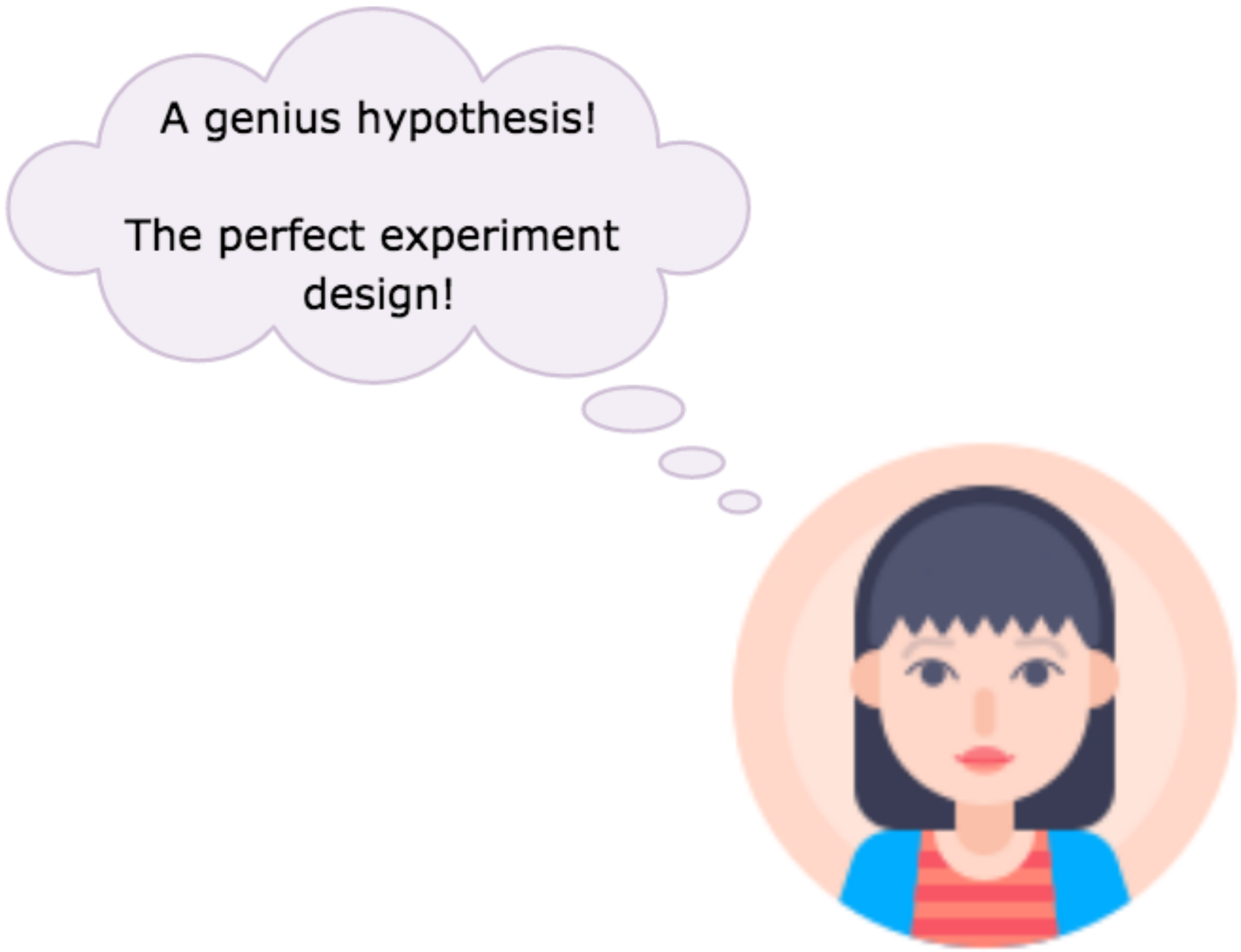
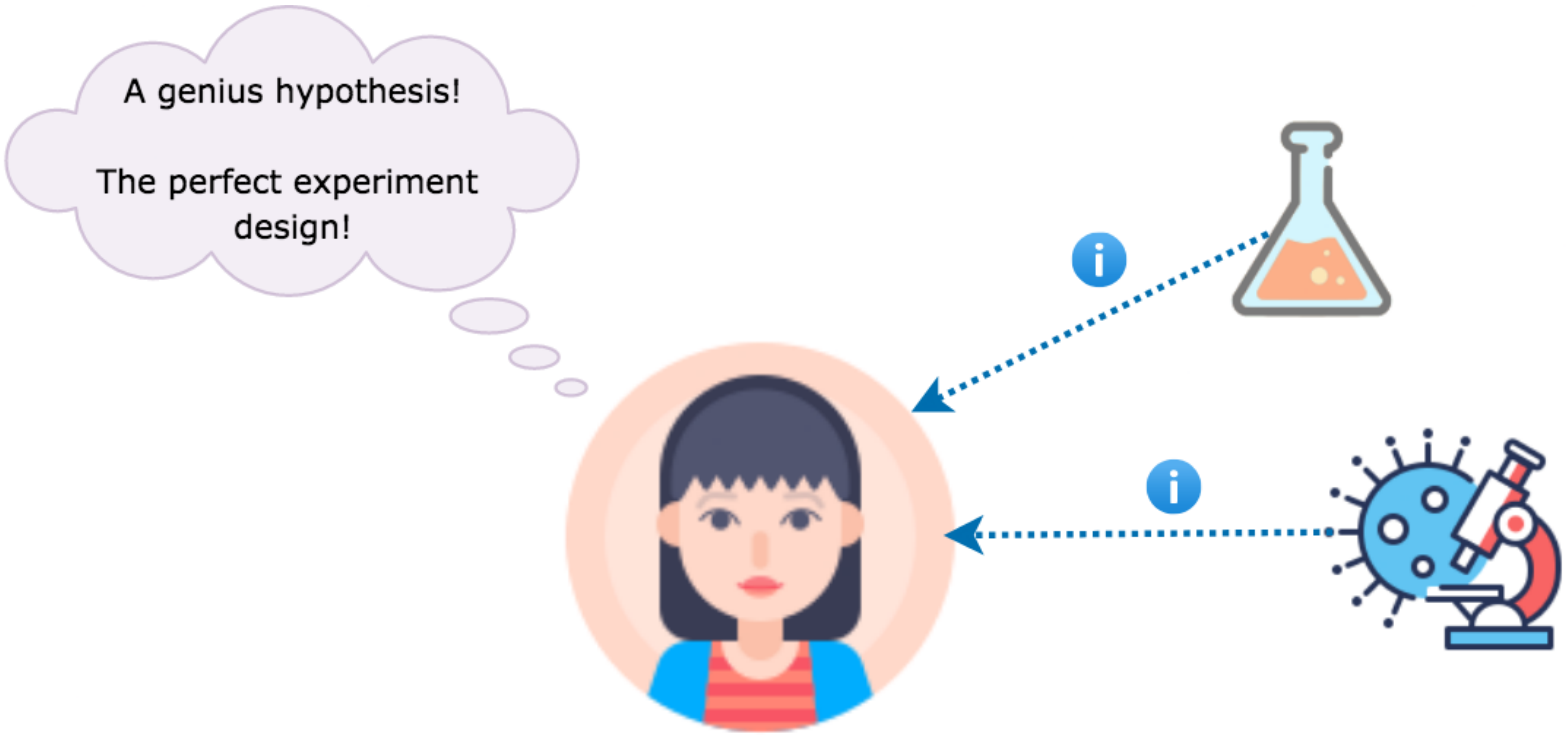
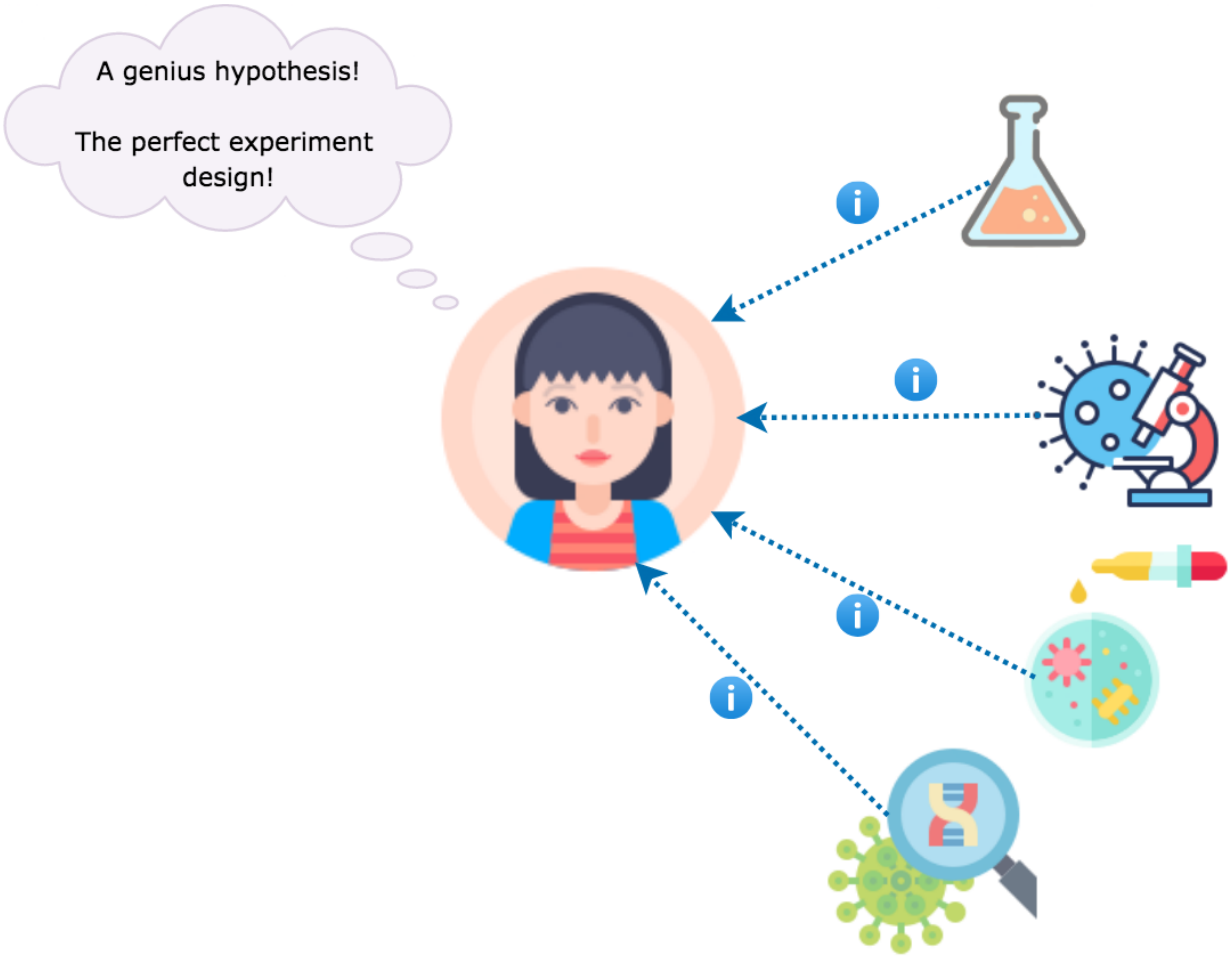
Covert Learning:

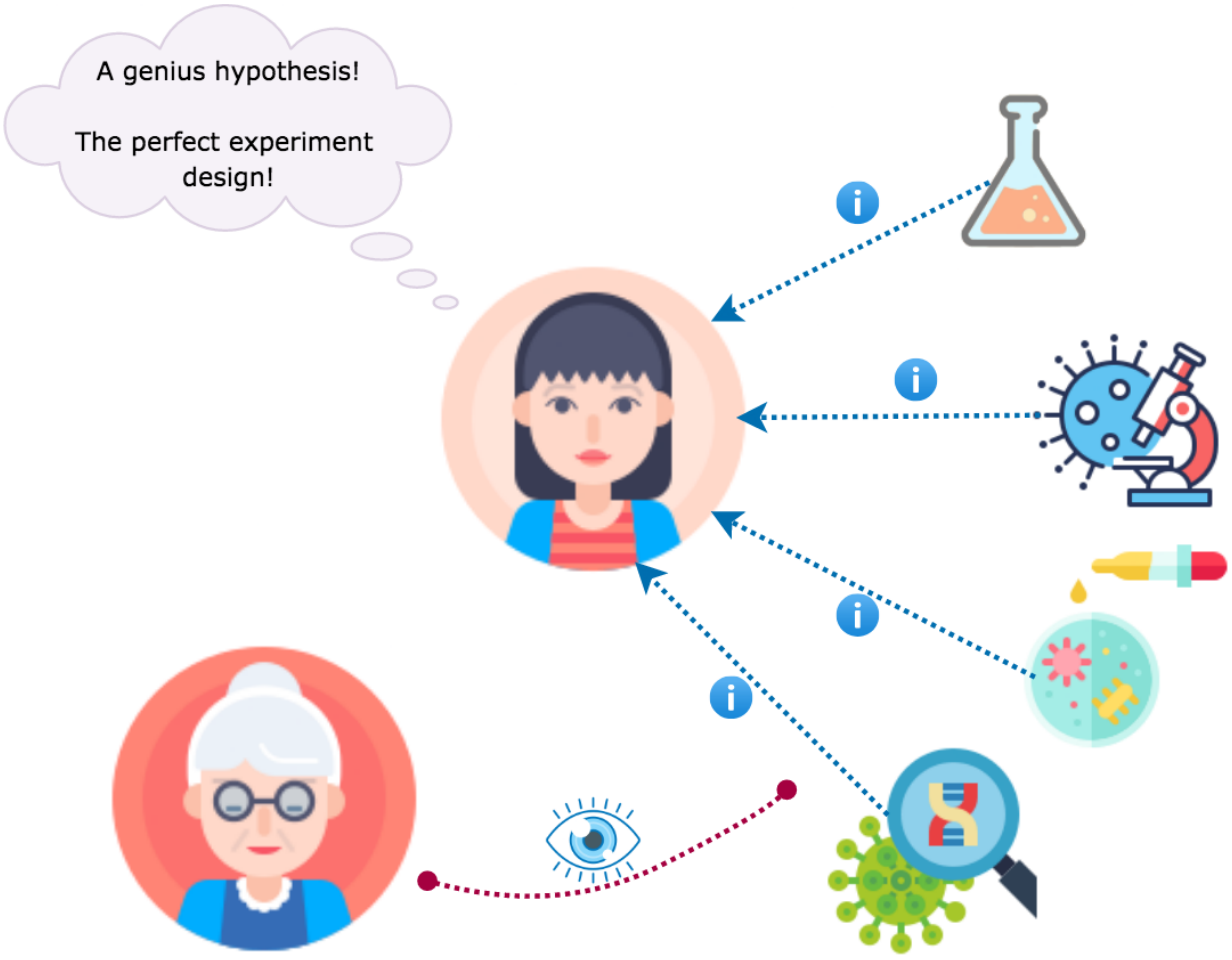
How to learn with an untrusted intermediary

