} + \pi_i$.
- Total Amortized Cost x Proof: essentially local cost.

Results coming soon in your nearest IACR eprint

ZKPROOF7