Stream Computation and Arthur-Merlin Communication

Justin Thaler, Yahoo! Labs

Joint Work with:

Amit Chakrabarti, Dartmouth

Graham Cormode, University of Warwick

Andrew McGregor, Umass Amherst

Suresh Venkatasubramanian, University of Utah

Outsourcing

- Many applications require outsourcing computation to untrusted service providers.
    - Main motivation: commercial cloud computing services.
    - Also, weak peripheral devices; fast but faulty co-processors.
    - Volunteer Computing (SETI@home, World Community Grid, etc.)
- User requires a guarantee that the cloud performed the computation correctly.

AWS Customer Agreement

WE… MAKE NO REPRESENTATIONS OF ANY KIND … THAT THE SERVICE OR THIRD PARTY CONTENT WILL BE UNINTERRUPTED, ERROR FREE OR FREE OF HARMFUL COMPONENTS, OR THAT ANY CONTENT … WILL BE SECURE OR NOT OTHERWISE LOST OR DAMAGED.

Goals of Verifiable Computation

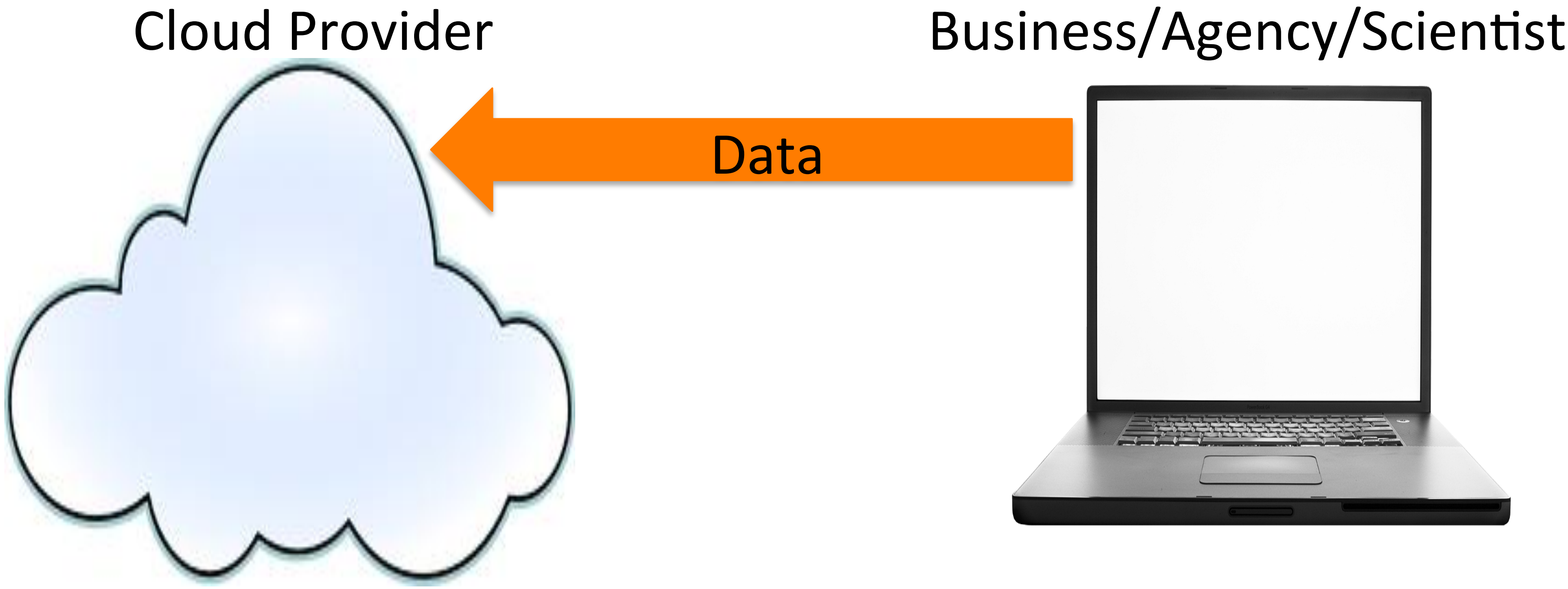
- Goal 1: Provide user with a correctness guarantee.
- Goal 2: User must operate within the restrictive **data streaming paradigm** (models a user who lacks the resources to store the input locally).