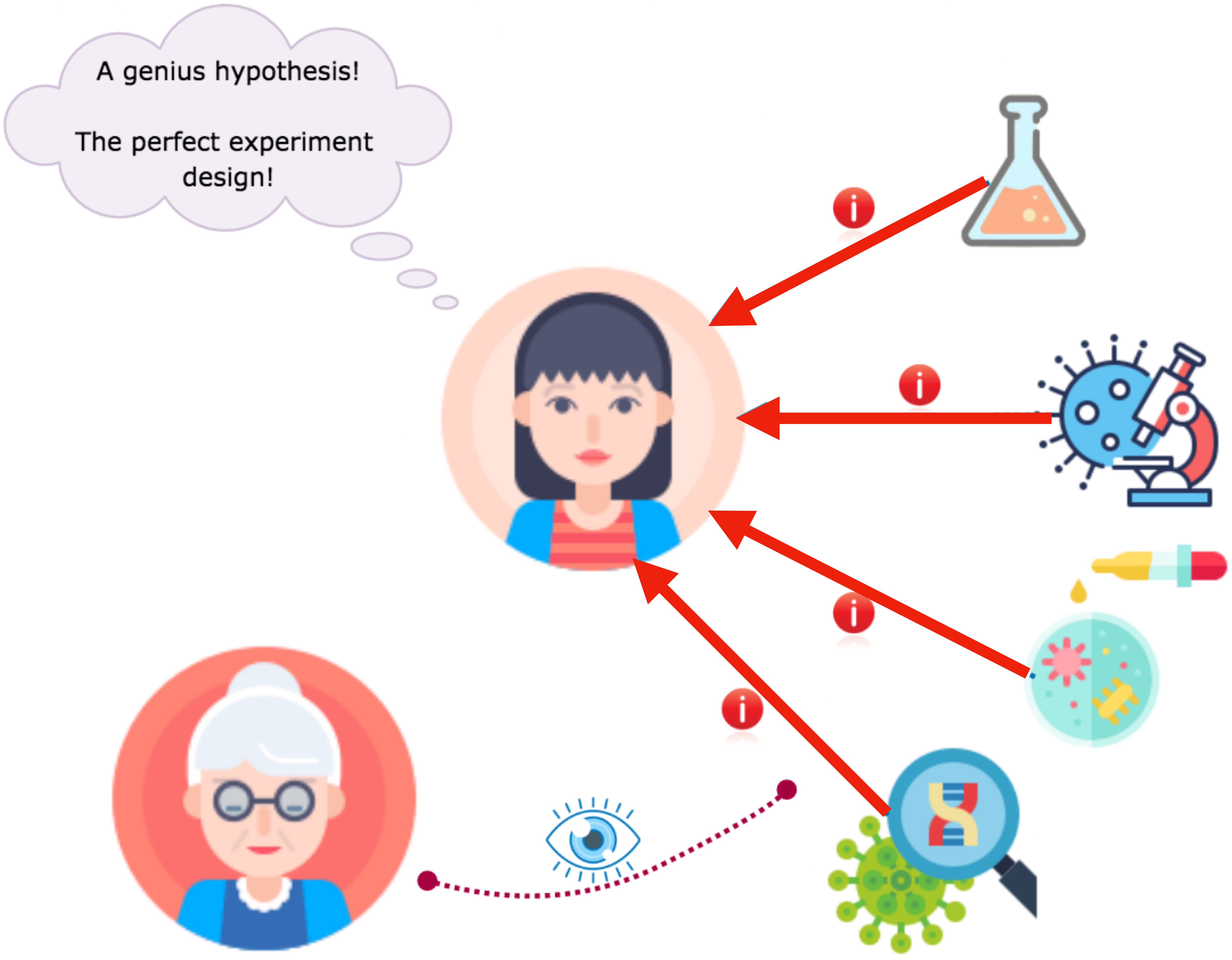
Ran Canetti, Ari Karchmer


A genius hypothesis!
The perfect experiment design!


A genius hypothesis!
The perfect experiment design!


A genius hypothesis!
The perfect experiment design!


A genius hypothesis!
The perfect experiment design!


A genius hypothesis!
The perfect experiment design!

Covert Verifiable Learning Goals

- **Learning:** If Eve reports the experiment results truthfully, Alice learns a good model from her experiments
- **Concept-hiding:** No (maybe, little) information about the molecular relationship is leaked
- **Hypothesis-hiding:** No information about Alice's hypothesis or domain knowledge used to influence the hypothesis is leaked
- **Verifiability:** If Eve tampers with the results, she cannot deceive Alice into learning a faulty model (here let Alice have private access to some random ``ground truth" experiments)

PAC-verification [GRSY20]

- **Learning:** If Prover reports the experiment results truthfully, Verifier learns a good model
- **Concept-hiding**
- **Hypothesis-hiding**
- **Verifiability:** If Prover tampers with the results, she cannot deceive Verifier into learning a faulty model (here let Verifier have private access to some random ``ground truth" examples)

Cryptographic Sensing [IKOS19]

- **Learning:** Sensor obtains an **exact** model/object from the experiments
- **Secrecy:** Object remains hidden to an adversary provided that it is drawn from a distribution of sufficient minimum entropy (entropic security)
- **Hypothesis-hiding**
- **Verifiability**

More Related Work

Or: what this talk is *not* about

- Differentially private learning [KLNRS11]
    - Data privacy — not *learner privacy*
- Verifiable computation [K92, M00]
    - Verify computation steps — not good learning outcomes
    - Requires computational entity

Example Application: Drug Discovery

- How does **A** prevent trade secrets being revealed by experimental design?
- How does **A** prevent the lab from “double-selling” data to competitor?
- How does **A** prevent the lab from returning faulty data (say, given access to some random ground truth examples)?

Example Application: Drug Discovery

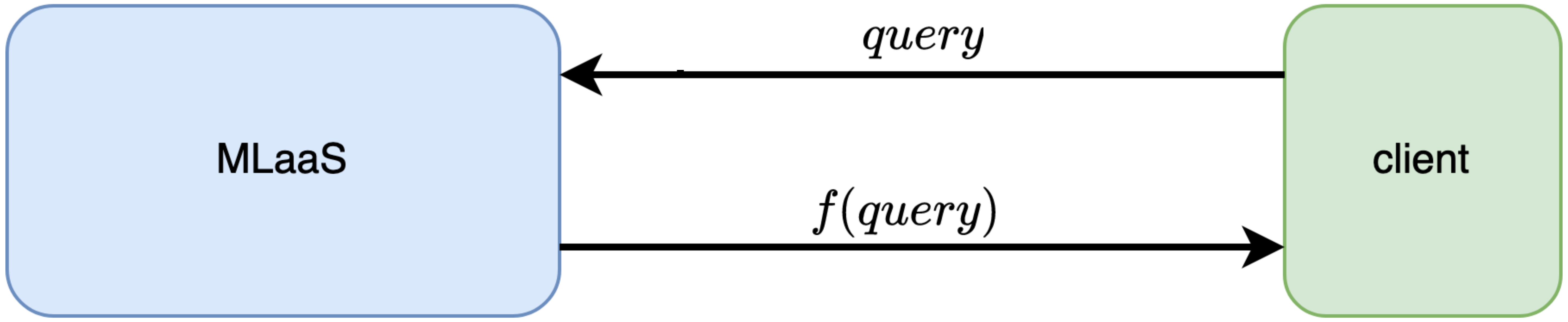
- Cannot execute protocol with the lab
- A must reveal some experiments at some point, while the lab must read the results at some point — no way around this
- We must design an algorithm that interacts only with **“Nature”**
- Q+A only

More applications

- Our work could be useful in any setting where we want to hide what we are learning but can't execute a protocol

- E.g. Model extraction attacks

  Client tries to extract model by using active learning strategies



Contributions

- The **Covert Learning** model, which augments PAC learning with MQ with **hiding** guarantees
    - **Covert Learning** algorithms for **parity functions** and **decision trees (agnostic)**
- The **Covert Verifiable Learning** model, which augments PAC learning with MQ with hiding **and verifiability guarantees**
    - **Covert *Verifiable* Learning** algorithms for **parity functions** and **decision trees (agnostic)**
    - Covert Verifiable Learning even with **no *privately* accessed** examples
    - Covert Verifiable Learning algorithms for O(log n)-juntas with **strong privacy and verifiability** (perfect, statistical)

Contributions Today

- The **Covert Learning** model, which augments PAC learning with MQ with **hiding** guarantees
    - **Covert Learning** algorithms for **parity functions** and **decision trees (agnostic)**
- The **Covert Verifiable Learning** model, which augments PAC learning with MQ with hiding **and verifiability guarantees**
    - **Covert *Verifiable* Learning** algorithms for **parity functions** and **decision trees (agnostic)**
    - Covert Verifiable Learning even with **no *privately* accessed** examples
    - Covert Verifiable Learning algorithms for O(log n)-juntas with **strong privacy and verifiability** (perfect, statistical)

Itinerary

- Establish the Covert Learning model
- Covert Learning for noisy parities
- Covert Learning for heavy Fourier coefficients
- Establish the Covert Verifiable Learning model
- Transform Covert Learning into Covert Verifiable Learning
- Concluding questions and remarks

Alice’s task?
Learning with membership queries

- The structure-activity relationship is a boolean function

  $f : \{feature, \neg feature\}^n \rightarrow \{bind, \neg bind\}$

- Alice's initially hypothesized set of structure-activity models $\iff$ A hypothesis class

- Experiments $\iff$ synthesized membership queries

PAC Learning terminology

- For every $n \in \mathbb{N}$, A **hypothesis class** $\mathscr{H}_n$ is a class of functions $\subseteq \left\{f : \{0,1\}^n \to \{0,1\}\right\}$

- For every $n \in \mathbb{N}$, $\mathscr{D}_n$ is a **concept class**, where every $D_n \in \mathscr{D}_n$ is a distribution over $\{0,1\}^n \times \{0,1\}$

- For an **example** $(x, y) \sim D_n$, x is an **input**, and y is a **label**

- A loss function (w.r.t. a concept D_n) $\mathscr{L}_{D_n} : \mathscr{H}_n \to [0,1]$ quantifies how good a hypothesis $h \in \mathscr{H}$ is at approximating a concept. For instance misclassification error: $\mathscr{L}_D(h) = \Pr_{(x,y)\sim D}\left[h(x) \neq y\right]$

Agnostic PAC Learning

A hypothesis class H is agnostic PAC learnable (w.r.t. concept class $\mathscr{D}_n$) if (roughly)

