Learning to Reconstruct

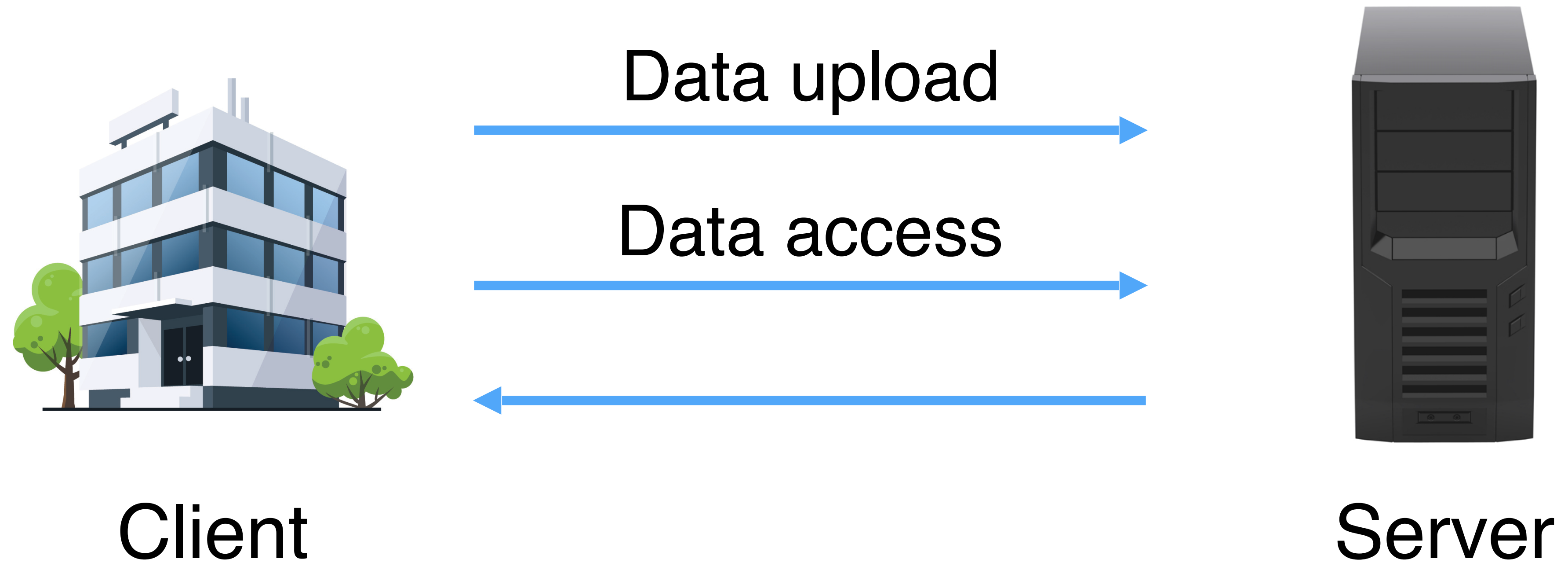
Statistical Learning Theory and Encrypted Database Attacks

Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, Kenny Paterson

eprint 2019/011 and IEEE S&P 2019.

C2 seminar, Rennes, 2019

Outsourcing Data



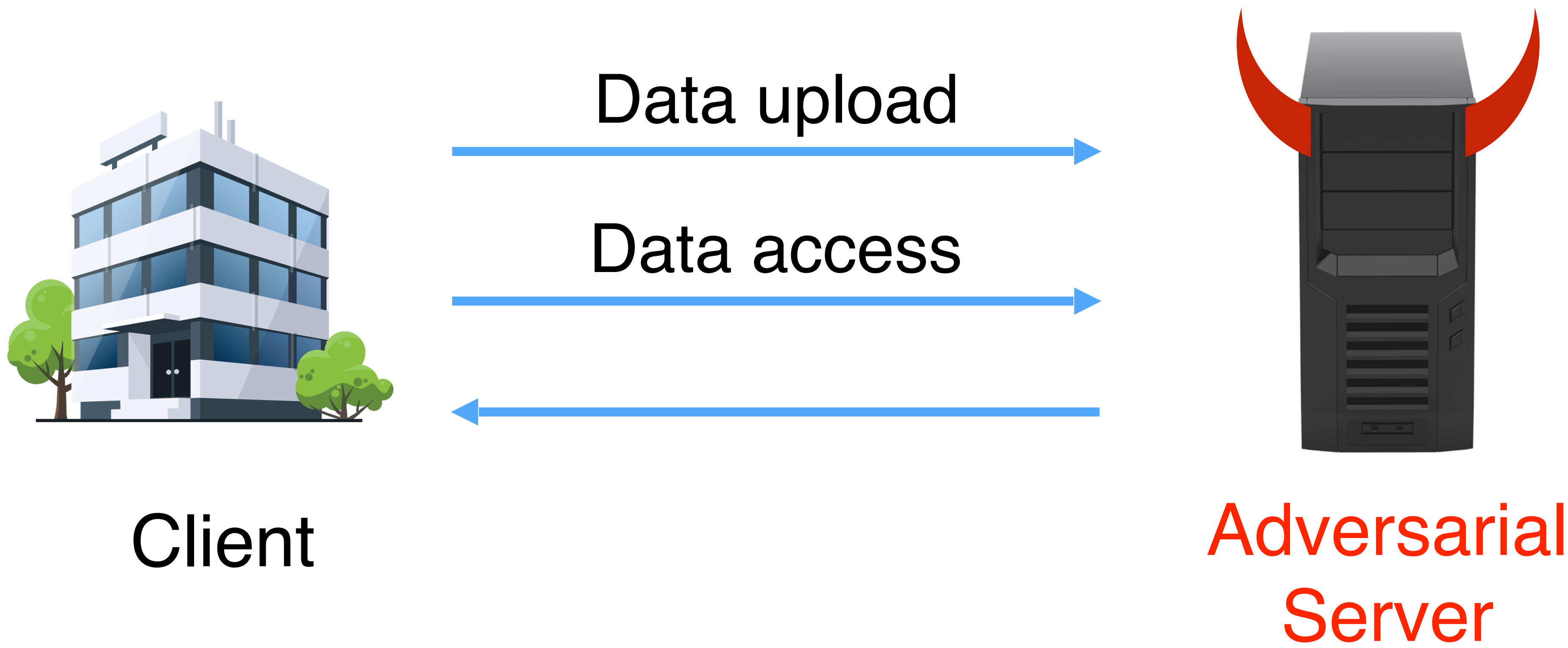
Sensitive data → encryption needed.