Interactive Proofs

Cloud Provider

Business/Agency/Scientist

Interactive Proofs

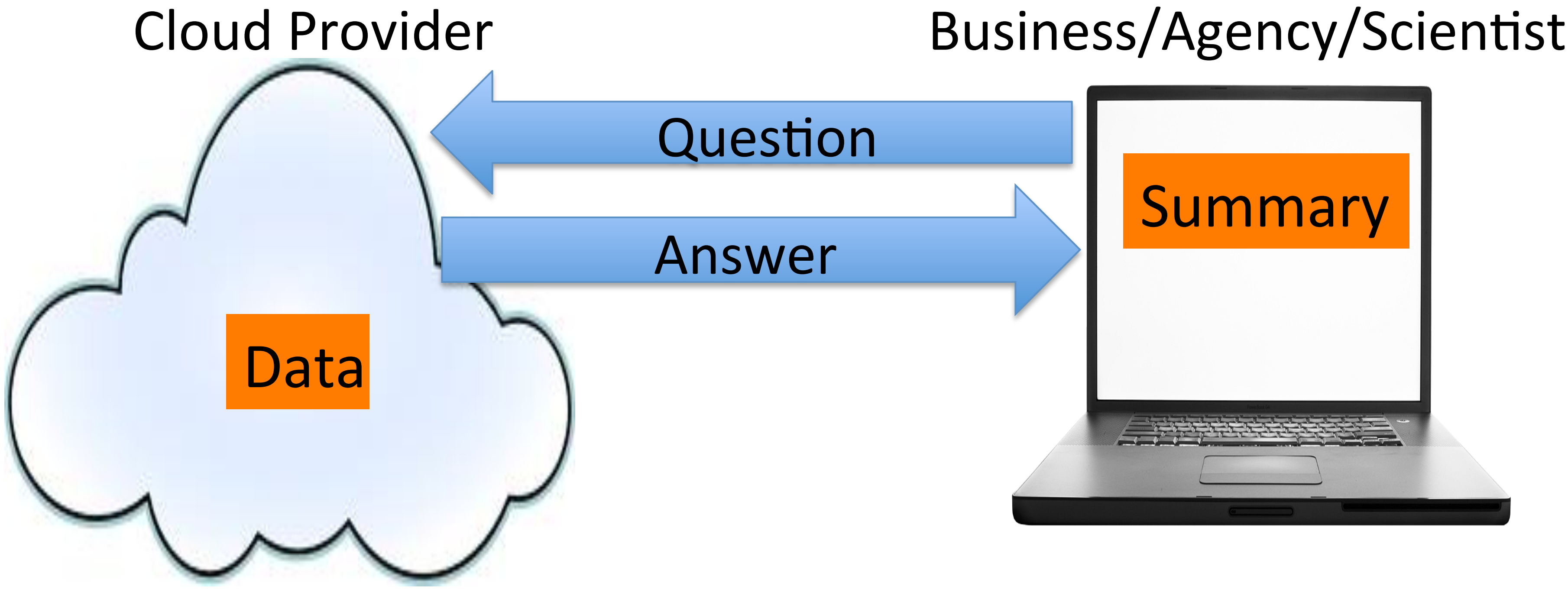


Interactive Proofs

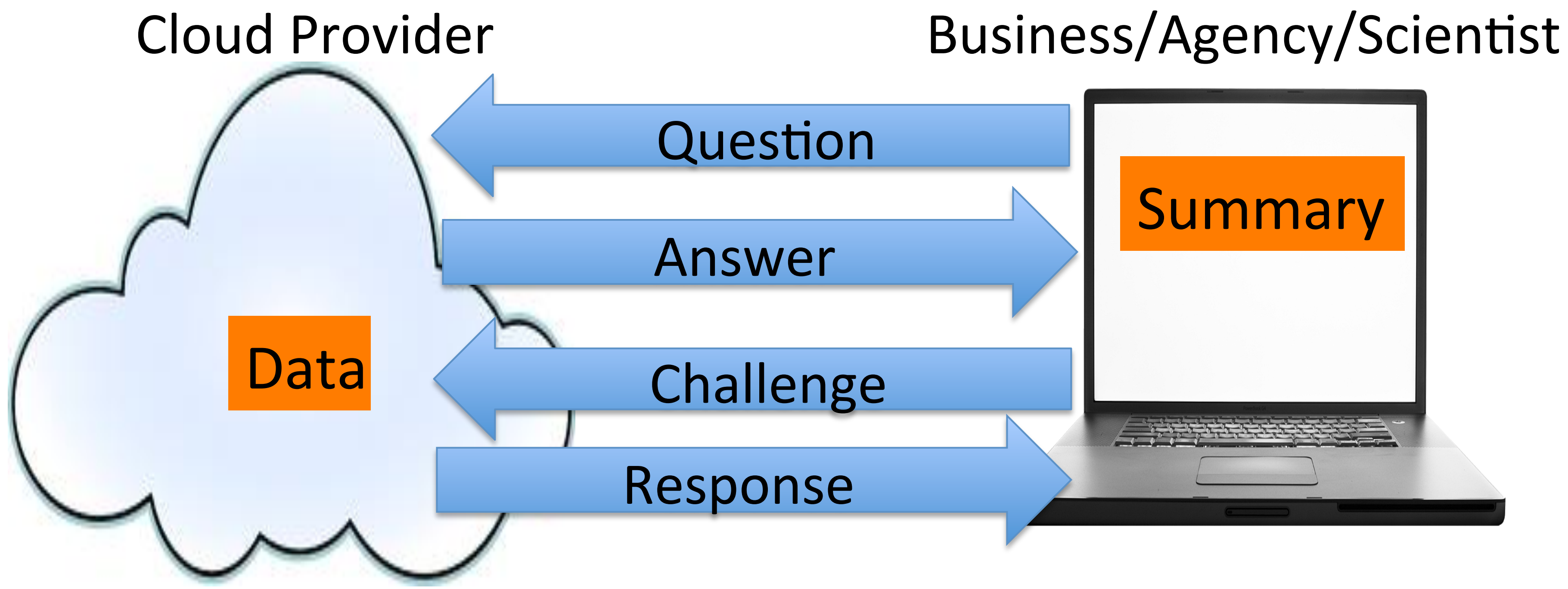
Cloud Provider

Business/Agency/Scientist

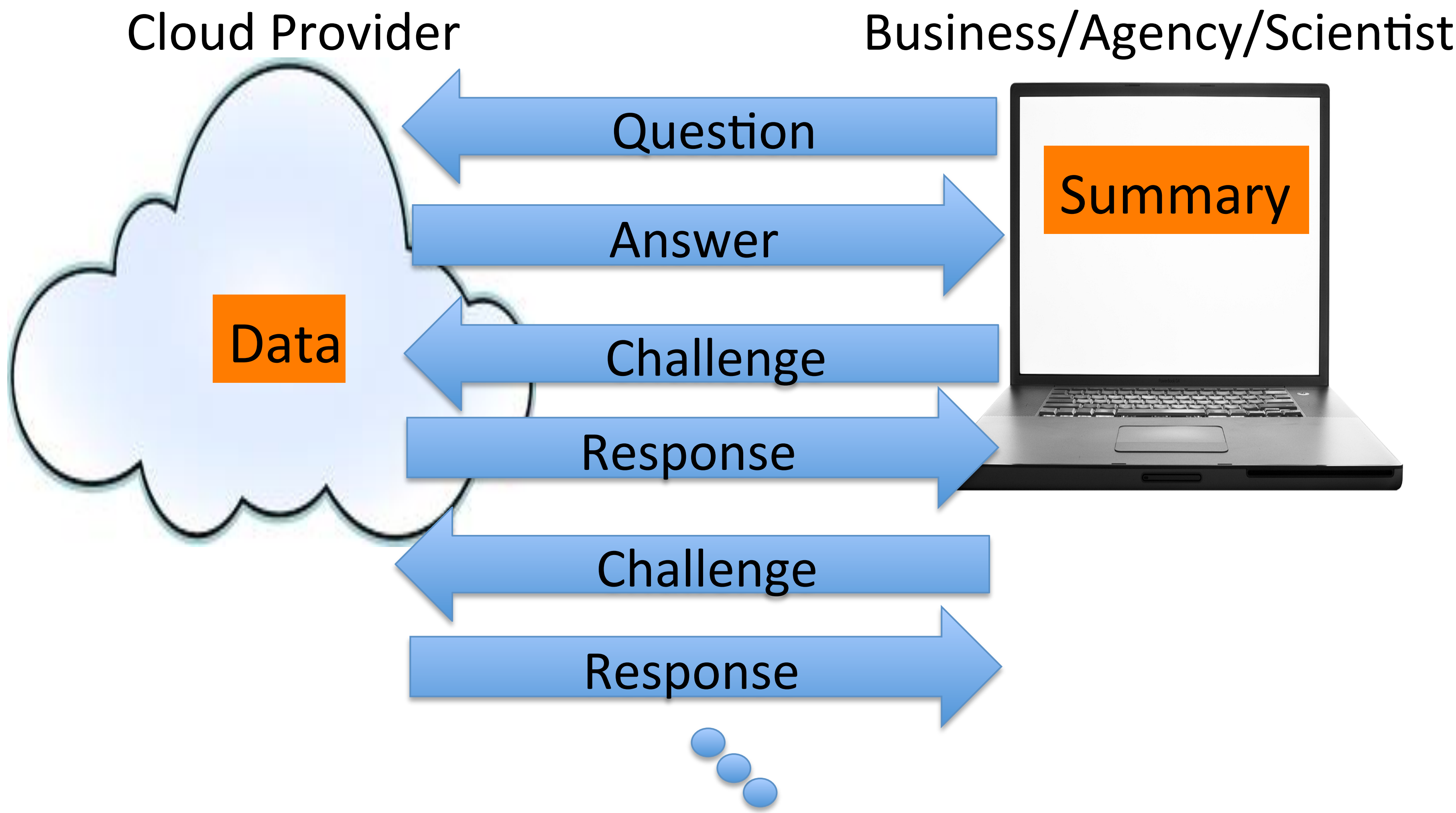
Interactive Proofs



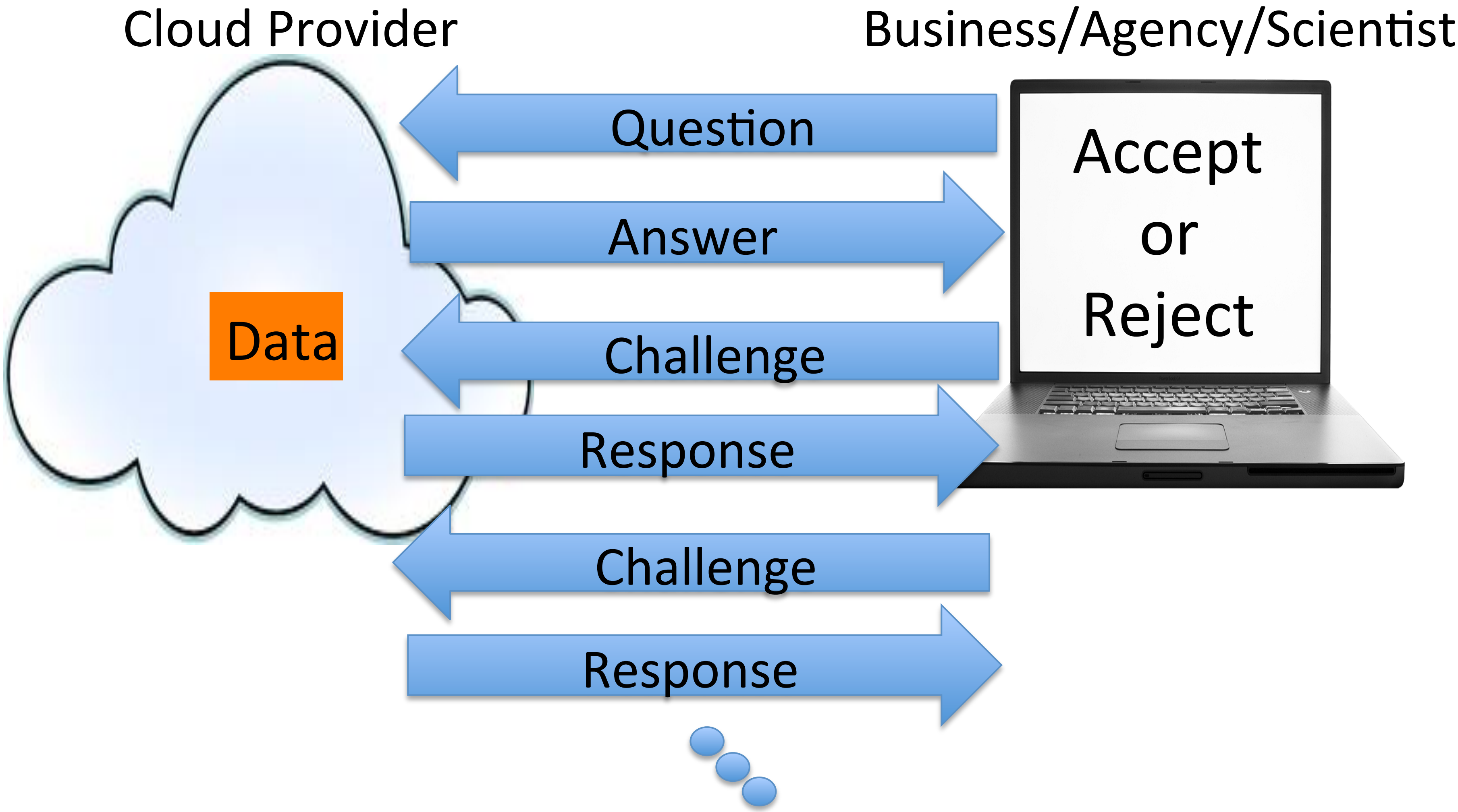
Interactive Proofs



Interactive Proofs



Interactive Proofs



Interactive Proofs

- Prover P and Verifier V.

- P solves problem, tells V the answer.
  - Then P and V have a conversation.
  - P's goal: convince V the answer is correct.

- Requirements:
  - 1. Completeness: an honest P can convince V to accept.
  - 2. Soundness: V will catch a lying P with high probability (secure even if P is computationally unbounded).

Streaming Interactive Proof (SIP) Model [CTY12]

- **After** both observe stream, P and V have a conversation.
- Fits cloud computing well: streaming pass by V can occur while uploading data to cloud.
- V never needs to store entirety of data.

Costs of SIPs

- Two main costs: amount communication, and V's working memory. Both must be **sublinear** (ideally **polylogarithmic**) in input size.
- Other costs: running time, number of messages.

History of Streaming Interactive Proofs

- [CTY12] introduced streaming interactive proofs (SIPs), gave logarithmic cost protocols for many problems.
- Earlier work [CCM09] had introduced a more restricted model corresponding to one-message SIPs.
- [KP13, GR13, CTY12, CCMTV14, KP14] study variants of these models.
- [CMT12] gave efficient implementations of protocols from [CCM09, CMT10] (and from the literature on "classical" interactive proofs).

Talk Outline

- Part 1: Exponentially more efficient two-message SIPs for many problems.
- Part 2: New communication models that allow us to investigate the **limitations** of constant-round SIPs.

Part I: Exponentially More Efficient Constant-Round SIPs

INDEX Problem

- Data stream specifies a vector $\mathbf{x}$ followed by an index $\mathbf{i}$. Goal is to output $\mathbf{x_i}$.
- Requires $\Omega(n)$ space in the standard streaming model.

Prior Work on SIPs for INDEX

- [CCM09/CCMT14]: A **1-message** protocol with space and comm. costs $O(\sqrt{n})$. Showed this is optimal.
- [CTY12]: A (2k-1)-message protocol with cost $O(n^{1/(k+1)})$.
- All of these protocols based on public-coin **sum-check** techniques [LFKN90].
- [KP13] claimed a matching lower bound for any k>0.

Prior Work on SIPs for INDEX

- [CCM09/CCMT14]: A **1-message** protocol with space and comm. costs $O(\sqrt{n})$. Showed this is optimal.
- [CTY12]: A (2k-1)-message protocol with cost $O(n^{1/(k+1)})$.
- All of these protocols based on public-coin **sum-check** techniques [LFKN90].
- ~~[KP13]: obtained matching lower bound for any k > 0.~~
- We show [KP13] lower bound only applies to "public coin" SIPs.
- We give a 2-message protocol with cost $O(\log n \log \log n)$.
- Protocol adapts result of [Raz05] on **IP/rpoly**. See also [CKLR11].
- Later, we'll build on this protocol to solve more complicated problems (NNS, RangeCount, PatternMatching, Median, etc).

The 2-message SIP for INDEX

A general technique

- Arithmetization: Given function g defined on small domain, replace g with its **multilinear extension** $\tilde{g}$ as a polynomial defined over a large field.

- Can view $\tilde{g}$ as error-corrected encoding of g: If two (boolean) functions differ in one location, their multilinear extensions will differ in almost all locations.

- Error-correcting properties give V considerable power over P.

The INDEX Problem

- Data stream specifies a vector $\mathbf{x}$ followed by an index $\mathbf{i}$. Goal is to output $\mathbf{x_i}$.

The INDEX Protocol, Part 1

- View $\mathbf{x}$ as a function mapping $\{0,1\}^{\log n} \to \{0,1\}$ via: $\mathbf{x}(j_1,\ldots,j_{\log n}) = \mathbf{x}_j$, where $(j_1,\ldots,j_{\log n})$ is the binary representation of j.
- Fix a finite field $\mathbf{F}$ of size at least $4\log n$.
- V picks a random vector $\mathbf{r} \in \mathbf{F}^{\log n}$, and evaluates $\tilde{\mathbf{x}}(\mathbf{r})$ in streaming pass over $\mathbf{x}$ (requires space $O(\log n \log |\mathbf{F}|)$).

How Can V Evaluate $\tilde{\mathbf{x}}(\mathbf{r})$?