There is an algorithm A receiving enoug

For any concept $D \in \mathscr{D}_n, \epsilon, \delta > 0$,

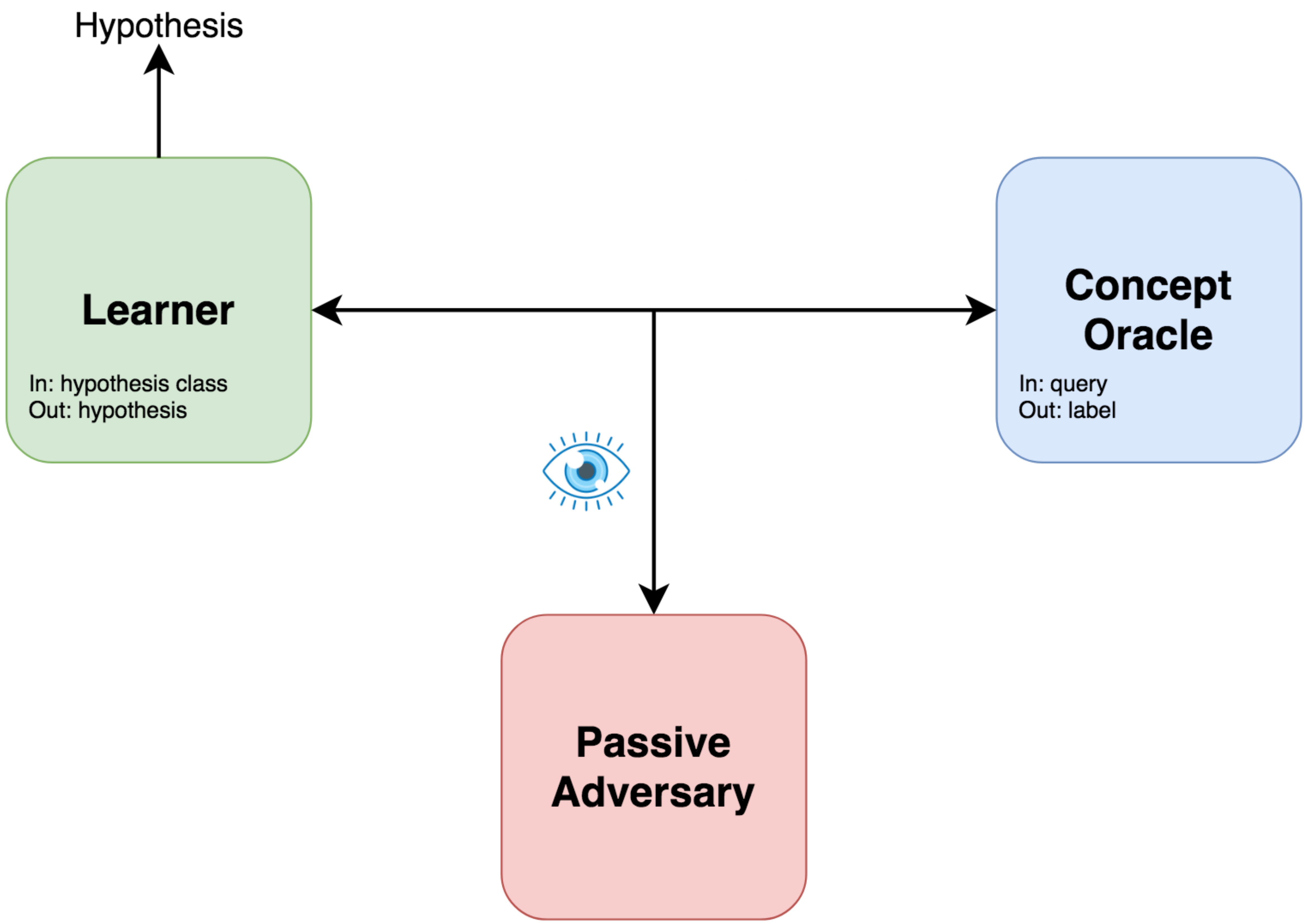
A outputs $h \in H$ s.t. $\Pr_A \Big[\mathscr{L}_D(h) \leq \mathscr{L}_D($

$\mathscr{L}_D(H) = \inf_{h \in H} \mathscr{L}_D(h)$

- With membership queries: there is also an oracle that allows the learning algorithm to directly query the concept
- **Agnostic**: H need not contain a perfect hypothesis for the concept
- H has a big impact on the resulting function which is output by $A \longrightarrow$ important choice, whatever H is

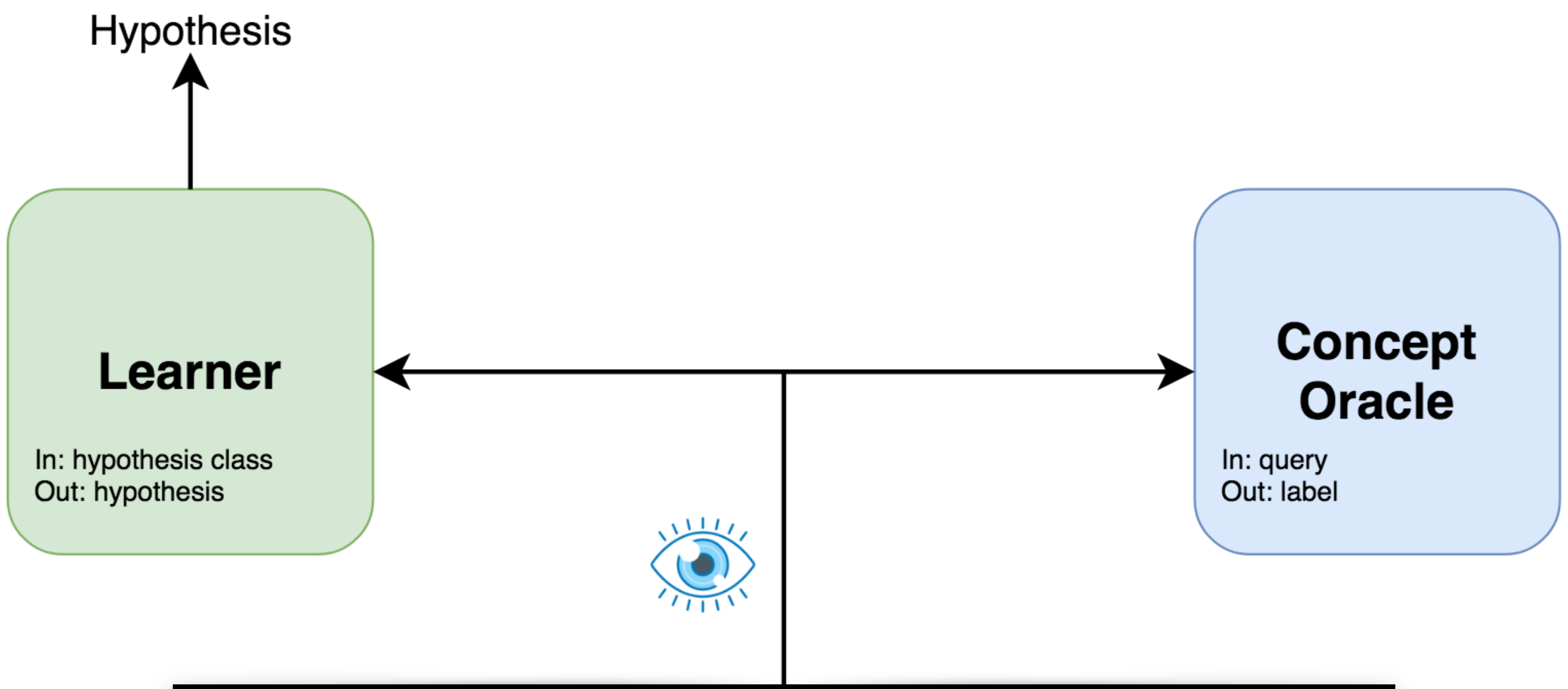
The Covert Learning Model

- $\mathscr{C}_n = \{H_n^i\}_{i \in [m]}$ a **collection** of hypothesis classes
- Learner input: $\epsilon, \delta < 1,\ H_n \in \mathscr{C}_n$
- Learner queries concept
- Learner outputs: for any $D_n \in \mathscr{D}_n$, a hypothesis $h \in H_n$ such that $\Pr_A \left[\mathscr{L}_D(h) \leq \mathscr{L}_D(H) + \epsilon \right] \geq 1 - \delta$



The Covert Learning Model

- Adversary attempts to deduce information about either the concept $D_n \in \mathscr{D}_n$ or the hypothesis class $H_n \in \mathscr{C}_n$
- **Objection**: some concept classes we cannot hide, e.g. the constant function
- Adversary could apply Occam learning



Occam Learning (informal) An Occam learning algorithm is an algorithm that, given any set of examples to a concept, outputs a hypothesis which is

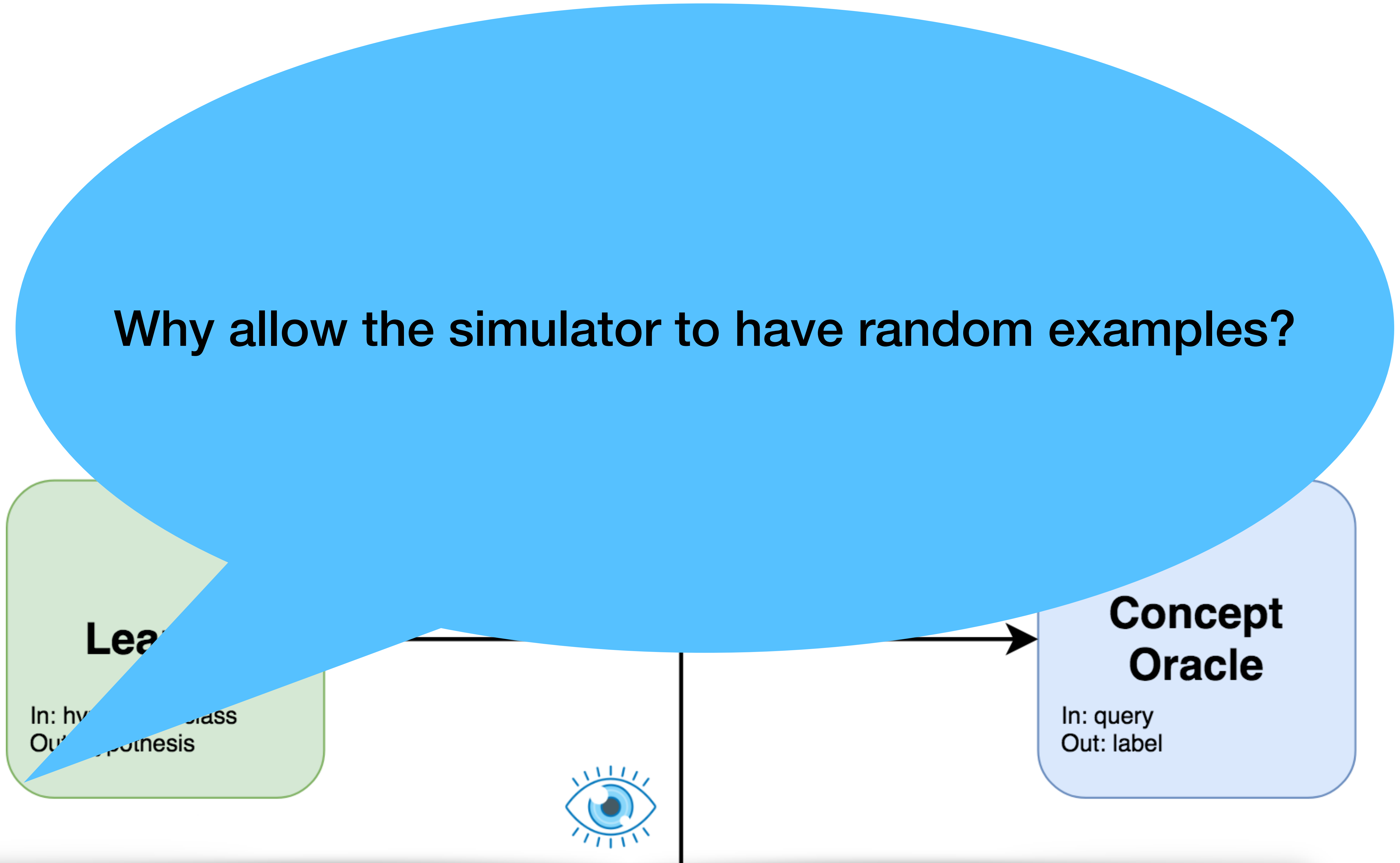
1) consistent with all examples,
2) “succint”

The Covert Learning Model

There exists a p.p.t. **simulator** such that for every $H_n \in \mathscr{C}_n, D_n \in \mathscr{D}_n$

$$\{Sim(S)\} \approx \{interaction\ transcript\}$$

where $S \sim D_n$ (random examples of D_n).



“Zero-knowledge-like” in the presence of public data set of random examples

Many learning problems thought to be computationally hard to learn from random examples (e.g. decision trees, small depth circuits, parities with noise)

Random examples are the minimum leakage, not just a result of the definition

Trivial example

Covert Learning is easy when PAC learning is possible

- Every $H_n \in \mathscr{C}_n$ is efficiently PAC learnable
- Covert Learning algorithm:
  - Request a sufficiently large set of random examples from the oracle.
  - Run the PAC learning algorithm.
  - The simulator returns the random examples given as input.
- Examples: Constant term DNFs, parities (in the absence of noise).

Itinerary

- Establish the Covert Learning model
- Covert Learning for noisy parities
- Covert Learning for heavy Fourier coefficients
- Establish the Covert Verifiable Learning model
- Transform Covert Learning into Covert Verifiable Learning
- Concluding questions and remarks

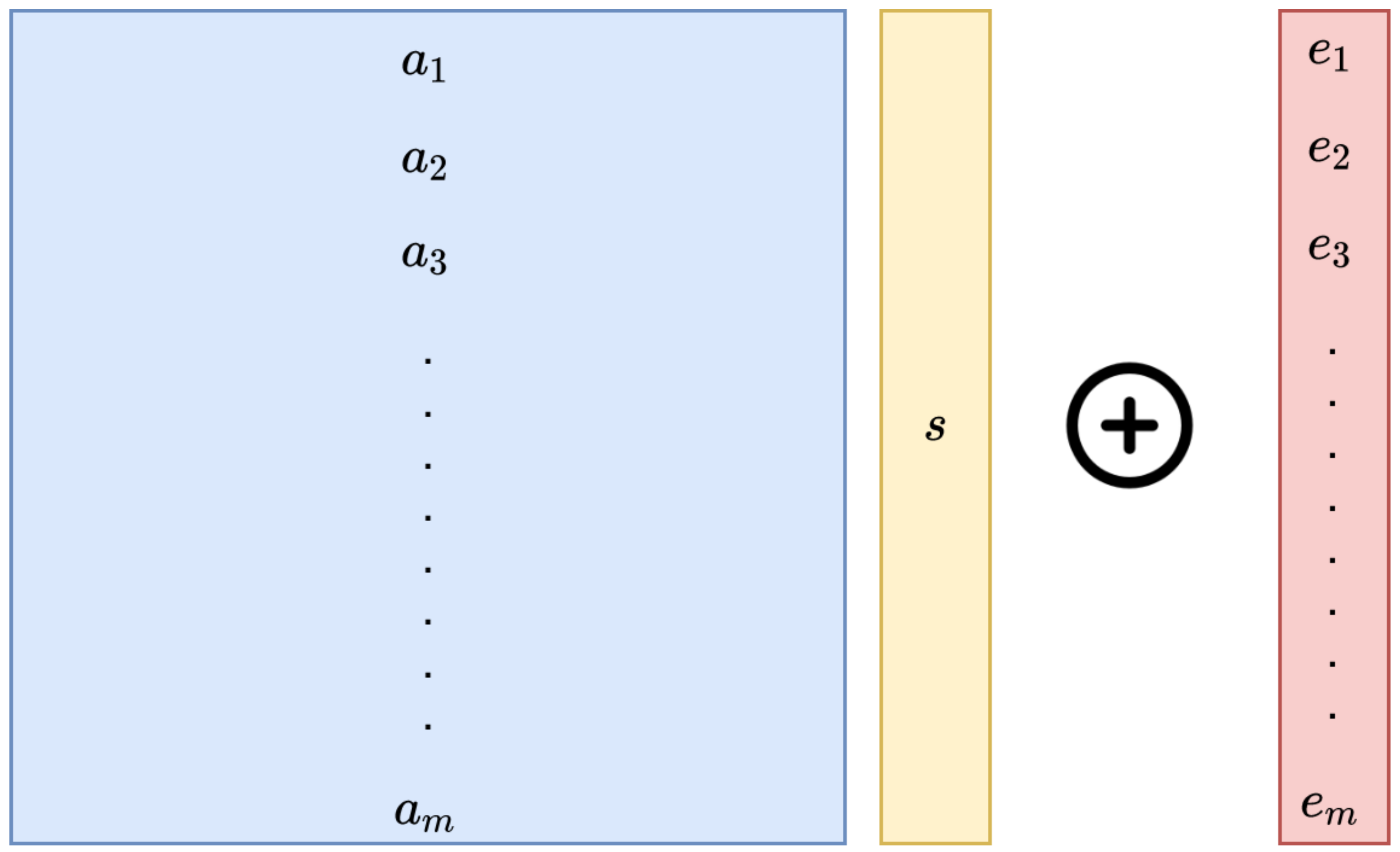
Covert Learning for noisy parities

Parities with no noise are trivial, so what about the noisy case?

Focus on the concept hiding guarantee

Covert Learning for noisy parities

LPN



LPN distribution [BFKL92]:

Sample s $\in \{0,1\}^n$, $e \in \{0,1\}$ from Bernoulli r.v. with mean constant $p < 1/2$

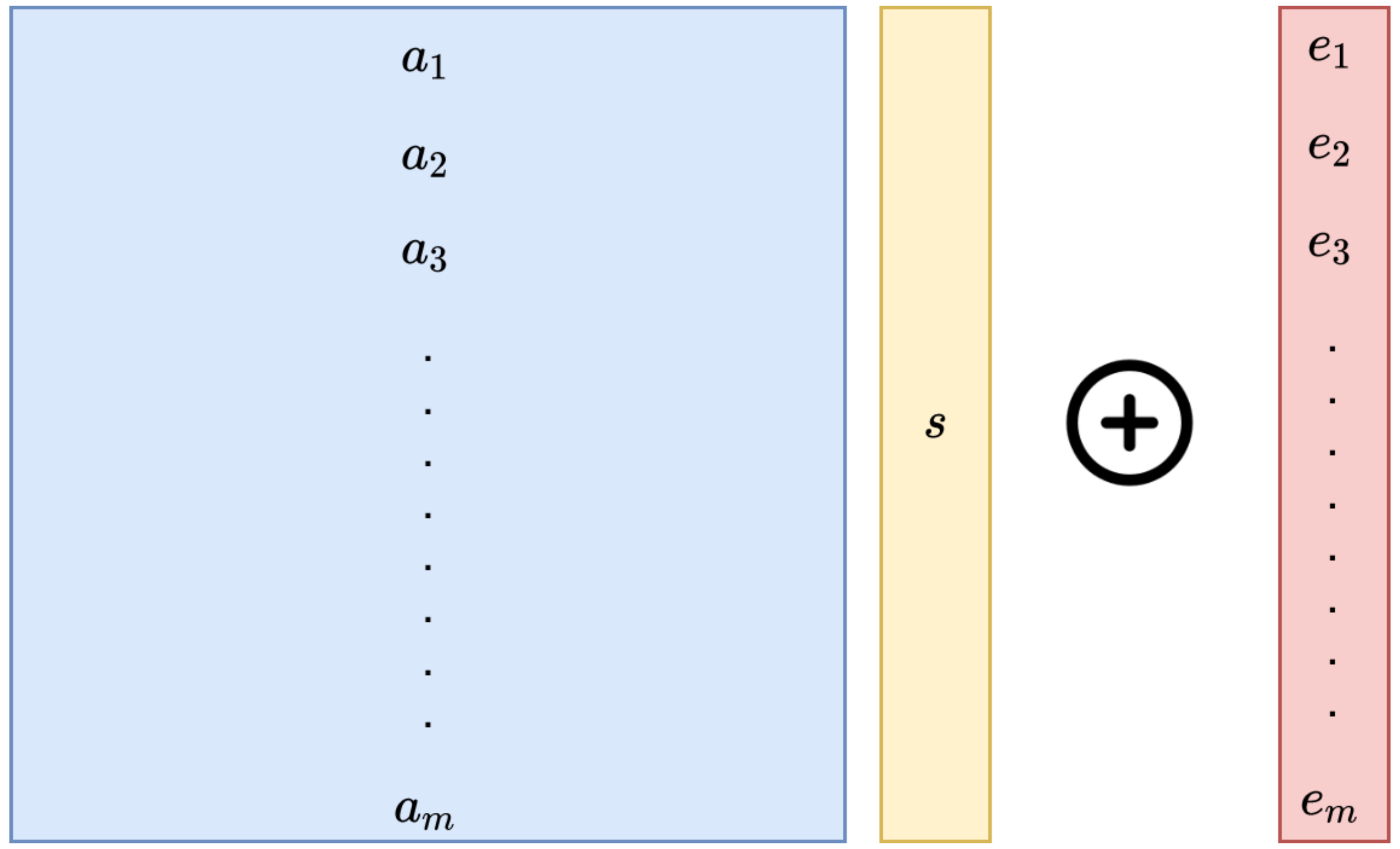
Sample $a \in \{0,1\}^n$ uniformly at random

Return $a, \langle a, \mathrm{s} \rangle \oplus e$ (s is persistent over each example)

- **Search LPN:** find s.
- **Decision LPN:** distinguish $a, \langle a, \mathrm{s} \rangle \oplus e$ from uniformly random. Both are thought to be subexponentially-hard

Covert Learning for noisy parities

Low-noise LPN



Low-noise LPN distribution [Ale03]:

Sample s $\in \{0,1\}^n$, $e \in \{0,1\}$ from Bernoulli r.v. with mean $p = 1/\sqrt{n}$

Sample $a \in \{0,1\}^n$ uniformly at random

Return $a, \langle a, \mathrm{s}\rangle \oplus e$ (s is persistent over each example)

- **Search LPN:** find s.
- **Decision LPN:** distinguish $a, \langle a, \mathrm{s}\rangle \oplus e$ from uniformly random. Both are thought to be subexponentially-hard

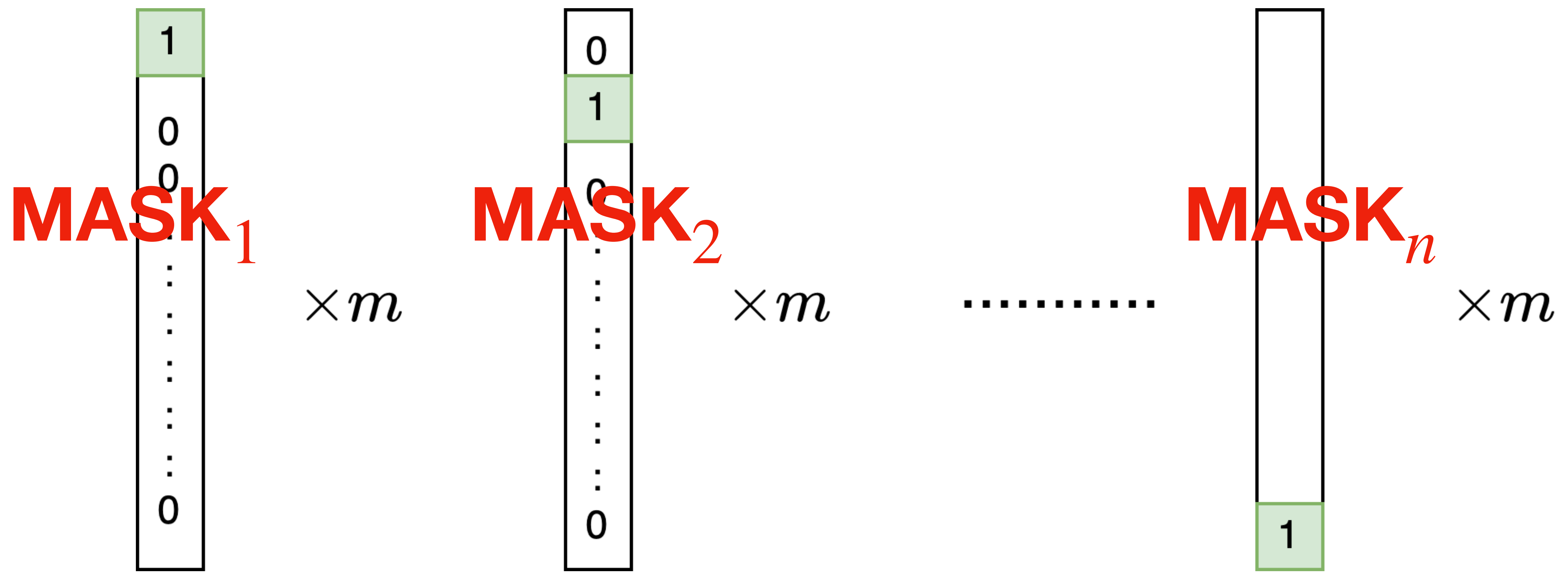
Is there a covert learning algorithm for learning s?

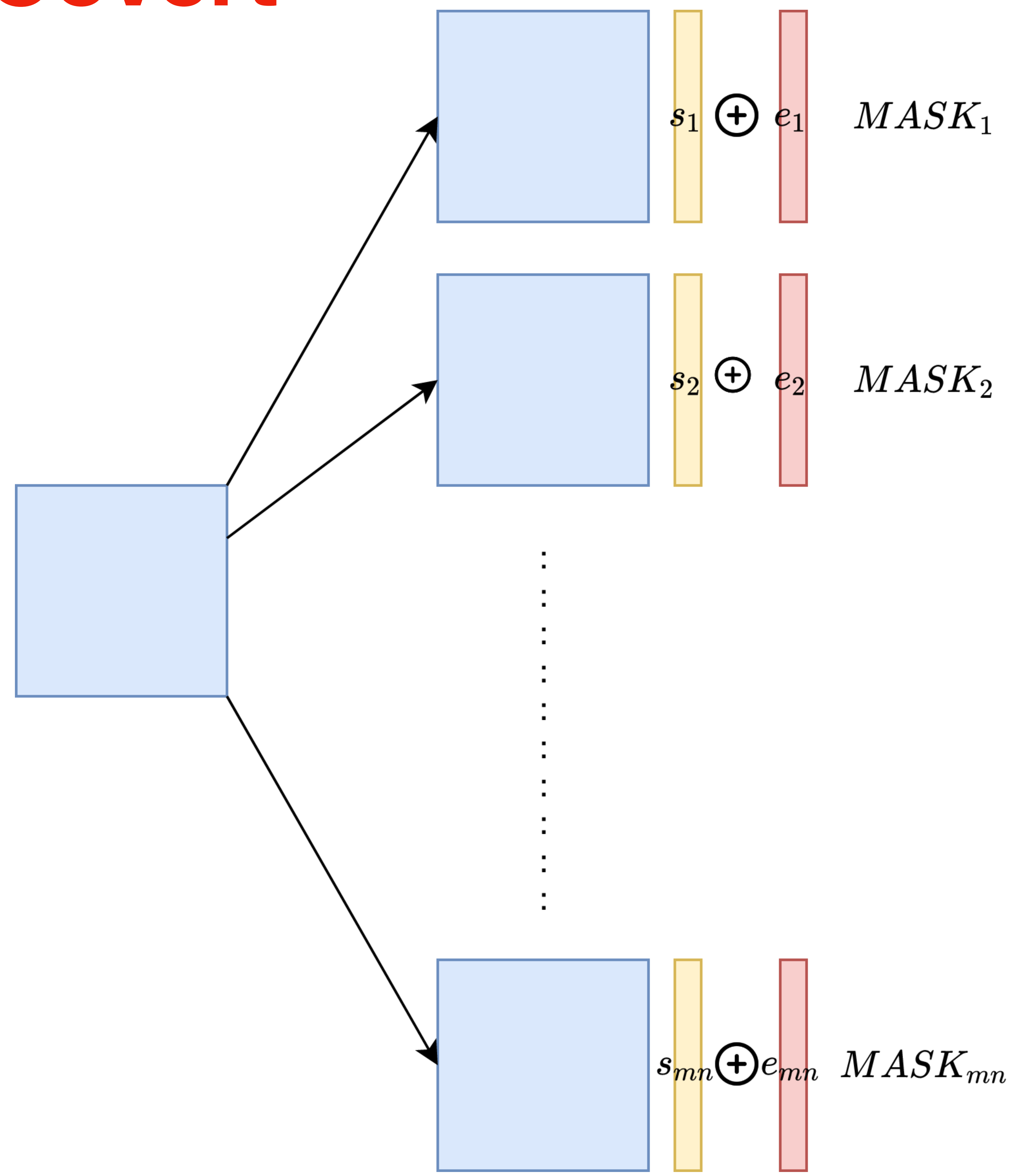
Covert Learning for noisy parities

Theorem: (informal) If low-noise LPN is hard, then there exists a Covert Learning algorithm for parities with respect to the low-noise LPN distribution

(actually, there exists a Covert Learning algorithm for noisy parities **unconditionally**. Recall previous example.)