- For each $\mathbf{j} \in \{0,1\}^{\log n}$, define $\delta_{\mathbf{j}} : \{0,1\}^{\log n} \to \{0,1\}$ via:

  $$\delta_{\mathbf{j}}(\mathbf{k}) := 1 \text{ if } \mathbf{j}{=}\mathbf{k} \text{ and } \delta_{\mathbf{j}}(\mathbf{k}) := 0 \text{ otherwise.}$$

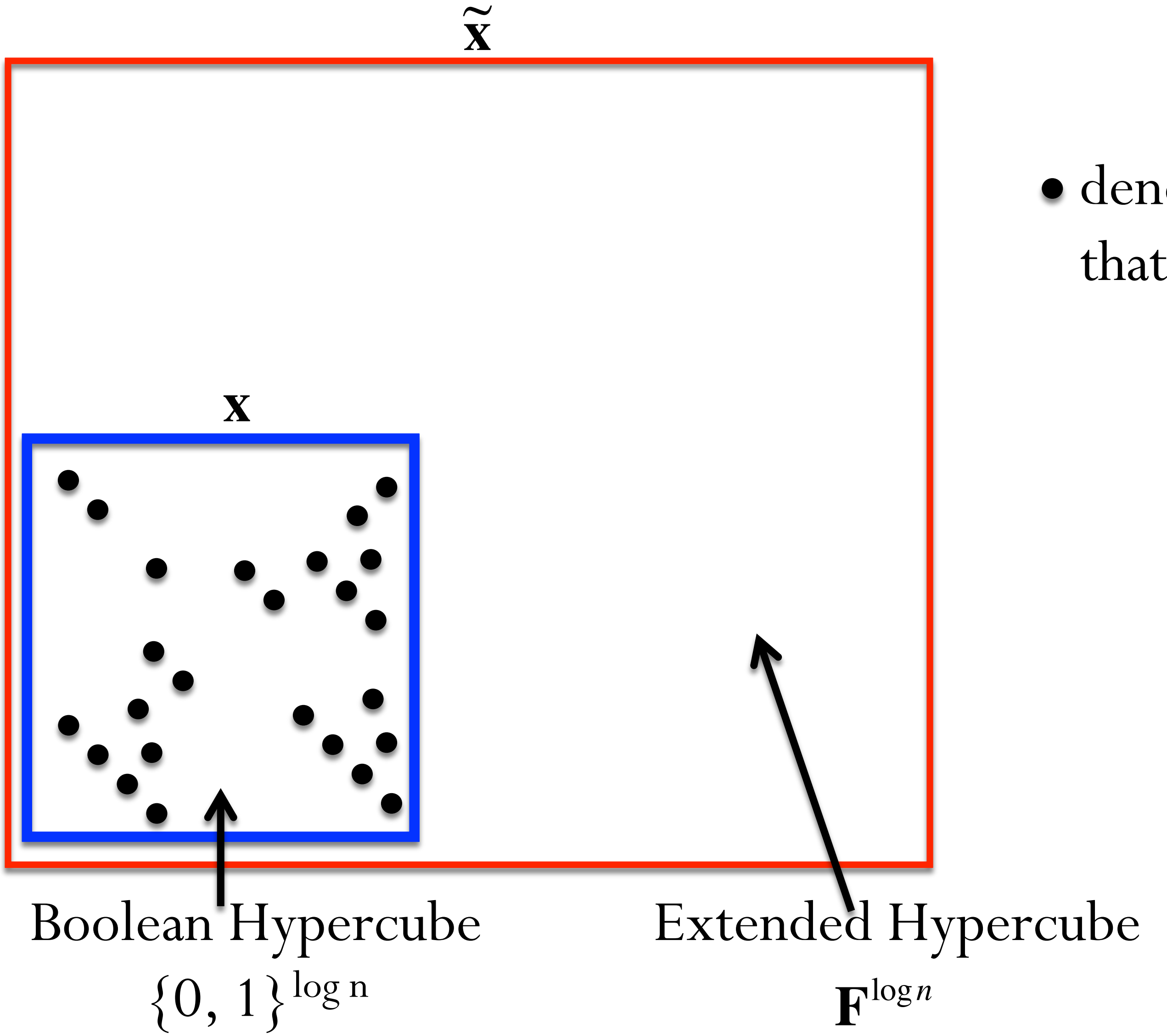
- Note: $\tilde{\mathbf{x}} = \sum_{\mathbf{j} \in \{0,1\}^{\log n}} \mathbf{x}_{\mathbf{j}} \tilde{\delta}_{\mathbf{j}}$ as formal polynomials, where $\tilde{\delta}_{\mathbf{j}}$

  is the multilinear extension of $\delta_{\mathbf{j}}$.

- So $\tilde{\mathbf{x}}(\mathbf{r}) = \sum_{\mathbf{j} \in \{0,1\}^{\log n}} \mathbf{x}_{\mathbf{j}} \tilde{\delta}_{\mathbf{j}}(\mathbf{r})$.

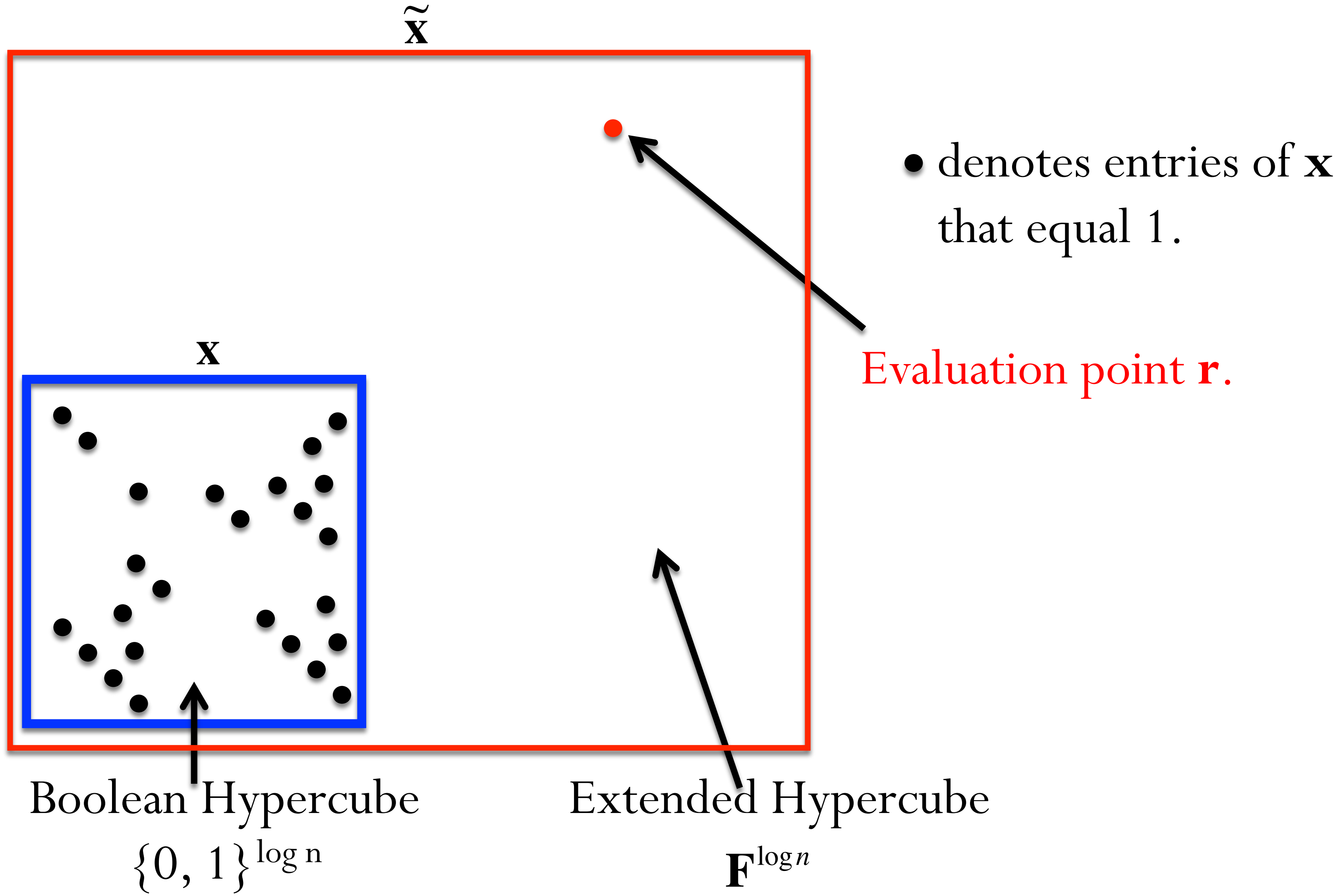
- i.e., each entry $\mathbf{j}$ of $\mathbf{x}$ contributes **independently** to $\tilde{\mathbf{x}}(\mathbf{r})$ (V can just keep a running sum while observing stream).

The INDEX Protocol, Part 2

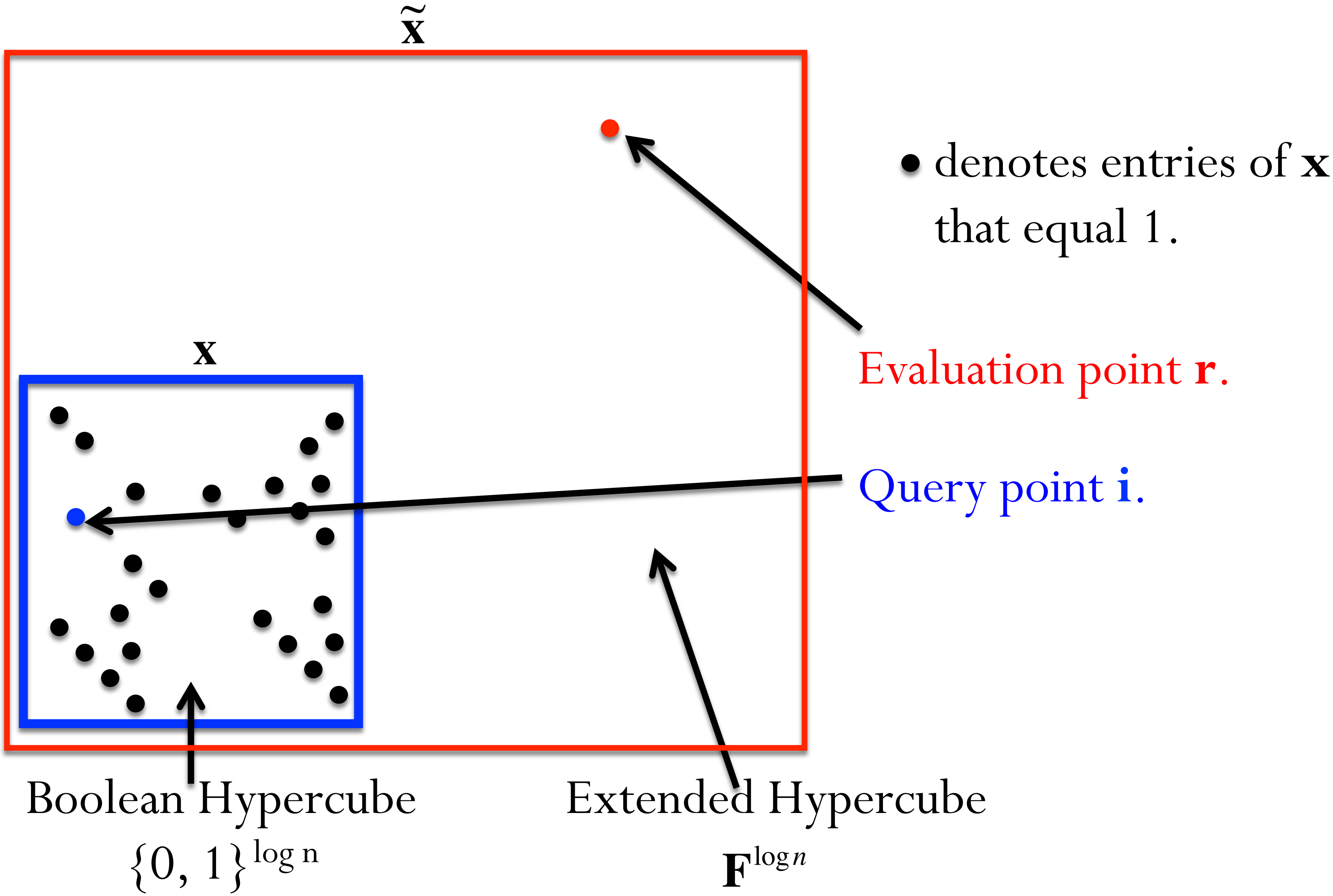


- denotes entries of $\mathbf{x}$ that equal 1.