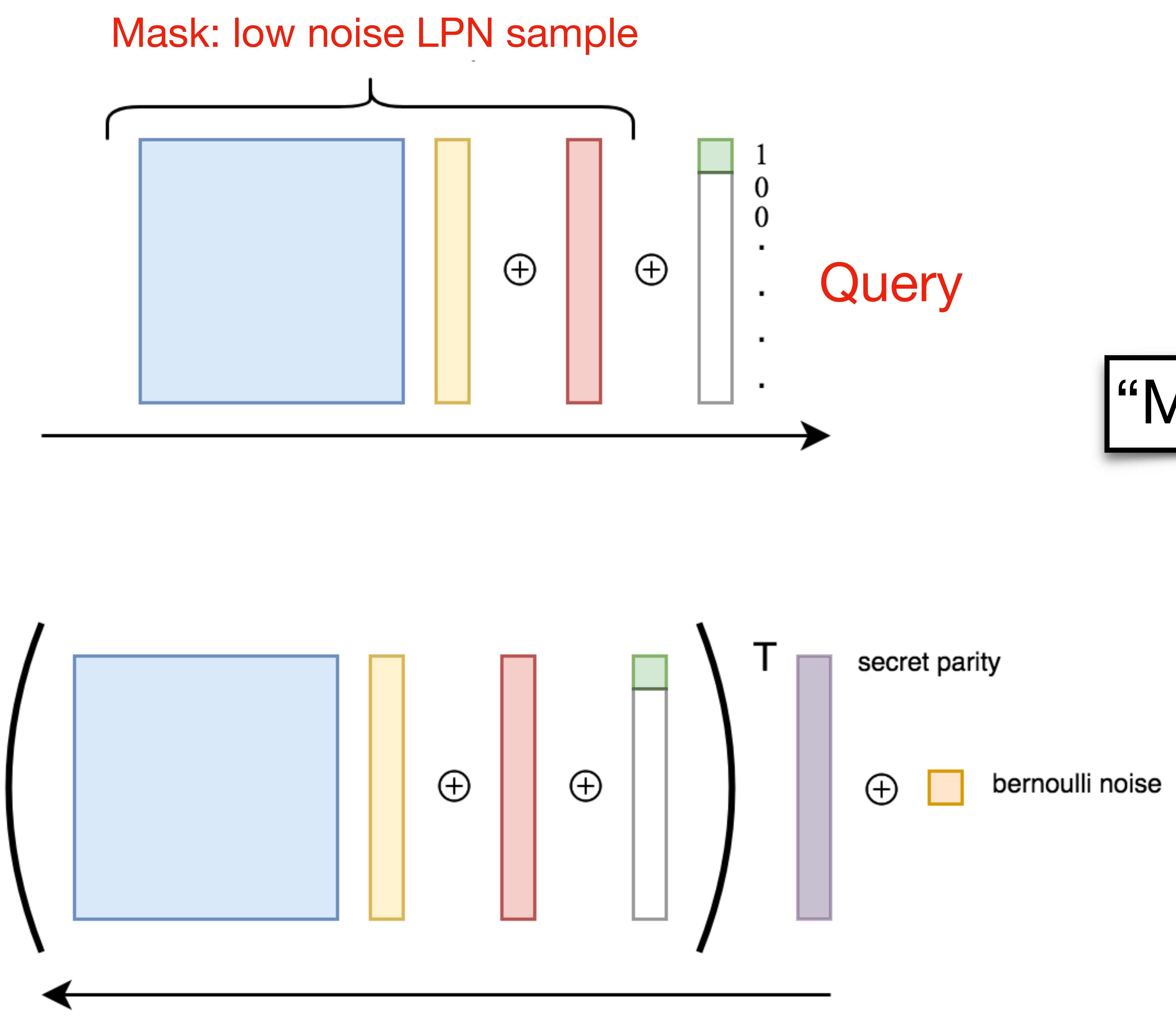
Learning noisy parities the ~~leaky~~ Covert way





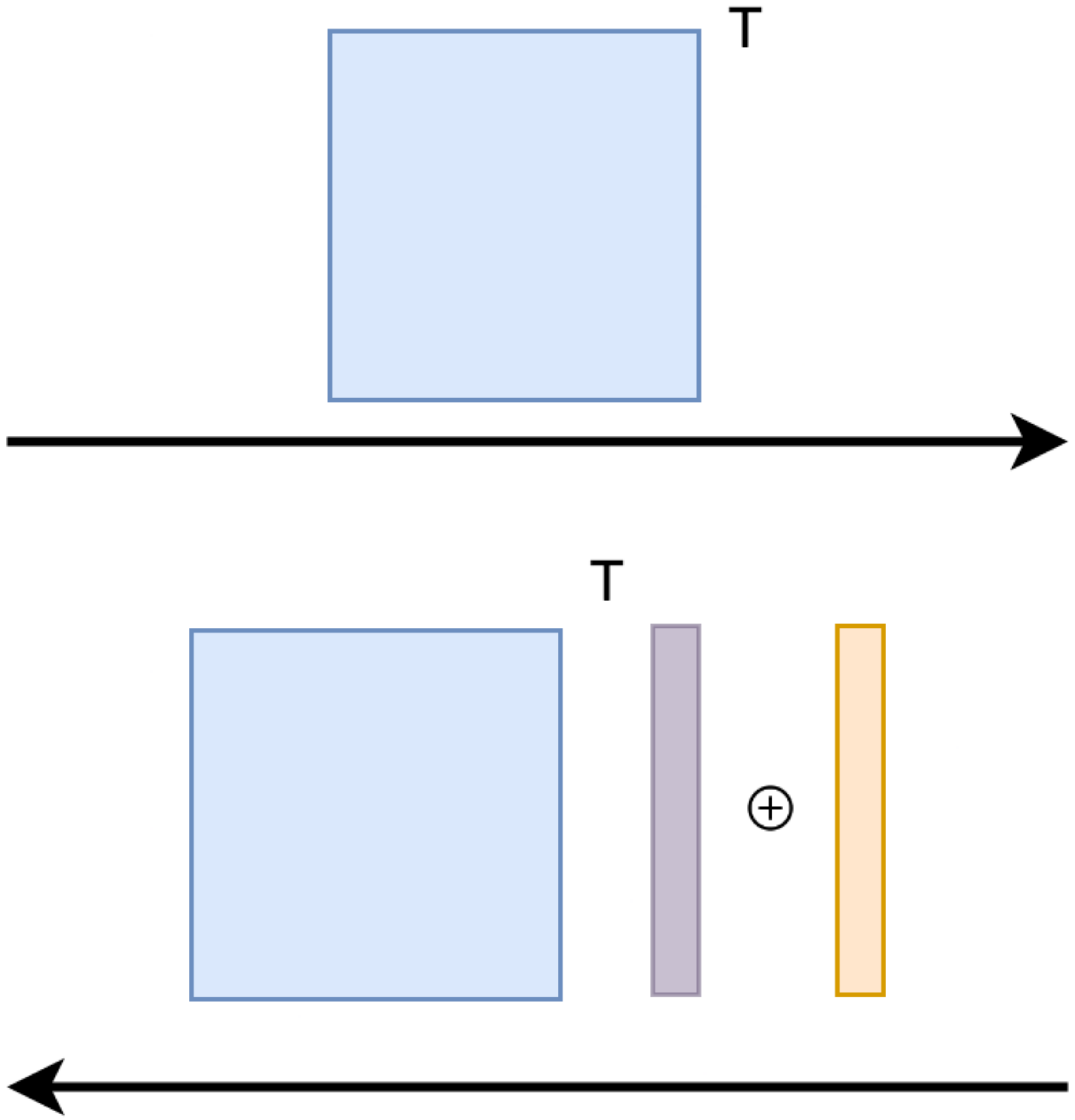
- Send pseudorandom queries
- Generate a pseudorandom mask for each query by using LPN distribution

Covert Learning for noisy parities

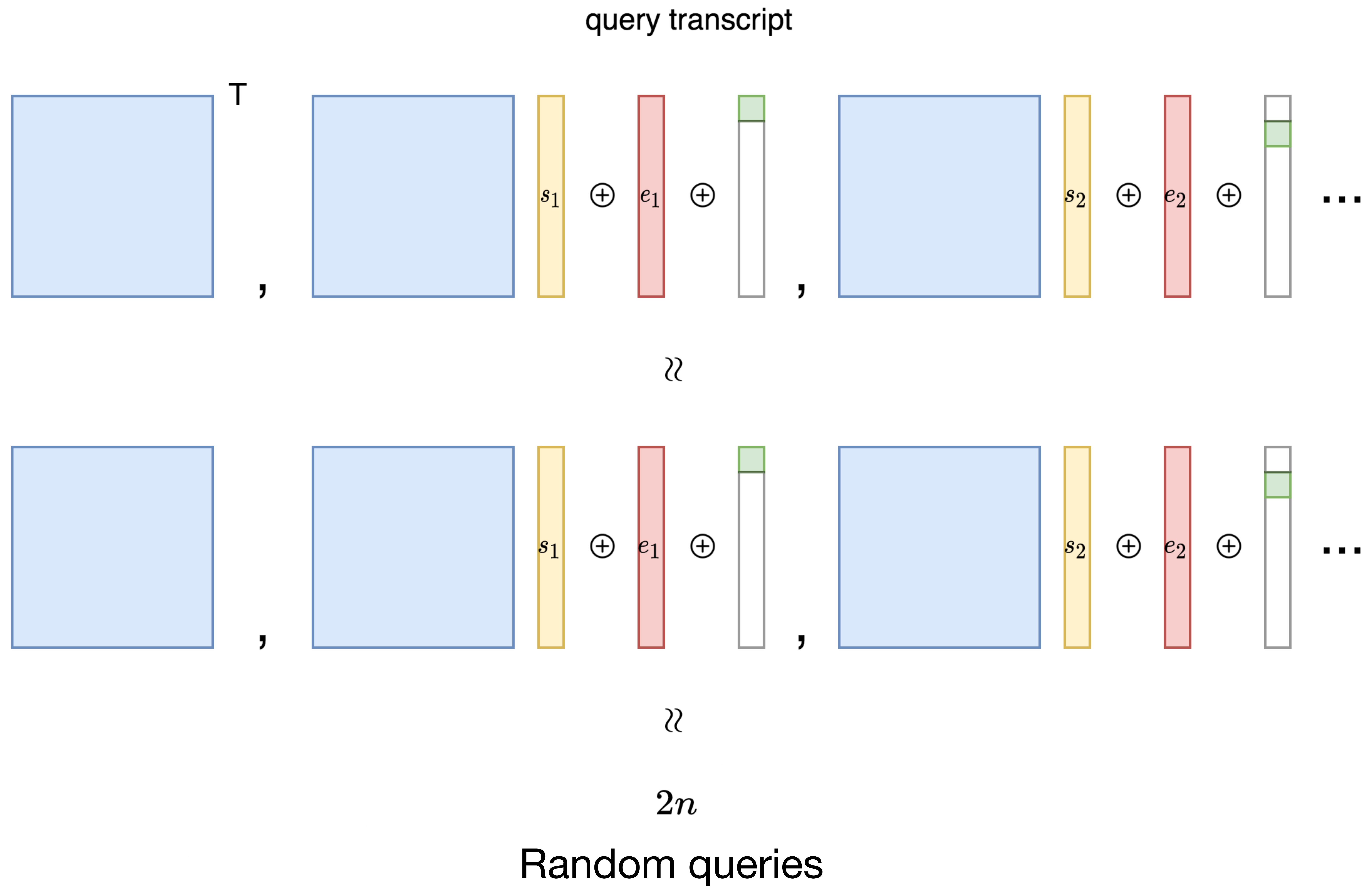


“Masked queries”