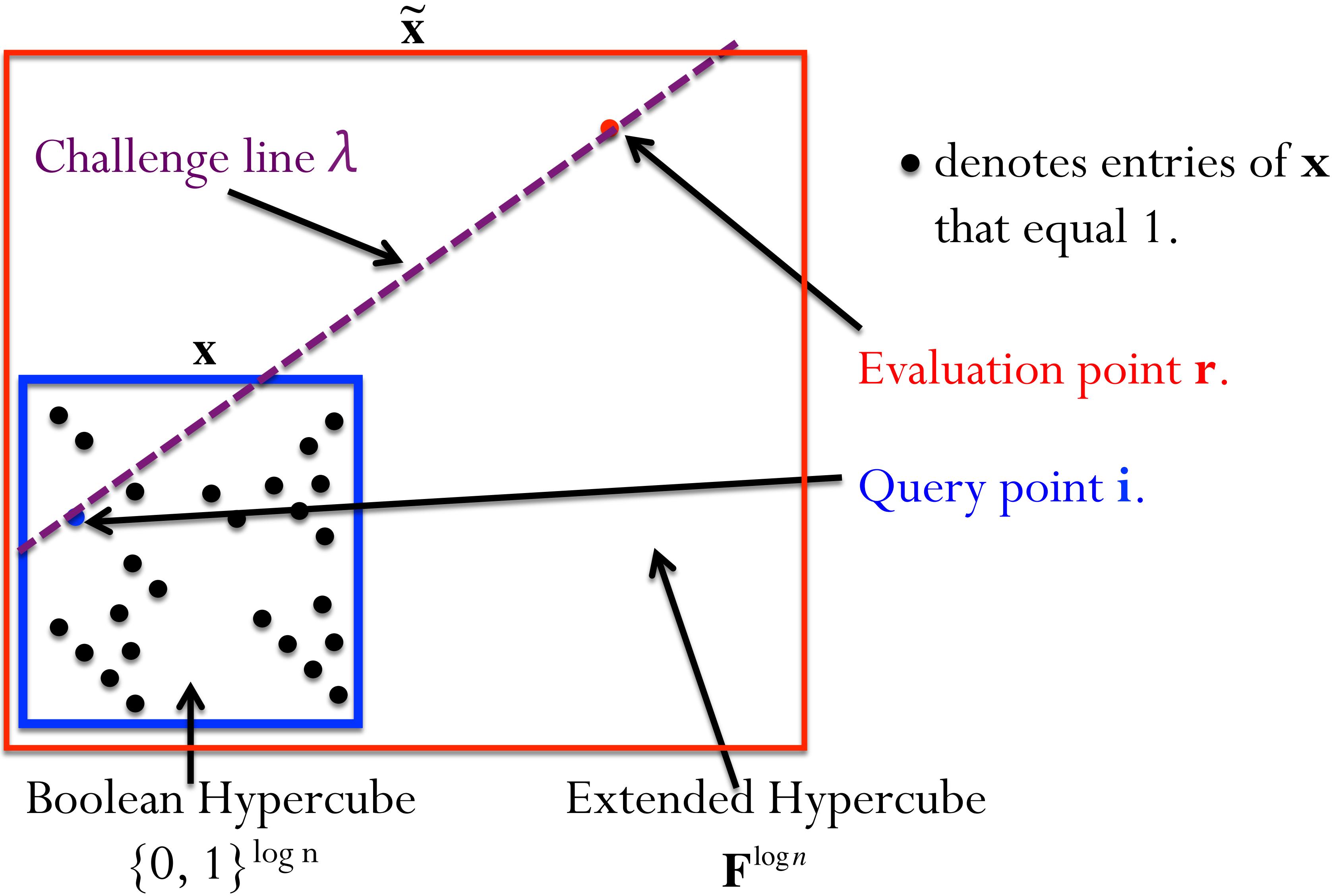
The INDEX Protocol, Part 2



The INDEX Protocol, Part 2



The INDEX Protocol, Part 2



The INDEX Protocol, Part 2

- Let λ denote the unique line through both $\mathbf{i}$ and $\mathbf{r}$.
- Upon learning $\mathbf{i}$, V sends λ to P.
- P responds with a **univariate** polynomial $G(t)$ of degree at most $\log n$ claimed to equal $\tilde{\mathbf{x}}(\lambda(t))$.
- Let t^* be such that $\lambda(t^*) = \mathbf{r}$.
- V rejects if $G(t^*) \neq \tilde{\mathbf{x}}(\mathbf{r})$, and accepts otherwise.
- Total communication is $O(\log n \log |\mathbf{F}|)$ bits.

Completeness

- If P actually sends $G(t) = \tilde{\mathbf{x}}(\lambda(t))$ then V's check will pass, since in this case $G(t^*) = \tilde{\mathbf{x}}(\mathbf{r})$.

Soundness

- If P actually sends $G(t) \neq \tilde{\mathbf{x}}(\lambda(t))$ then $G(t)$ and $\tilde{\mathbf{x}}(\lambda(t))$ can only agree at $\log n / |\mathbf{F}|$ points.
- From P's perspective, once he receives the message $\lambda(t)$, $\mathbf{r}$ is uniformly distributed in $\text{Range}(\lambda) \setminus \{i\}$. So the probability that $G(t^*) = \tilde{\mathbf{x}}(\lambda(t^*)) = \mathbf{x}(\mathbf{r})$ is at most $\log n / (|\mathbf{F}| - 1) < 1/3$.

Extensions of the INDEX Protocol

Polylogarithmic Cost Protocols

- We give polylogarithmic cost protocols for the following problems.
  - Nearest Neighbor Search under many standard metrics (L_1, L_2, L_∞, etc.)
  - Median and Quantiles.
  - RangeCount Queries.
  - PatternMatching (with wildcards).

Overview of RangeCount Protocol

- RangeCount Problem: Fix a data universe [n] and a range space $\mathbf{R} \subseteq 2^{[n]}$. The input is list of points $\{x_1, \ldots, x_m\}$ from [n], followed by a range $R^* \in \mathbf{R}$. Goal is to output $|\{i: x_i \in R^*\}|$.

Overview of RangeCount Protocol

- RangeCount Problem: Fix a data universe [n] and a range space $\mathbf{R} \subseteq 2^{[n]}$. The input is list of points $\{x_1, \ldots, x_m\}$ from [n], followed by a range $R^* \in \mathbf{R}$. Goal is to output $|\{i: x_i \in R^*\}|$.
- Basic idea: Reduce to the (Generalized) INDEX problem.
  - Create a "derived stream" consisting entirely of **ranges**.
  - On stream update x_i, insert a copy of **every** range R that x_i is in.
  - V needs to know the frequency of R^* in derived stream. Can answer this with the (Generalized) INDEX protocol.
  - Space and communication costs are only $O(\log|\mathbf{R}|\log\log|\mathbf{R}|)$.
  - Problem: V requires $|\mathbf{R}|$ time per stream update!

Online Interactive Proofs (Communication Model)

So How Powerful Are O(1)-Round SIPs?

- INDEX has a two-message protocol of logarithmic cost.
- Does a similar protocol exist for "harder" problems such as DISJOINTNESS?

So How Powerful Are O(1)-Round SIPs?

- INDEX has a two-message protocol of logarithmic cost.
- Does a similar protocol exist for "harder" problems such as DISJOINTNESS?
- To investigate, we introduce two hierarchies of communication models called OIP_+ and OIP.
- $OIP_+[k]$ can simulate **all** k-message SIPs. So lower bounds against OIP_+ protocols imply ones against SIPs.
- OIP[k] is weaker, but can still simulate all **known** SIPs, and captures the fundamental way SIPs differ from IPs.

AM^{cc} [BFS86]

Merlin

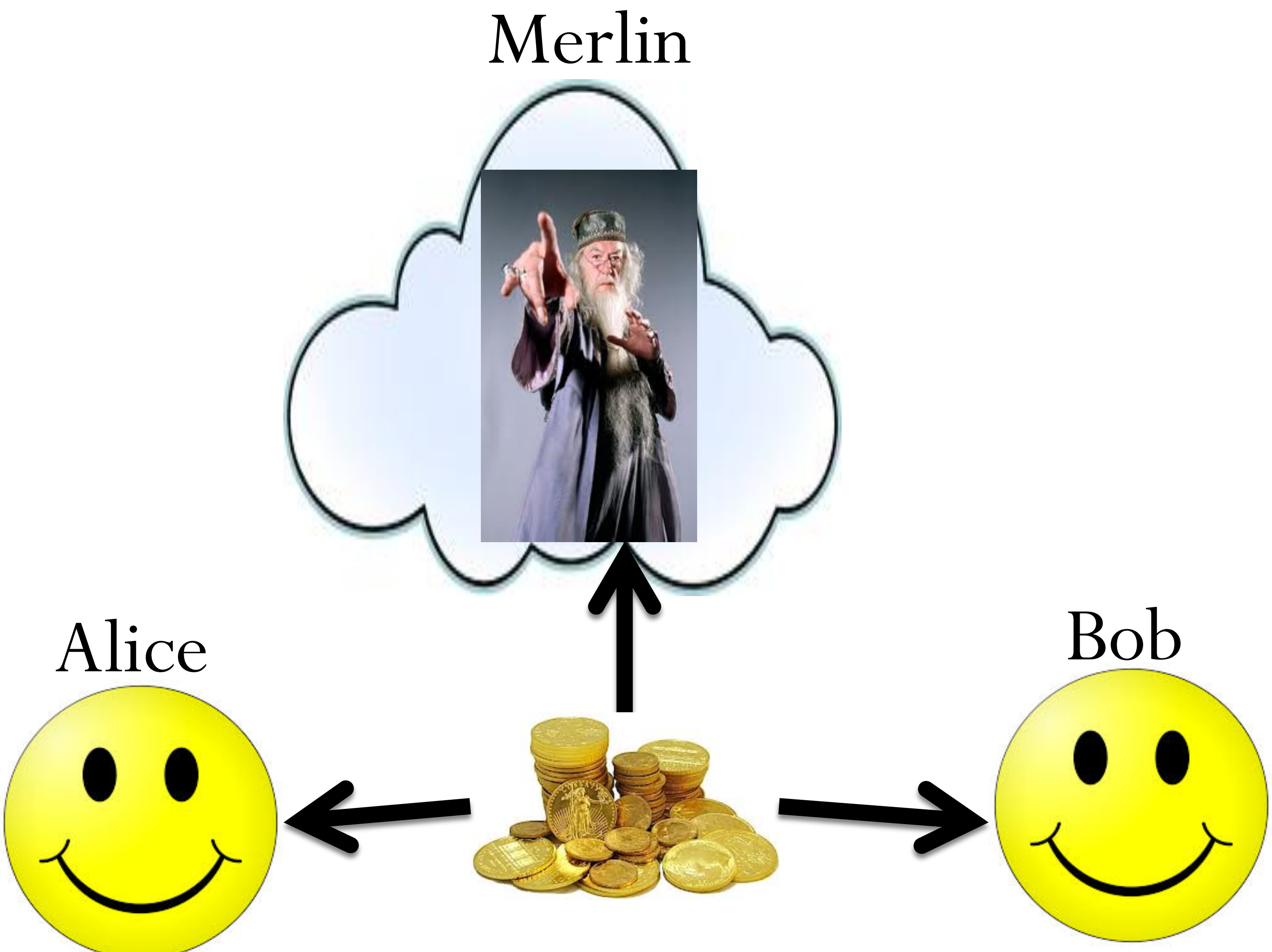
Alice

Bob

Goal: Compute f(x,y)

x

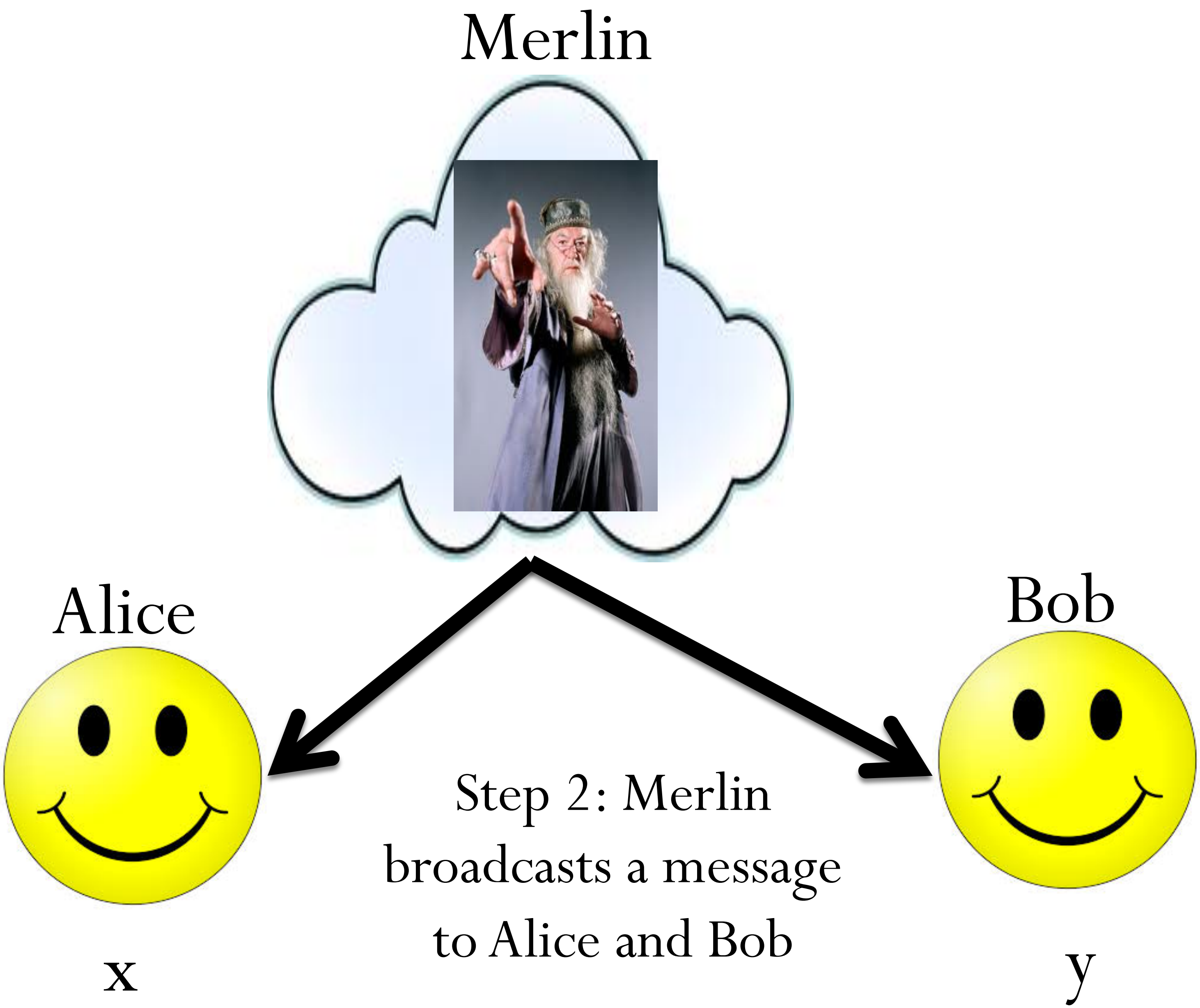
y

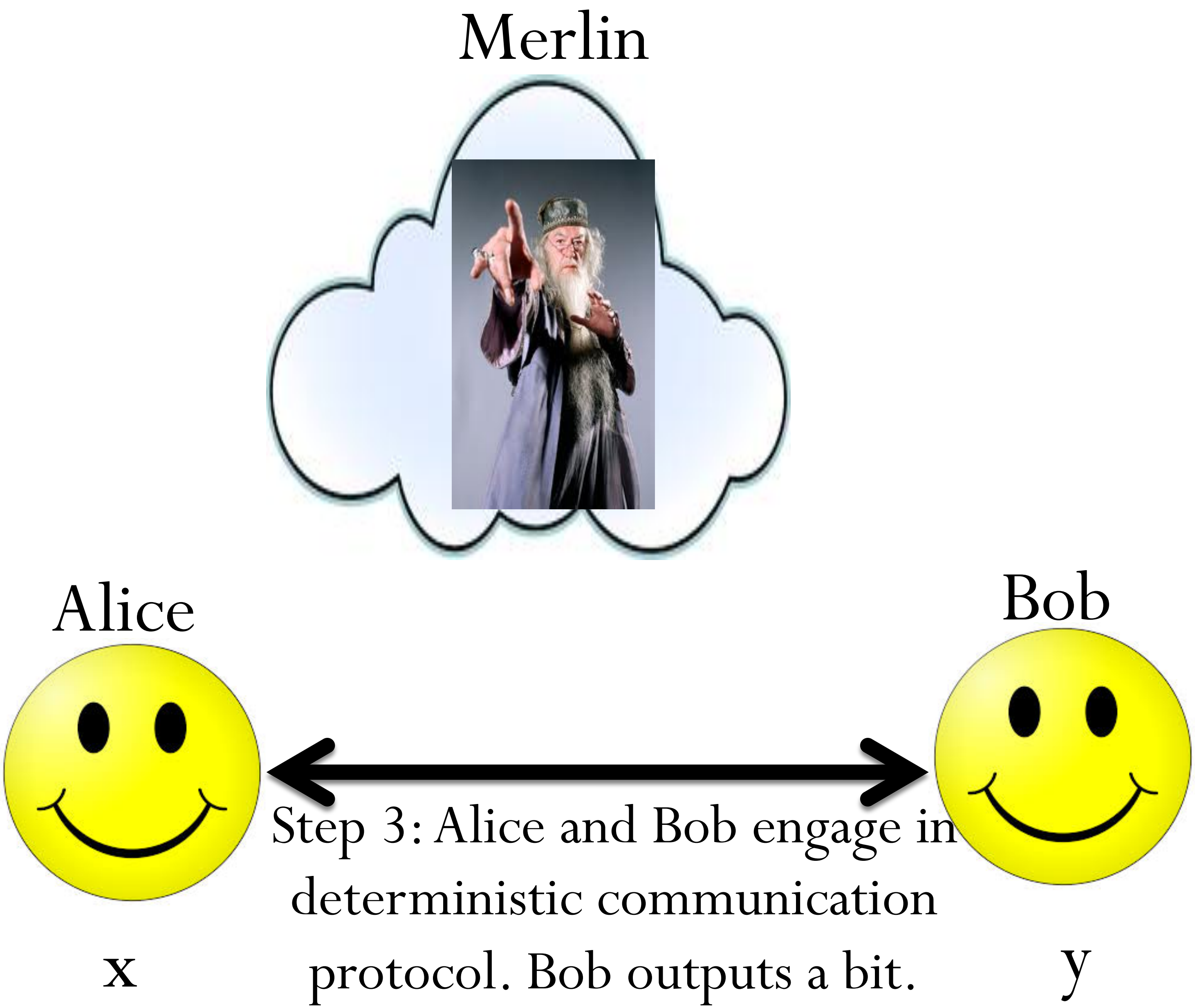


x

Step 1: Random coins are broadcasted.