Covert Learning for noisy parities



Covert Learning for noisy parities

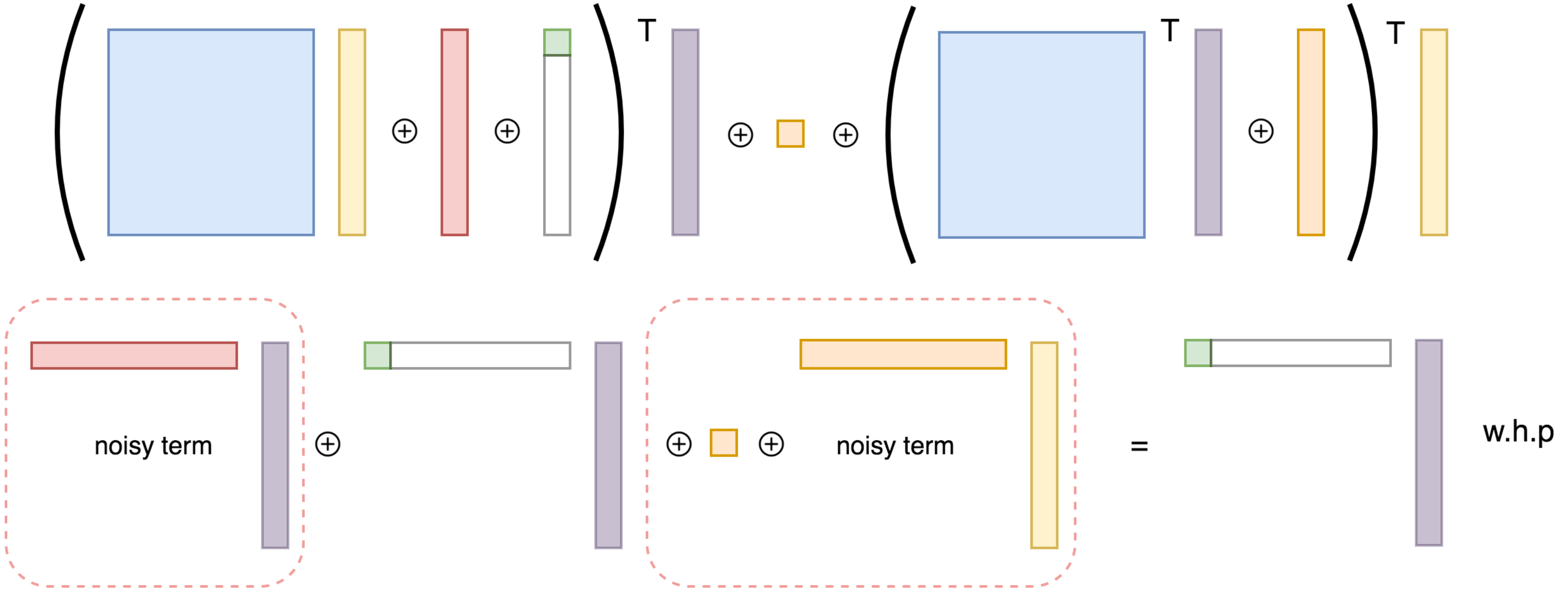


Covert Learning for noisy parities

X-or lemma For $\mu \in [0,\ 1/2]$, and random variables $E_1 \ \cdots \ E_m$ that are i.i.d. from Bernoulli r.v. with mean μ, we have that

$$\Pr\left[\bigoplus_{i=1}^{m} E_i = 0\right] = \frac{1}{2} + \frac{1}{2}(1-2\mu)^m$$

Decoding procedure:



Noisy terms are biased to 0 w.p. $\dfrac{1}{2} + \Omega(1)$

Repetition and majority voting to decode bit-by-bit

Can we covertly learn more interesting concepts?

Itinerary

- Establish the Covert Learning model
- Covert Learning for noisy parities
- Covert Learning for heavy Fourier coefficients
- Establish the Covert Verifiable Learning model
- Transform Covert Learning into Covert Verifiable Learning
- Concluding questions and remarks

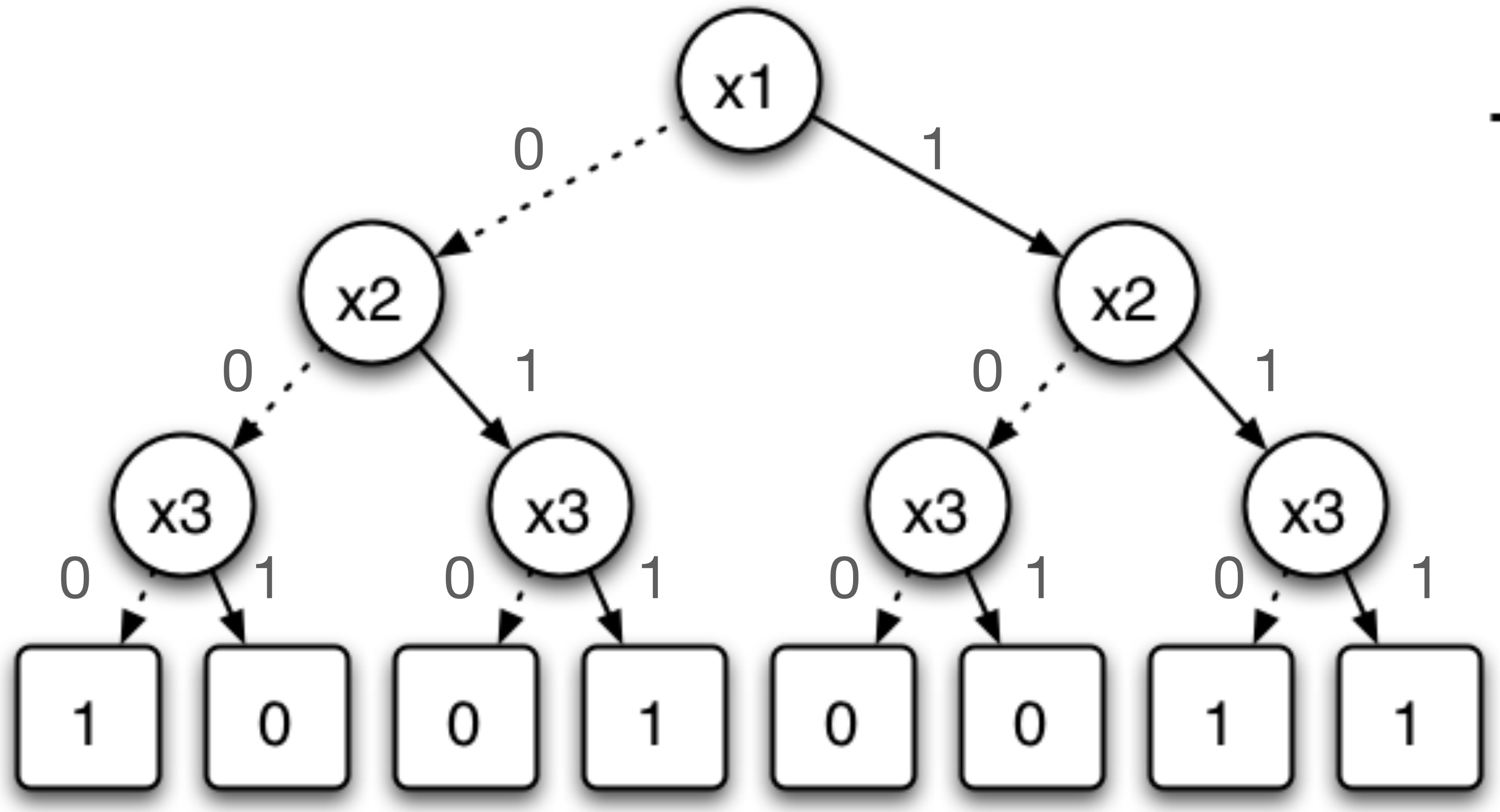
Boolean Fourier Analysis

Review

- Let $f : \mathbb{F}_2^n \to \mathbb{R}$. In particular, we care about functions $f : \mathbb{F}_2^n \to \{-1,1\}$
- For $S \subseteq [n]$, we define $\chi_S : \mathbb{F}_2^n \to \mathbb{R}$ by $\chi_S(\underset{\hat{f}(S)}{x}) = (-1)^{\sum_{i \in S} x_i}$
- The Fourier expansion of $f : \mathbb{F}_2^n \to \mathbb{R}$ is $f(x) = \sum_{S \subseteq [n]} \hat{f}(S)\chi_S(x)$ where $\hat{f}(S)$ is the Fourier coefficient on S
- Every function can be uniquely represented by its Fourier expansion
- We say that $\hat{f}(S)$ is "heavy" if $\hat{f}(S) \geq 1/poly(n)$, and the degree of $\hat{f}(S)$ is $|S|$

Why Learn Fourier Coefficients?

- By definition, $\hat{f}(S) = \mathbb{E}_x[f(x)\chi_S(x)]$
- If we know S such that $\hat{f}(S) \geq \tau$, then we have a hypothesis that agrees with f on $1/2 + \tau/2$ fraction of inputs (just take χ_S)
- Good for learning w.r.t. uniform distribution

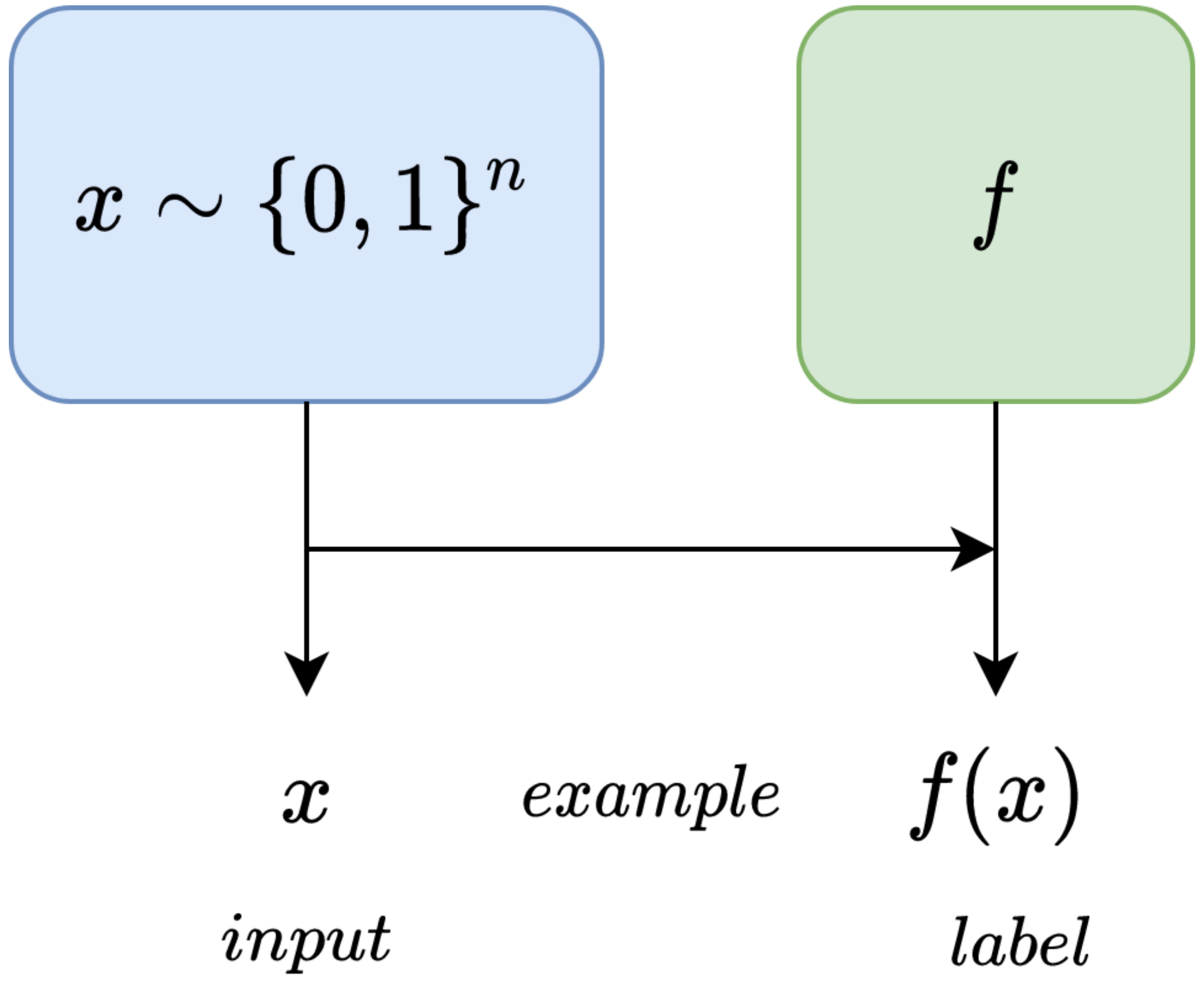


x1	x2	x3	*f*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

- Finding all heavy Fourier coefficients of degree $O(log\ n)$ is strong enough to efficiently learn $poly(n)$-size decision trees. Not known in plain PAC learning model
- Our goal: find (covertly) the $O(log\ n)$ Fourier coefficients of a function

Covert Learning of low degree Fourier coefficients

$f : \{0,1\}^n \to \{-1,1\}$



- Membership query q bypasses distribution over inputs and returns $f(q)$
- **Goal:** find Fourier coefficients of f while satisfying privacy guarantees

Covert Learning of low degree Fourier coefficients

Highlighting hypothesis-hiding

- Learner has a hypothesis (due to expert domain knowledge) that all heavy Fourier coefficients of a function are contained in a subset $T \subseteq [n]$.
- **Valuable:** this knowledge could improve the efficiency of a normal learning algorithm (e.g. KM algorithm).
- **Leaky:** repeated restrictions of KM queries clearly reveals information on T
- How to hide this expert domain knowledge?

Answer: do Covert Learning for collection of hypothesis classes, where each hypothesis class in indexed by $T \subseteq [n]$

Covert Learning then hides all information about T

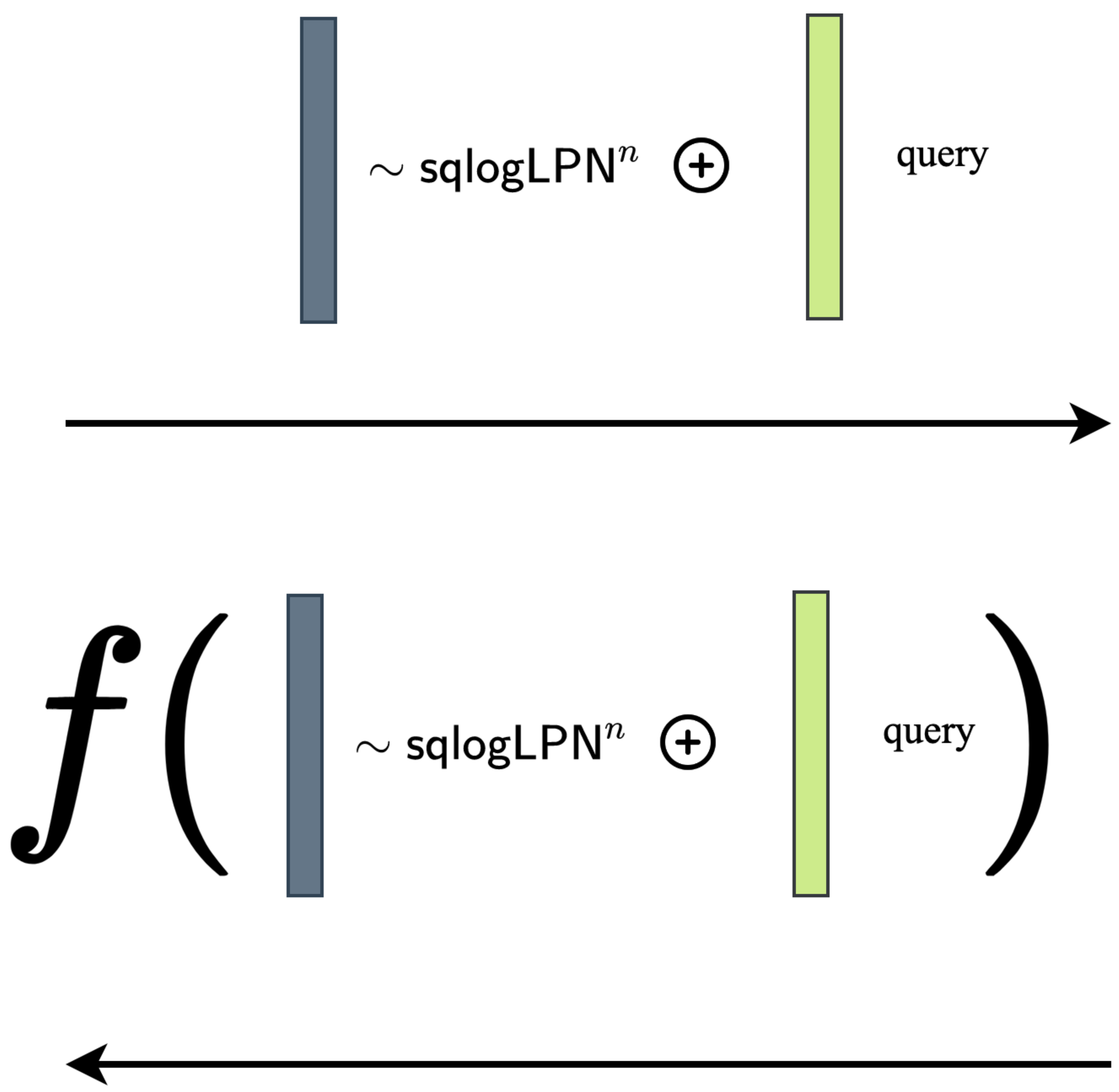
Theorem: (informal) Under the subexponential LPN assumption, the collection of all low-degree Fourier subsets is covertly learnable.

Covert Learning of low degree Fourier coefficients

Squared-log entropy assumption

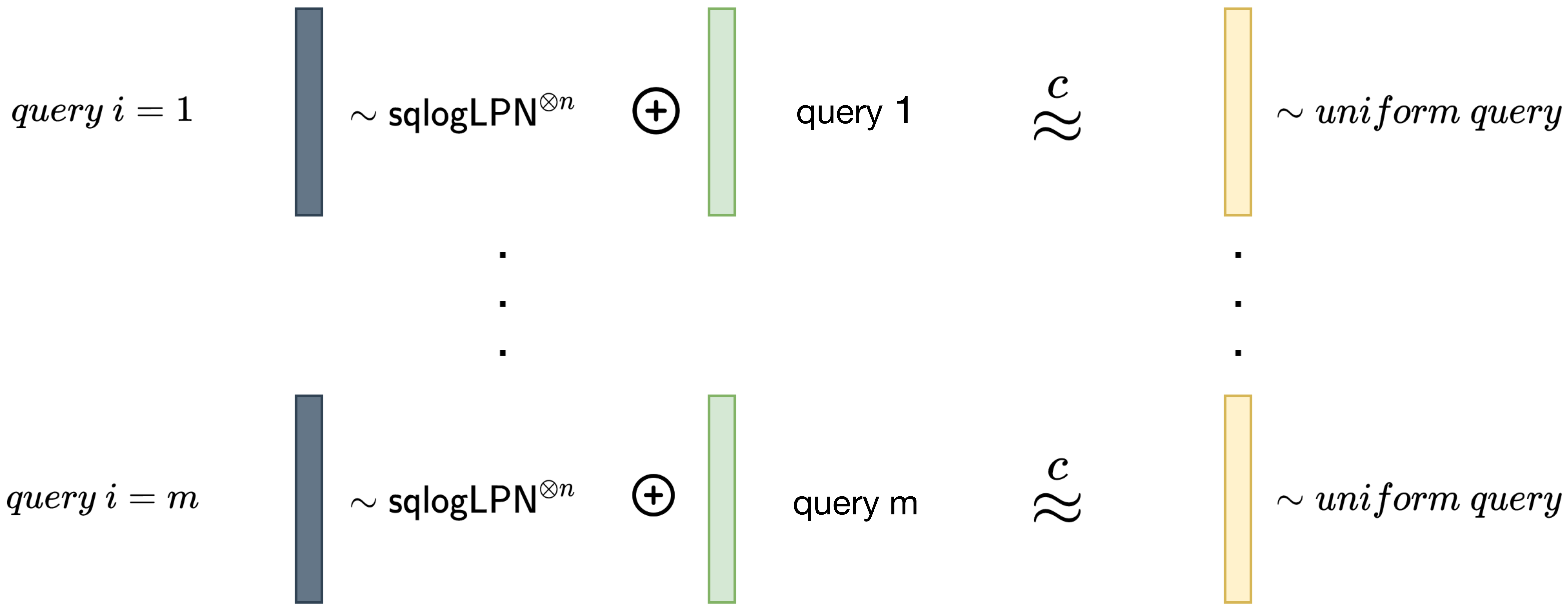
- Extend the technique from the previous result: “masked queries"
- Need **decisional** LPN variant $(\mathsf{sqlogLPN})$ that provides super-polynomial hardness with n-bit secrets of just $\Theta(log^2(n))$ entropy (due to [YZ16])
- Variant implied by subexponential LPN: $2^{\sqrt{n}}$-hard with $2^{\sqrt{n}}$ samples [YZ16]

Covert Learning of low degree Fourier coefficients



Covert Learning of low degree Fourier coefficients

Simulator



Covert Learning of low degree Fourier coefficients

- Let otp $\sim$ sqlogLPN^n
- For query $q \in \{0,1\}^n$, let $\hat{q} = q \oplus$ otp ("masked query" technique)
- What can we say about $f(\hat{q})$?
- Let ϕ be a randomized mapping defined $\phi(f(\hat{q})) := f(\hat{q}) \cdot \chi_s(r)$, where s is the sqlogLPN secret used on the otp for q, and r is a random n-bit string

Lemma: If $k \subseteq [n]$ s.t. $|k| = O(log\ n)$ and $\hat{f}(k) \geq \tau$, then there exists a small constant c such that

$$\mathop{\mathbb{E}}_{\phi,q,\hat{q}}\left[\phi(f(\hat{q})) \ \cdot \ \chi_k(q)\right] \geq \Omega(\tau n^{-c})$$

In particular, if $\hat{f}(S) \geq 1/poly(n)$, then $\mathbb{E}[\phi(f(\hat{q}))\chi_k(q)] \geq 1/poly(n)$.

Covert Learning of low degree Fourier coefficients

- The lemma provides a "Goldreich-Levin type" environment

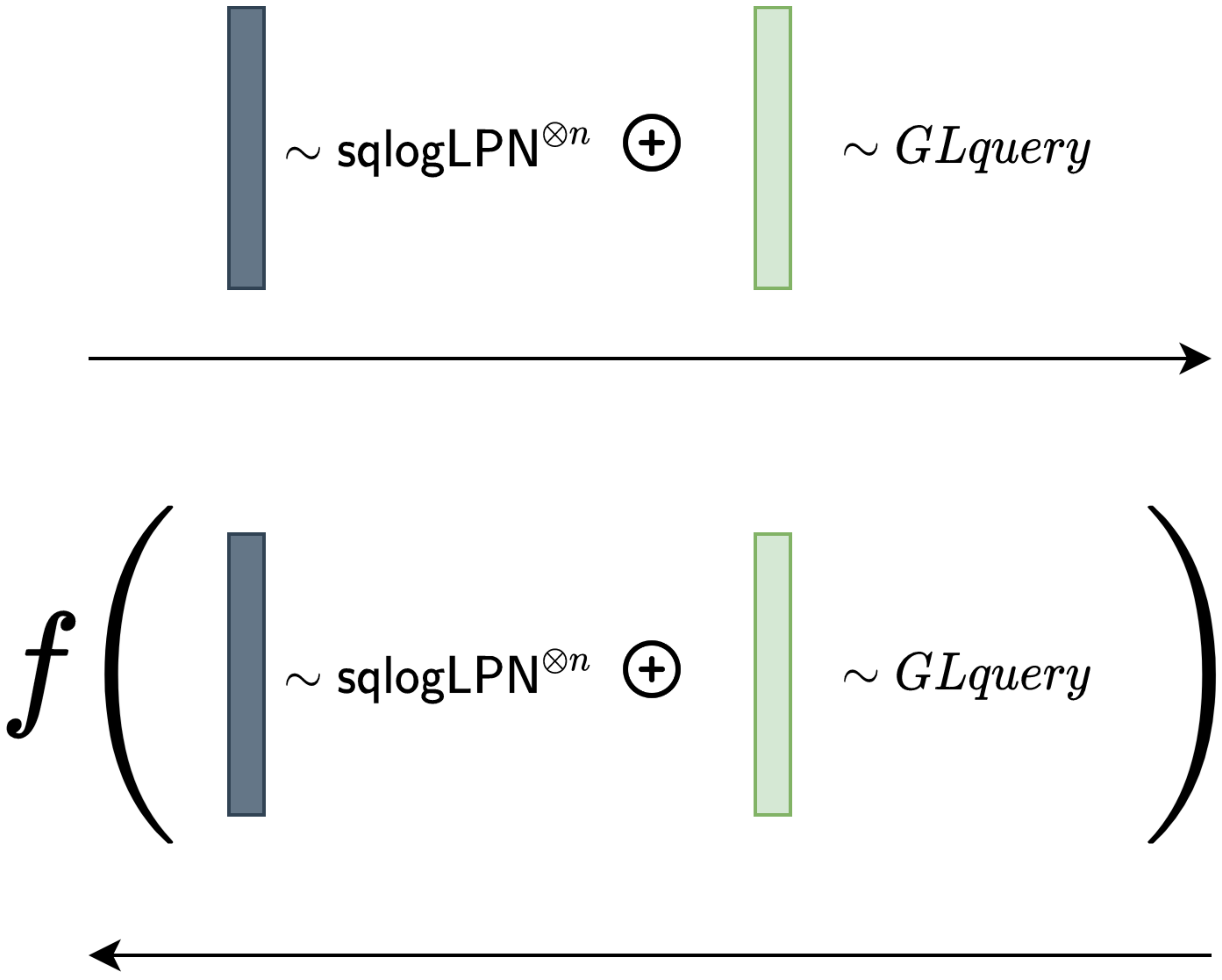
$$\mathop{\mathbb{E}}_{\phi,q,\hat{q}}\left[\boxed{\phi(f(\hat{q}))}\cdot\chi_k(q)\right] \geq \Omega(\tau n^{-c})$$

Noisy Predictor

- The Goldriech-Levin reduction/algorithm gives a method for extracting a k s.t. $\hat{f}(k) \geq \tau$ in time $poly(n, \frac{1}{\tau})$ (w.h.p.)