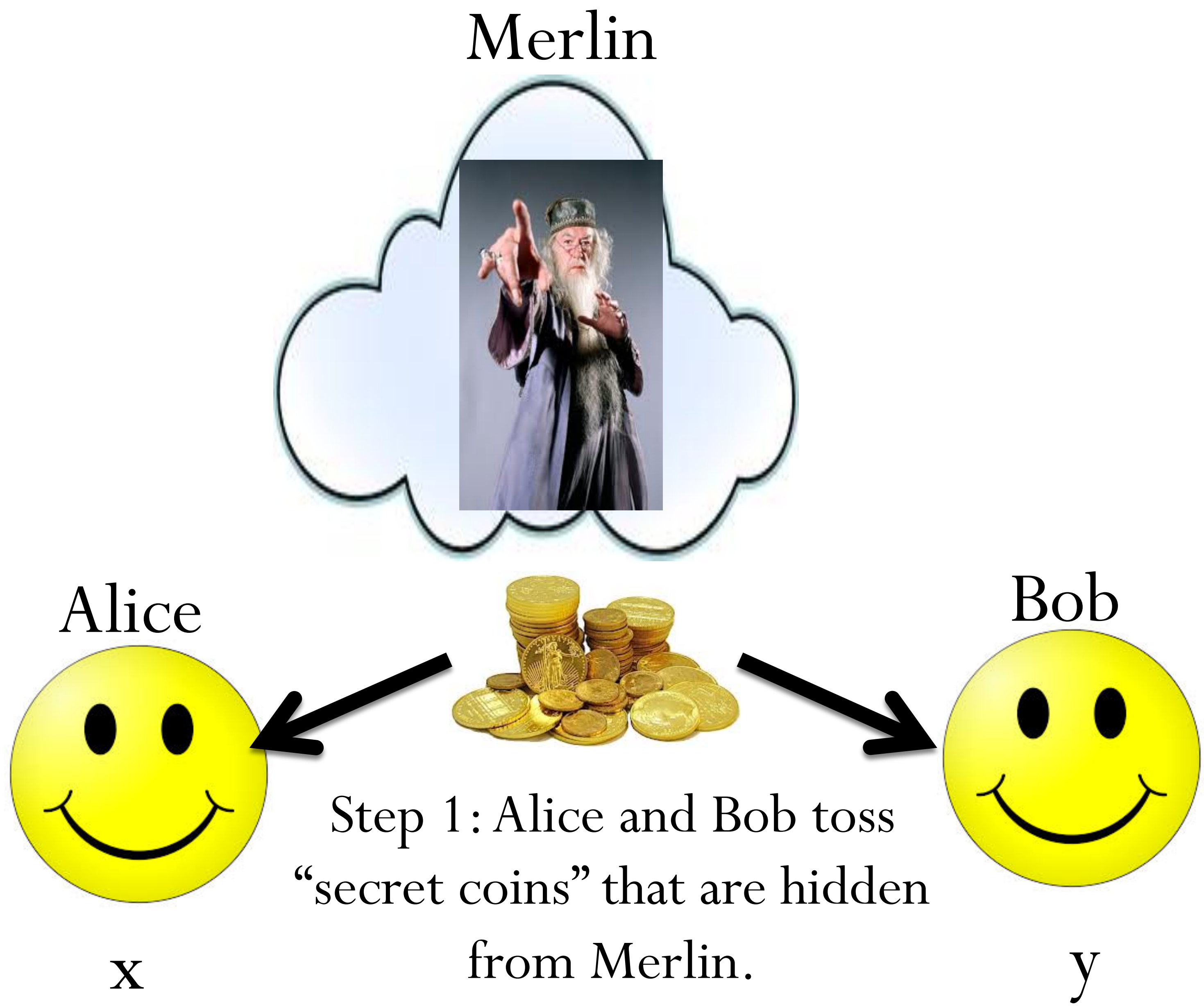
y

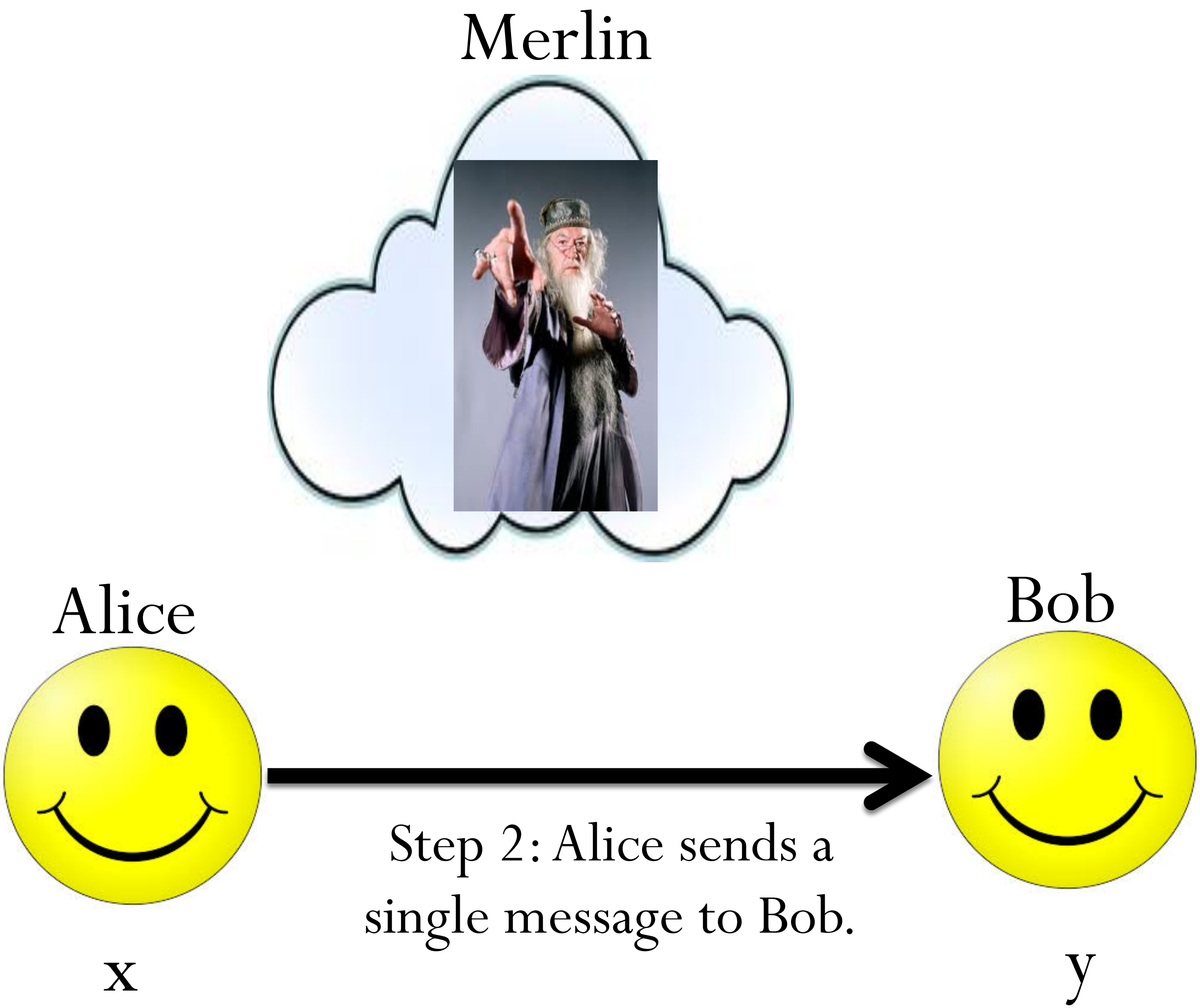

Merlin
Alice
Bob
Step 2: Merlin broadcasts a message to Alice and Bob
x
y

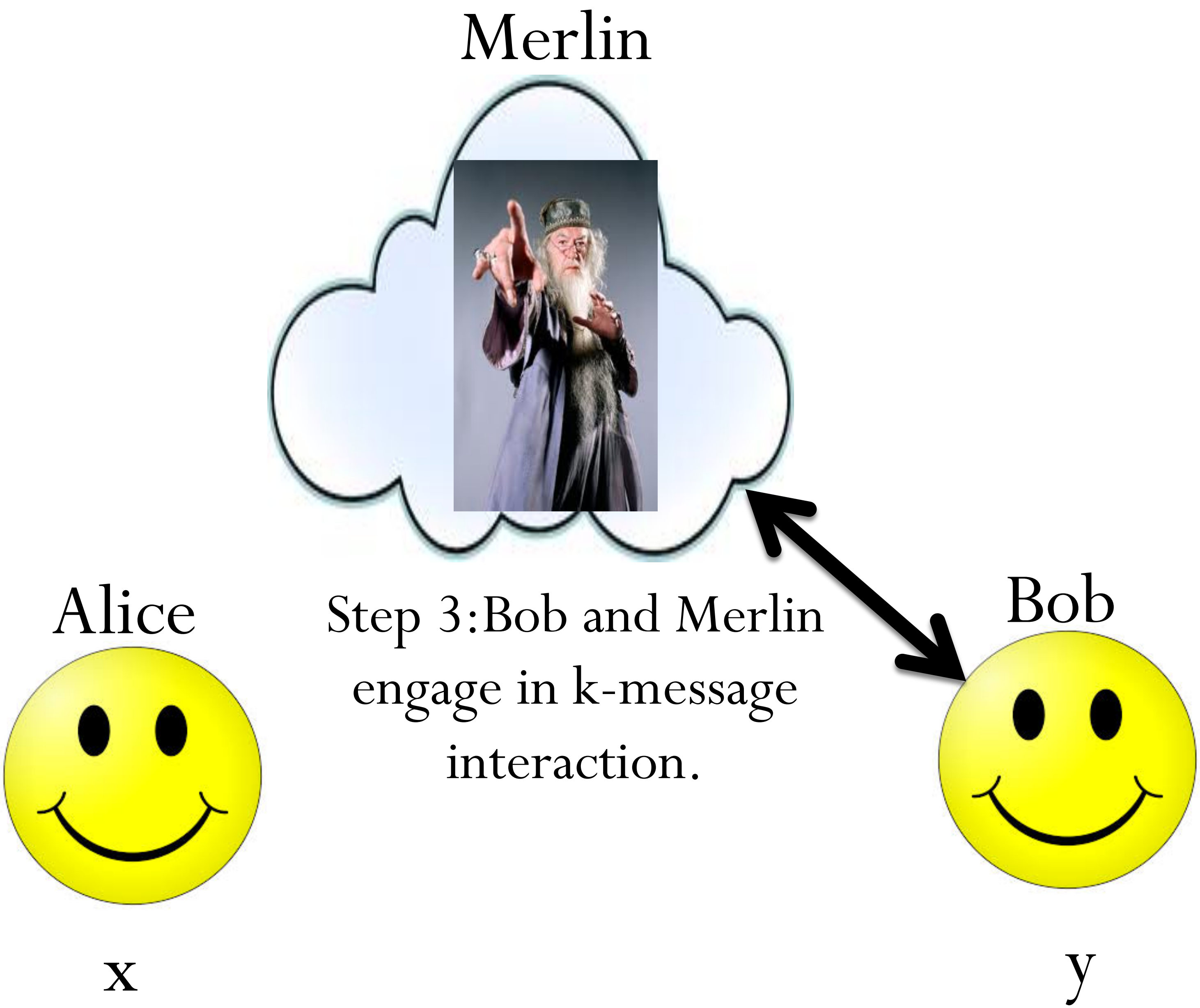


Step 3: Alice and Bob engage in deterministic communication protocol. Bob outputs a bit.