Covert Learning of low degree Fourier coefficients

- Run the Goldreich-Levin algorithm in the "masked query regime"
- Can run Goldreich-Levin on some subset of interesting indices
    - This encodes secret hypothesis T
- Pseudorandomness hides T
- At the end we obtained $S \in T : |S| \leq O(\log n)$, $\hat{f}(S) \geq 1/poly(n)$



Covert Learning of poly(n) size decision trees

Sufficient to produce an approximation to f that is competitive with the optimal $poly(n)$-size decision tree approximating f.

Theorem: (informal) Under the subexponential LPN assumption, the collection of all subsets of decision trees is covertly learnable.

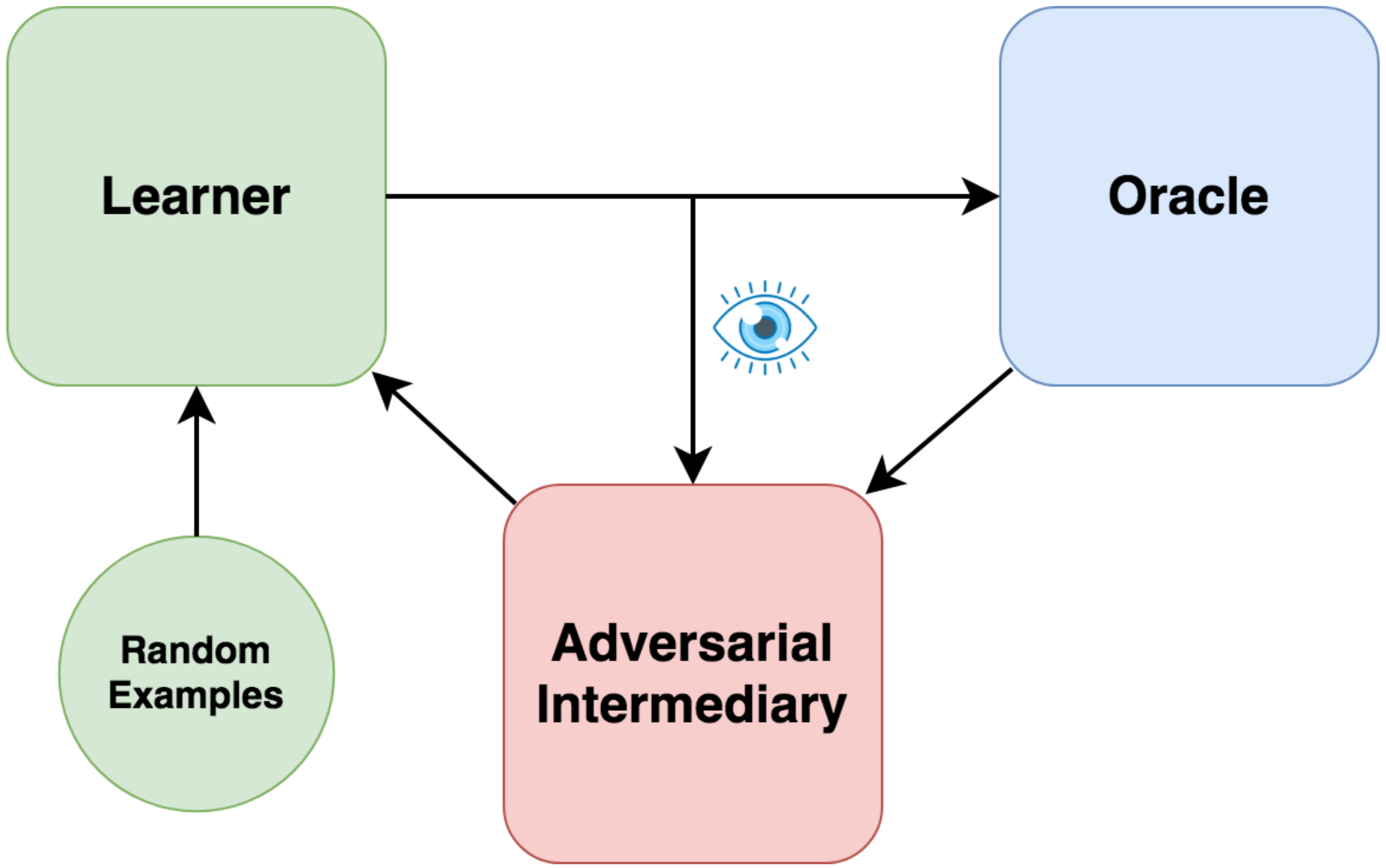
Itinerary

- Establish the Covert Learning model
- Covert Learning for noisy parities
- Covert Learning for heavy Fourier coefficients
- Establish the Covert Verifiable Learning model
- Transform Covert Learning into Covert Verifiable Learning
- Concluding questions and remarks

Covert Verifiable Learning

A malicious adversary

- Interactive between the learner and adversarial intermediary (AI)
- AI monitors access to the membership oracle
- The learner request queries from oracle, but responses my be corrupted by AI



Covert Verifiable Learning

The verifiability guarantee

Goal: If AI chooses to corrupt, the learner should not output a faulty hypothesis except with small probability (similar to soundness)

- Allow the learner to abort
- Access to set of uniformly random “ground truth” examples

Verifiability for learning algorithm A for collection $\mathscr{C}_n$

Inputs: $H_n \in \mathscr{C}_n, \epsilon, \delta, \mathcal{S}$

For any $D_n \in \mathscr{D}_n$, any $H_n \in \mathscr{C}_n$, $\mathcal{S} \sim D_n^m$, any AI I which corrupts oracle responses,

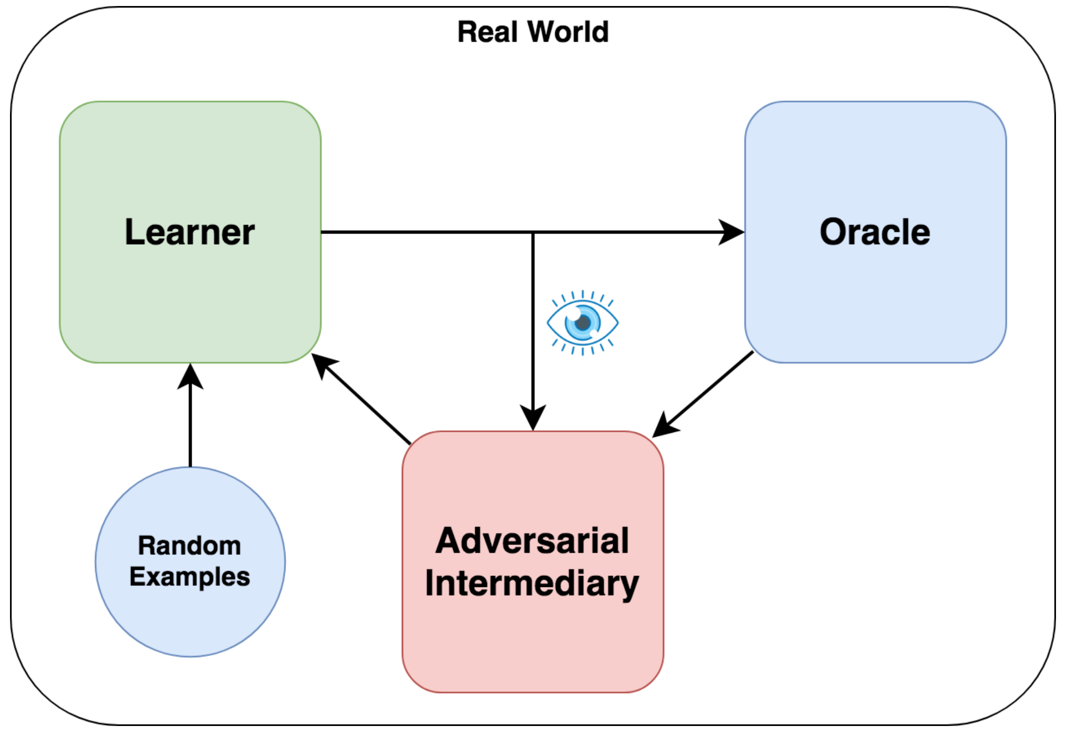
$$\Pr\left[\mathscr{L}(A) > \mathscr{L}(H_n) + \epsilon \,\middle|\, A \neq \text{abort}\right] < \delta$$

We say that verifiability is computational if I is p.p.t..

Covert Verifiable Learning

How to extend privacy?

Real:



Adversary (a distinguisher) chooses
$H_n \in \mathscr{C}_n, D_n \in \mathscr{D}_n, \epsilon, \delta$

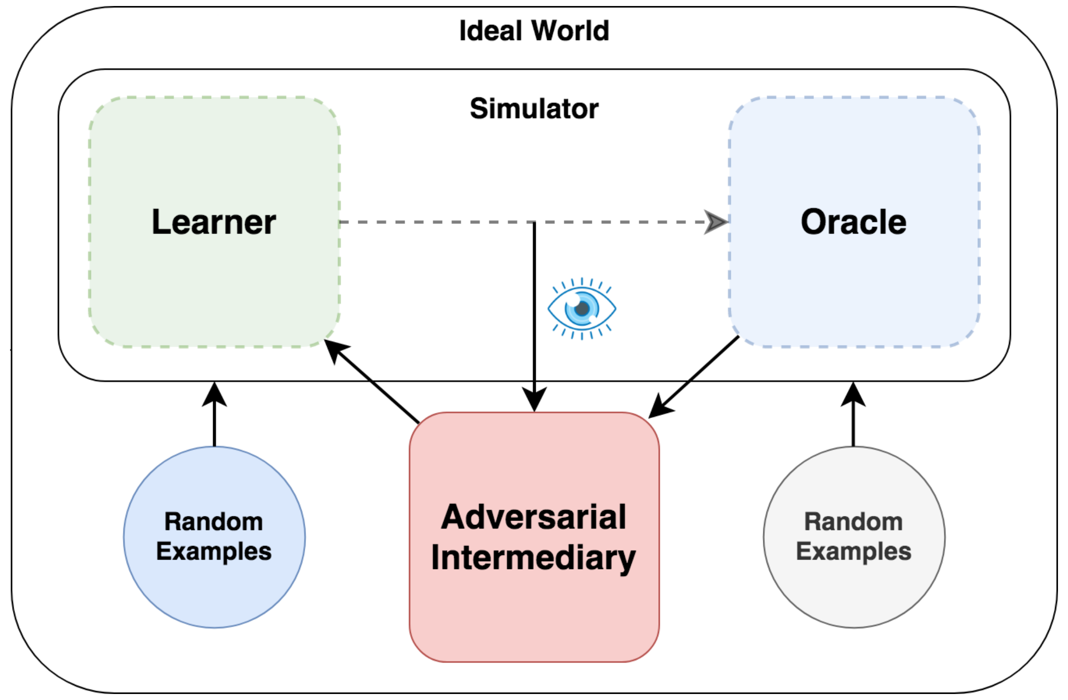
$\mathscr{S}$ is drawn from D_n

Learner gets $\mathscr{S}, H_n, \epsilon, \delta$, AI gets ϵ, δ

Learner tries to learn H_n with access to oracle. AI sees the learner's queries and responses and is given the chance to modify the responses. At the end of interaction, AI outputs a string denoted by $\text{real}_{A,I}^{\mathscr{D}_n}$

Output $\left(H_n, \epsilon, \delta, \mathscr{S}, \text{real}_{A,I}^{\mathscr{D}_n}\right)$

Ideal:



Adversary (a distinguisher) chooses
$H_n \in \mathscr{C}_n, D_n \in \mathscr{D}_n, \epsilon, \delta$

$\mathscr{S}'$ is drawn from D_n

Sim gets access to $\mathscr{S}, \mathscr{S}', H_n, \epsilon, \delta$, where $\mathscr{S}$ is the set of examples given to the **real** learner. AI gets ϵ, δ

Sim "interacts" with the oracle. AI "views" the queries and responses and may change the responses. The simulator outputs a string, ideal_I^{Sim}

Output $\left(H_n, \epsilon, \delta, \mathscr{S}, \text{ideal}_I^{Sim}\right)$

Itinerary

- Establish the Covert Learning model
- Covert Learning for noisy parities
- Covert Learning for heavy Fourier coefficients
- Establish the Covert Verifiable Learning model
- Transform Covert Learning into Covert Verifiable Learning
- Concluding questions and remarks

Augmenting Covert Learning with Verifiability

Claim Suppose that on one iteration, w.p. $1/2 + \epsilon$, the AI can corrupt at least 1 oracle response without causing the learner to abort. Then the squared-log LPN problem is efficiently solved with advantage ϵ.

— Follows from the fact that AI is always caught lying in case c = 1

- **Idea:** Alternate "testing" and "learning"
- Eventually AI is caught or one round is an uncorrupted "learning" phase
- Verifiability follows from learning guarantee of basic Covert Learning algorithm

```
Input: S = random examples

Repeat r times:
    Flip coin c
    If c = 0
        covert_learn()
    If c = 1
        Query section of S
        abort if results
        inconsistent
```

Theorem: (informal) Under the subexponential LPN assumption, the collection of all subsets of low-degree Fourier subsets is verifiably covertly learnable.

Theorem: (informal) Under the subexponential LPN assumption, the collection of all subsets of decision trees is verifiably covertly learnable.

Next Questions

Little questions

- Can we obtain more Covert Learning algorithms (e.g. new learning problems, large fields, real numbers)?

- In the verifiability setting, can we remove the assumption of privately accessed dataset? (Yes, if we relax the fully agnostic setting)

Big question

- When is there a general *compiler* for turning plain PAC learning with MQ algorithms into Covert (Verifiable) Learning algorithms?

Thank you