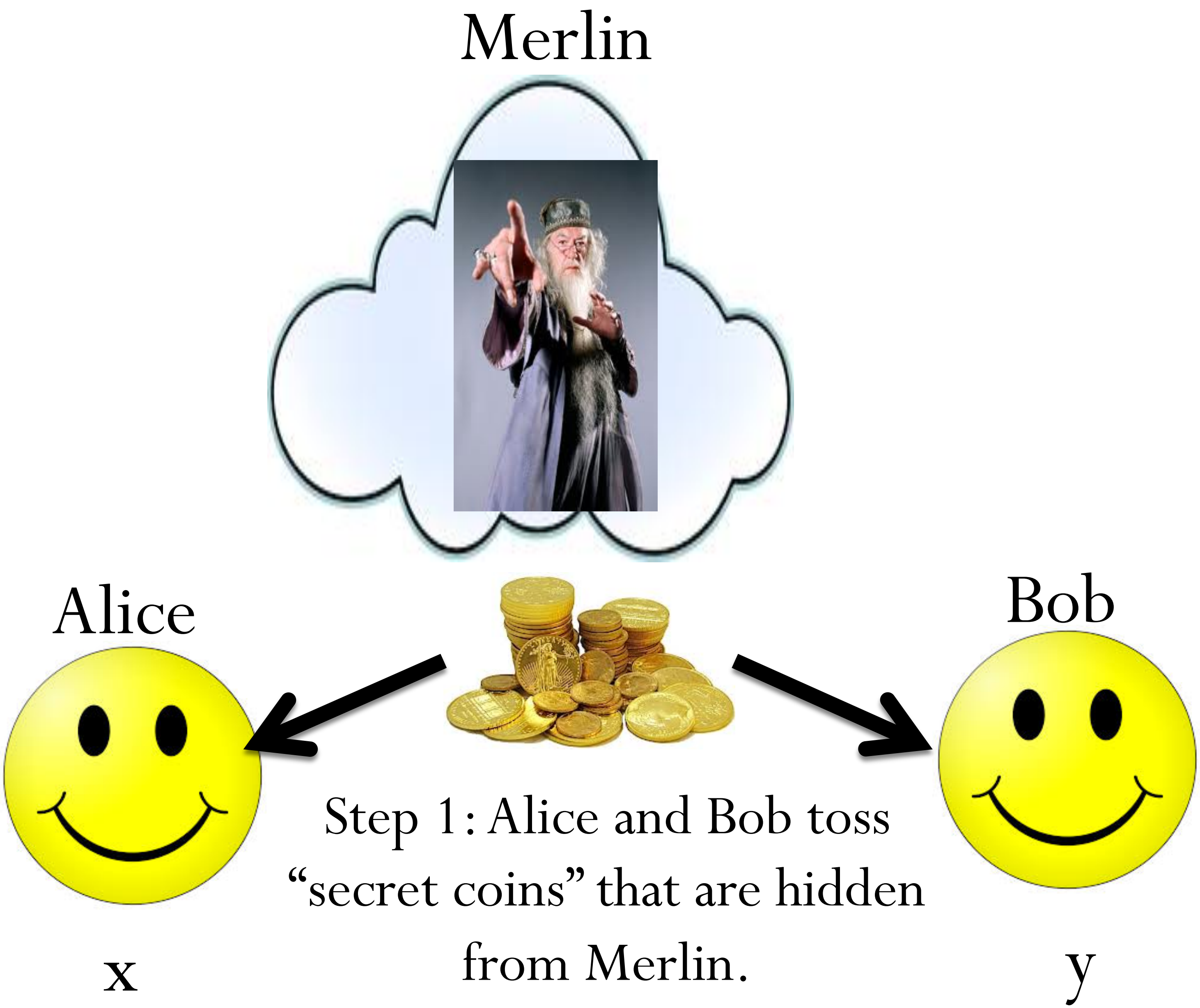
$OIP_+[k]$

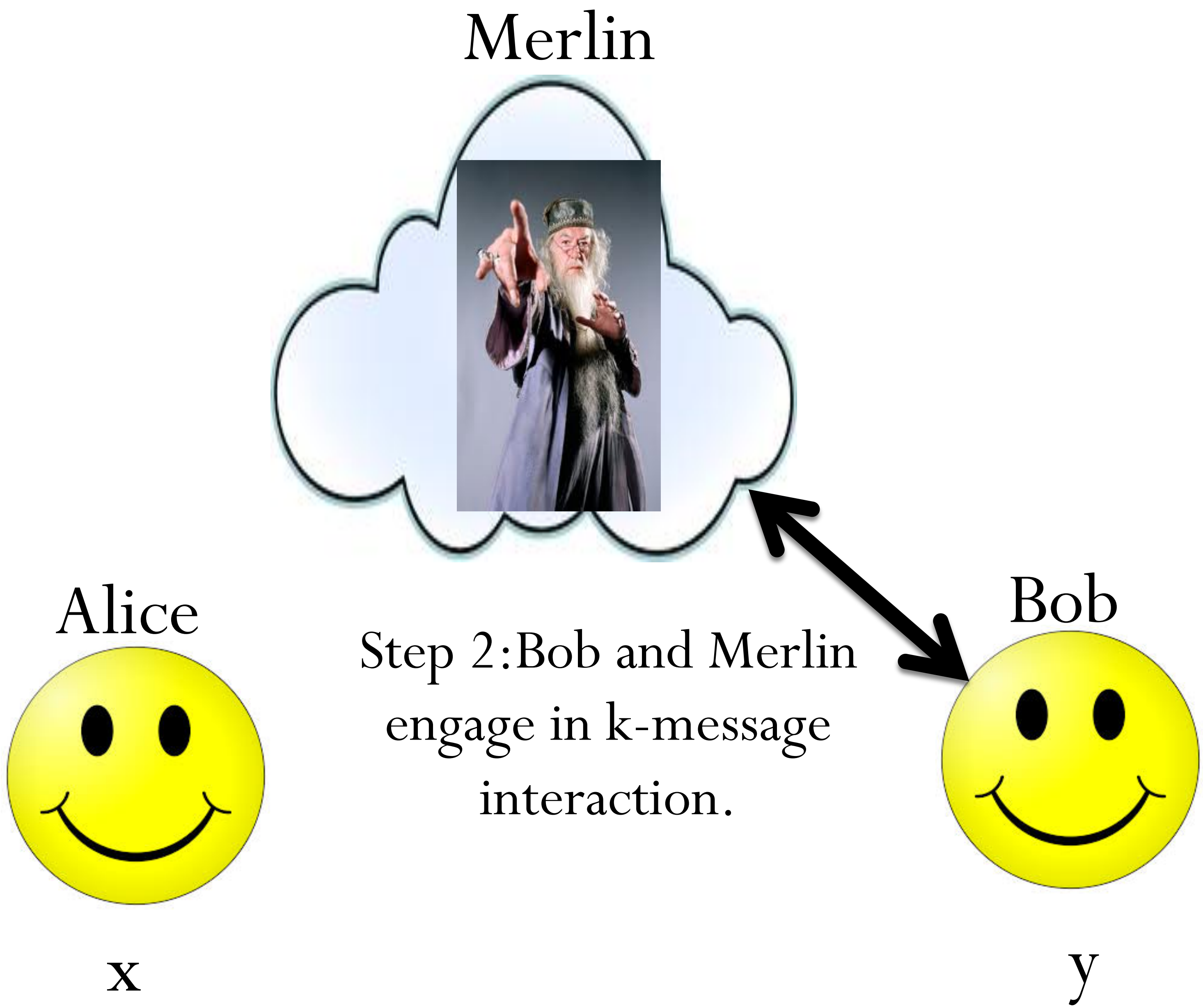

Merlin
Alice
Bob
x
y
Step 1: Alice and Bob toss "secret coins" that are hidden from Merlin.

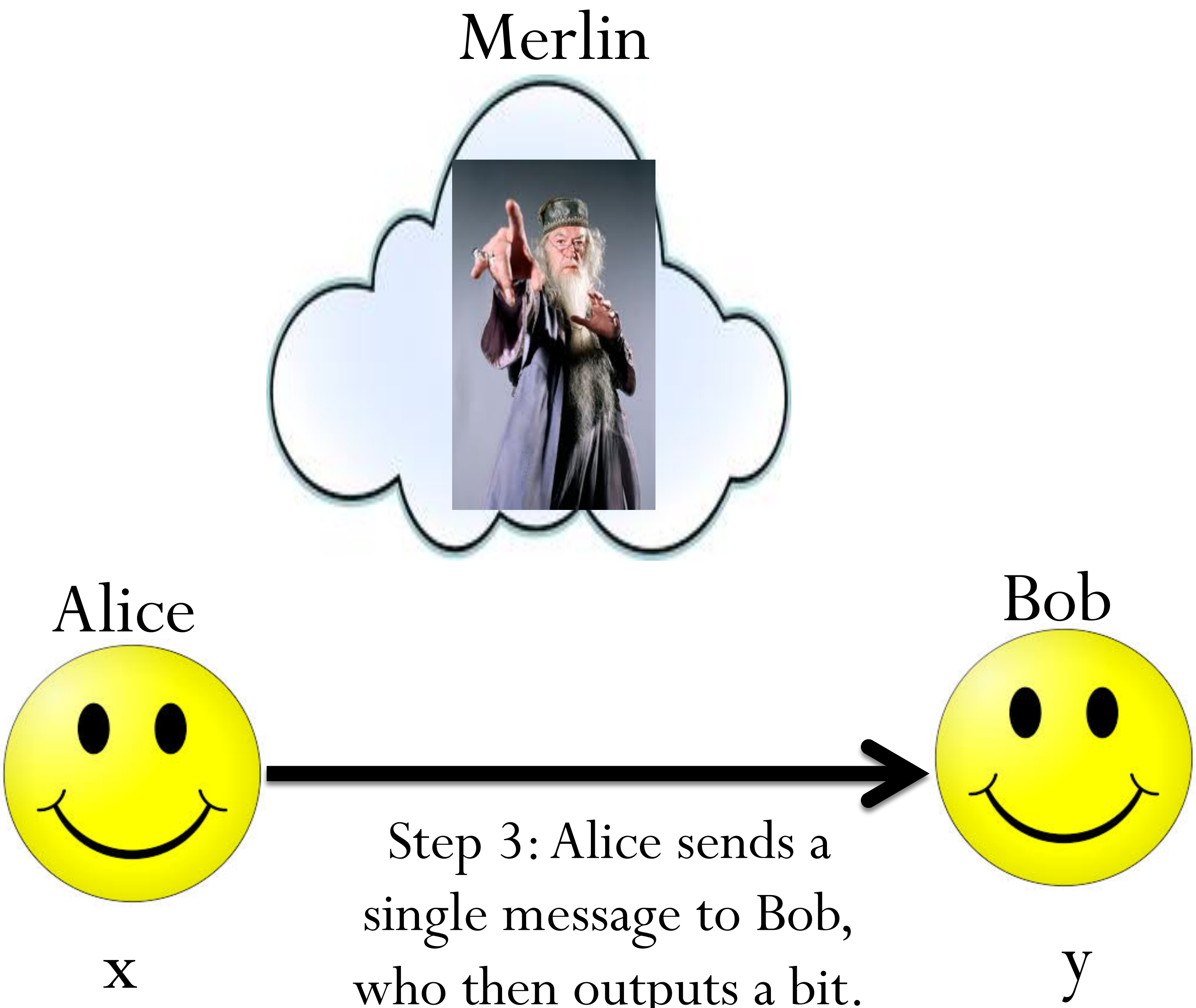

Merlin
Alice
Bob
Step 2: Alice sends a single message to Bob.
x
y


Merlin
Alice
Step 3:Bob and Merlin engage in k-message interaction.
Bob
x
y

OIP[k]


Merlin
Alice
Bob
Step 1: Alice and Bob toss "secret coins" that are hidden from Merlin.
x
y


Merlin
Alice
Bob
Step 2:Bob and Merlin engage in k-message interaction.
x
y



Step 3: Alice sends a single message to Bob, who then outputs a bit.

OIP[k] Can Simulate All Known k-message SIPs

OIP[2] protocol of cost O(log n loglog n) for INDEX.

Merlin

Alice

Goal: Output $\mathbf{x_i}$.

Bob

$\mathbf{x}$

$\mathbf{i}$

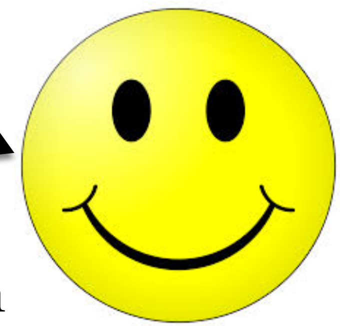
Merlin

Alice

x

Step 1: Alice and Bob toss "secret coins" that are hidden from Merlin to choose evaluation point **r**.

Bob



i

Merlin

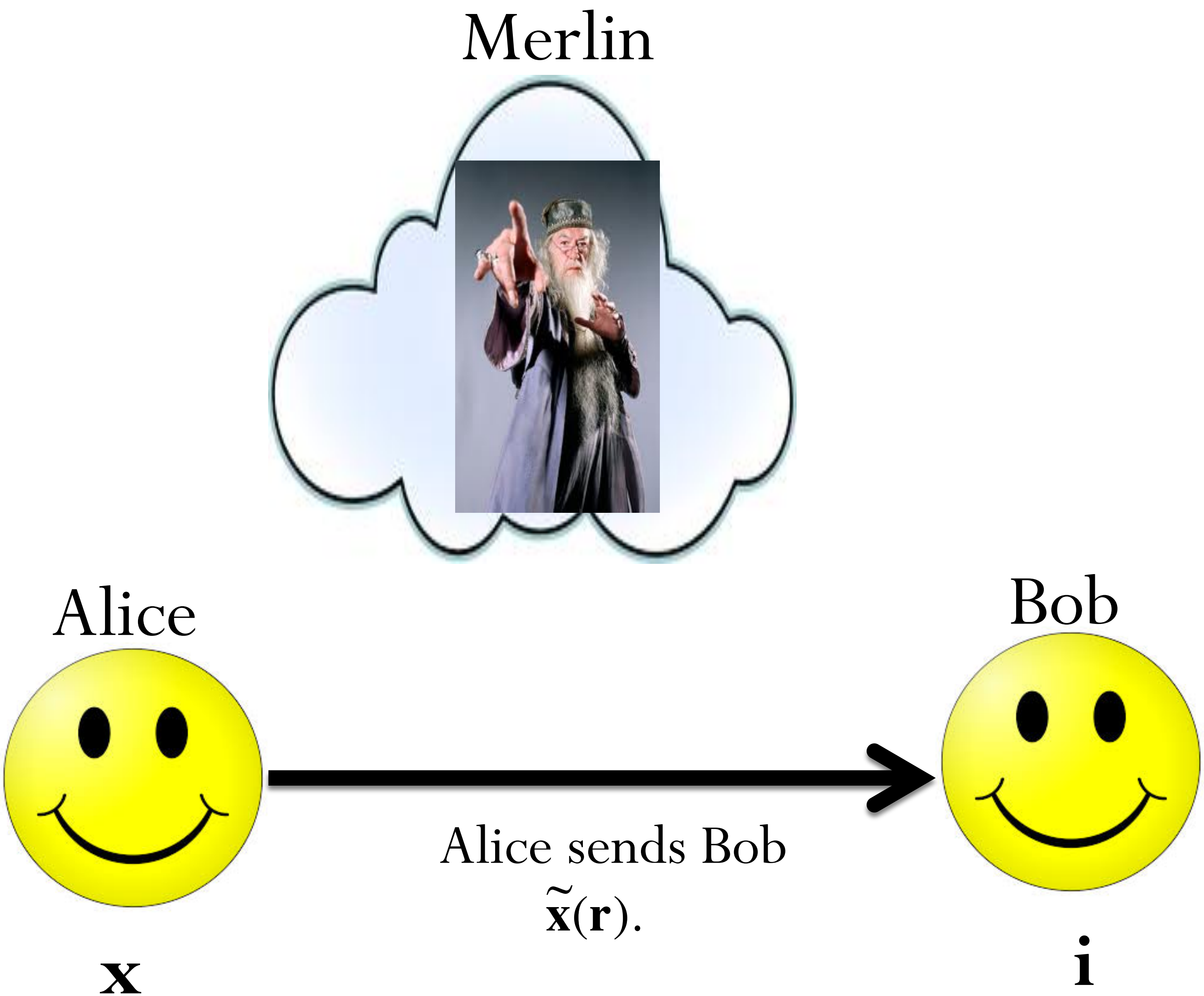
Alice

$\mathbf{x}$

Bob

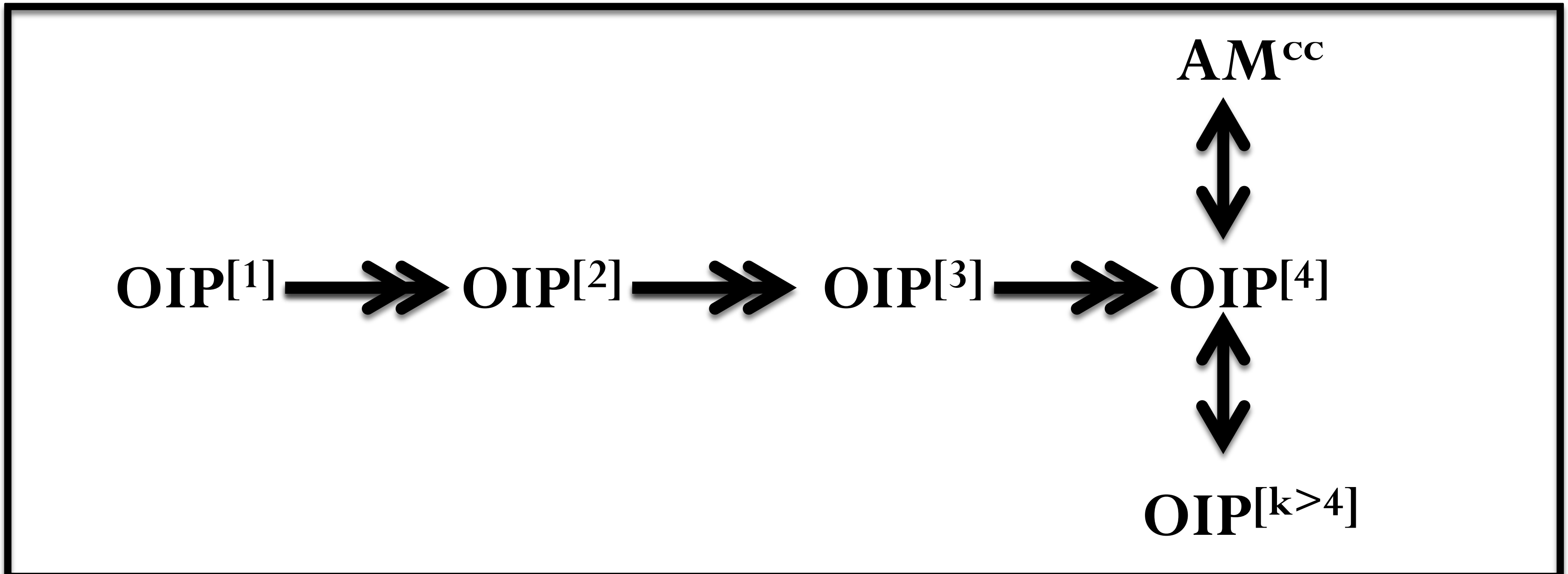
$\mathbf{i}$

Bob sends Merlin λ , the line through **r** and **i.** Merlin responds to univariate polynomial G(t) claimed to equal $\tilde{\mathbf{x}}(\lambda(t))$.



Merlin
Alice
Bob
Alice sends Bob $\tilde{\mathbf{x}}(\mathbf{r})$.
$\mathbf{x}$
$\mathbf{i}$

A Communication Complexity Zoo



Notation:

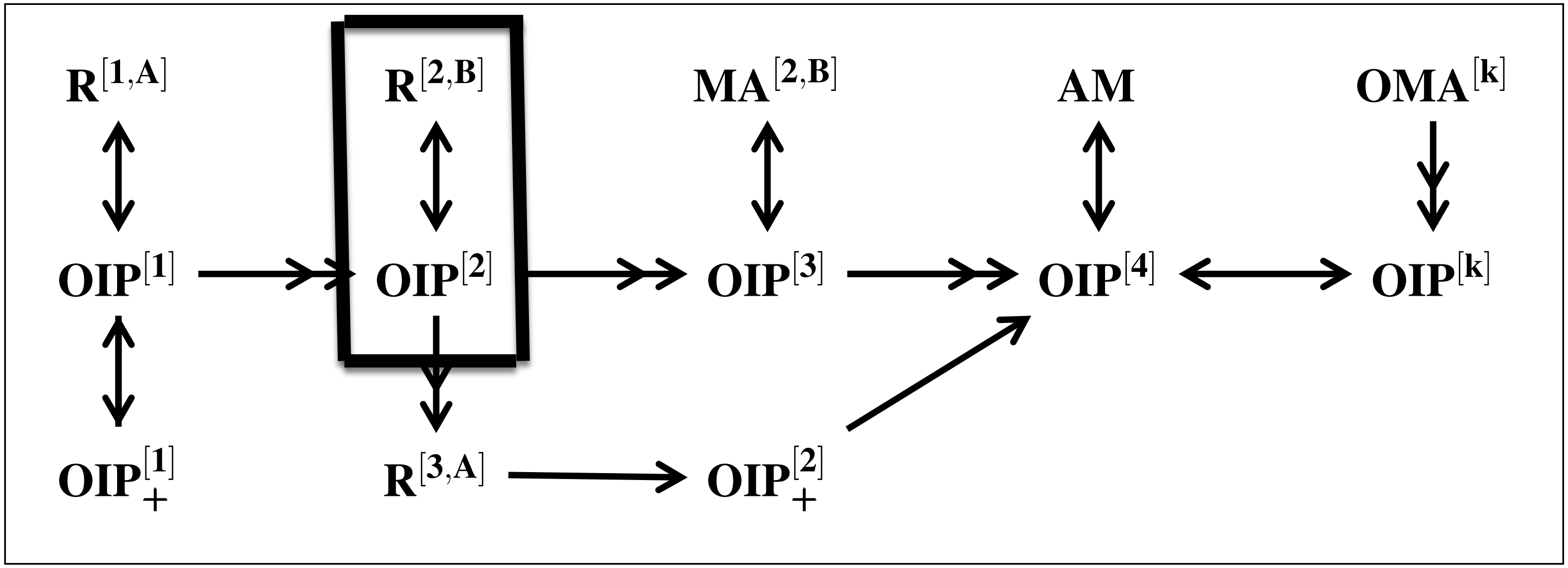
- $\mathbf{OIP^{[k]}}$ denotes class of functions solved by polylog cost OIP[k] protocols, $\mathbf{AM^{cc}}$ functions solved by polylog cost AM^{cc} protocols.
- ⇉ denotes containment with exponential separation.
- ⟷ denotes equality.

Main Findings

- Any OIP[2] or OIP[3] protocol for DISJOINTNESS has cost $\Omega(n^{1/2})$ and $\Omega(n^{1/3})$ respectively. Both bounds are tight.
  - i.e. There is no three-message SIP of polylog cost for DISJOINTNESS using "known techniques".
- OIP[4] is equivalent to AM^{cc}, a communication class beyond the reach of current lower bound methods.
  - i.e. Proving lower bounds on 4-message SIPs may be challenging.
- Generic round-reduction impossible in the OIP hierarchy.
  - In contrast, **AM[O(1)]=AM[2]** in classical interactive proofs.

Thank you!

A Communication Complexity Zoo



Notation:

- $R^{[k, A]}$ is class of functions solved by (standard) randomized k-message protocols of polylog cost, where Alice speaks first.
- $OMA^{[k]}$, $OIP^{[k]}$, and $OIP_+^{[k]}$ are classes of functions solved by polylog cost protocols at k'th level of OMA, OIP, and OIP_+ hierarchies.

Main Findings

- Any OIP[2] or OIP[3] protocol for DISJOINTNESS has cost
  - i.e. There is no three-message SIP of polylog cost for DISJOINTNESS using "known techniques".
- OIP[4] is equivalent to AM^{cc}, a communication class beyond the reach of current lower bound methods.
  - i.e. Proving lower bounds on 4-message SIPs may be challenging.
- Generic round-reduction impossible in the OIP hierarchy.
  - In contrast, **AM[O(1)]=AM[2]** in classical interactive proofs.

$R^{[2, B]} \subseteq OIP^{[2]}$

- Suppose we are given a 2-message randomized communication protocol Q_1 of cost C, with Bob speaking first.
- We give an OIP[2] protocol Q_2 that simulates Q_1 with a quadratic blowup in cost.

$R^{[2, B]} \subseteq OIP^{[2]}$

- Suppose we are given a 2-message randomized communication protocol Q_1 of cost C, with Bob speaking first.
- We give an OIP[2] protocol Q_2 that simulates Q_1 with a quadratic blowup in cost.
- Alice(Q_2) and Bob(Q_2) first toss the coins they would use in Q_1.
- Alice(Q_2) then runs our (Generalized) INDEX protocol on the 2^C-dimensional vector x whose ith entry is what Alice(Q_1)'s response to Bob(Q_1) would be if Bob(Q_1)'s message to Alice(Q_1) was i.
- Our INDEX protocol must be run over field $\mathbf{F}$ of size $\sim 2^C$, so total cost is $O(C \log|\mathbf{F}|) = O(C^2)$.

$OIP^{[2]} \subseteq R^{[2, B]}$: Part 1

- Suppose we are given an OIP[2] protocol Q_2 of cost C.
- We give an $R^{[2, B]}$ protocol Q_1 that simulates Q_2 with a quadratic blowup in cost.

$OIP^{[2]} \subseteq R^{[2, B]}$: Part 1

- Suppose we are given an OIP[2] protocol Q_2 of cost C.
- We give an $R^{[2, B]}$ protocol Q_1 that simulates Q_2 with a quadratic blowup in cost.
- Alice(Q_1) and Bob(Q_1) first toss the "secret coins" they would use in Q_2. Say these coins come from a distribution D.
- Bob(Q_1) can then determine the message m_B that Bob(Q_2) would send to Merlin.
- Let D_{m_B} denote D conditioned on the event that Bob(Q_2)'s message to Merlin is m_B.
- Alice(Q_1) and Bob(Q_1)'s goal then becomes:

  Determine whether there exists a message m_M that Merlin could send in Q_2 that would cause Alice(Q_2) and Bob(Q_2) to output 1 with high probability if their secret coins come from distribution D_{m_B}.

$OIP^{[2]} \subseteq R^{[2, B]}$: Part 2

- Bob(Q_1) takes h=O(C) random samples $\mathbf{r}_1$, … $\mathbf{r}_h$ from the distribution D_{m_B}, and sends them all to Alice.
  - Bob can do this because m_B does not depend on Alice's input!
- For each $\mathbf{r}_i$, Alice(Q_1) tells Bob(Q_1) what message Alice(Q_2) would send given secret randomness $\mathbf{r}_i$.
- Now that Bob(Q_1) knows what Alice(Q_2) would say for all of the $\mathbf{r}_i$'s, he iterates over **all** possible Merlin messages m_M and outputs 1 iff there is some m_M that would cause Bob(Q_2) to accept for a **majority** of the $\mathbf{r}_i$'s.

Thank you!

So how Powerful Are O(1)-Round SIPs?

- INDEX has a two-message protocol of logarithmic cost.
- Does a similar protocol exist for "harder" problems such as DISJOINTNESS?
- Answer: Probably not.

Left-overs

Given a d-variate polynomial g and line **λ,** we let $g|_{\lambda}$ be the **univariate** polynomial $g(h_{\lambda})$, and call this the **restriction of** g **to** λ.

The Polynomial Agreement Protocol

- Suppose a data stream specifies a v-variate polynomial g over field **F,** followed by a point **i** in **F**^v. Goal is to evaluate g(**i**).
- As long as V can evaluate g at a random point **r** in space s, there is a two-message protocol of space cost s and comm. cost O(deg(g)log|**F**|) for this problem.

Median

- Input: a stream of numbers $<x_1, \ldots, x_m>$ from a universe of size n.
- Let $N_j = |\{j : x_j < i\}|$.
- Goal: output a number i such that $N_{i-1} < m/2$ and $N_{i+1} > m/2$.

Reducing Median to Poly. Agreement

- "Treat" stream update xj as an insertion of items xj, xj+1, …, n. This creates a "derived stream" S such that the **frequency** of item j in S is exactly Nj.
- Let **y** be the n-dimensional vector such that yj is the frequency of item j in
- At end of stream, P sends a claimed median i.
- To check that i is a median, it suffices for V to check that, in the derived stream