An **encrypted database** is of little use if it cannot be searched.

→ **Searchable Encryption**.

Examples: Private message server. Company/hospital outsourcing client/patient info.

Searchable Encryption



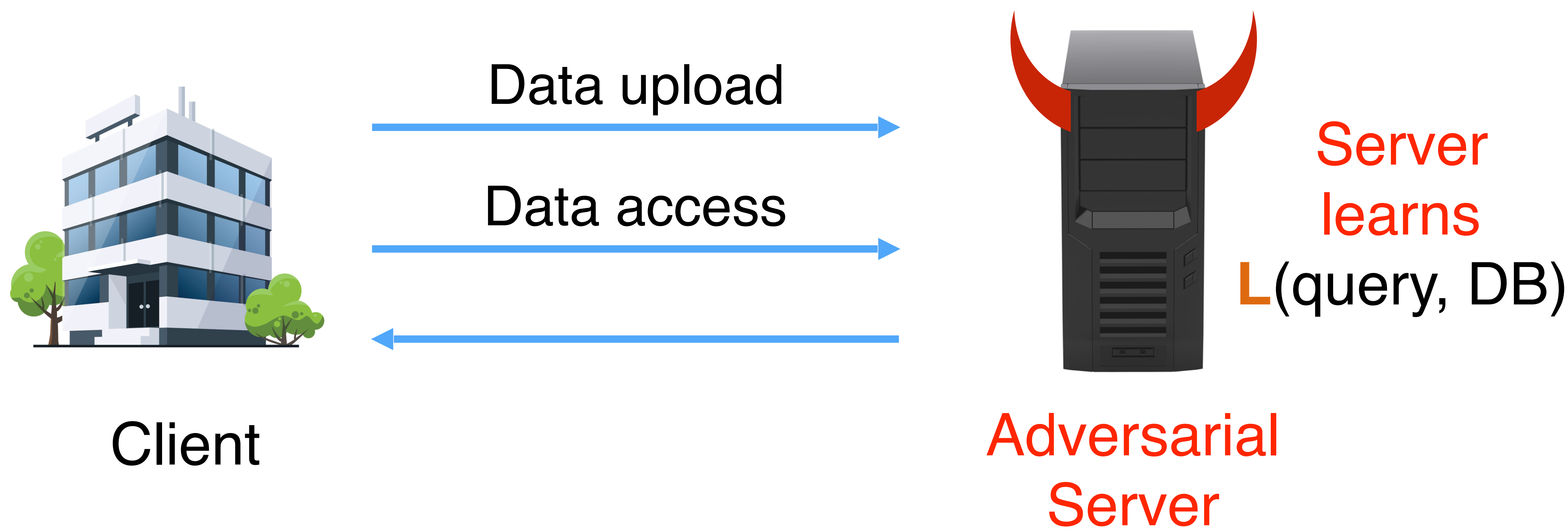
Adversary: **honest-but-curious** host server.

Security goal: **confidentiality** of **data** and **queries**.

Very active topic in research and industry.

[AKSX04], [BCLO09], [PKV+14], [BLR+15], [NKW15], [KKNO16], [LW16], [FVY+17], [SDY+17], [DP17], [HLK18], [PVC18], [MPC+18]…

Security Model



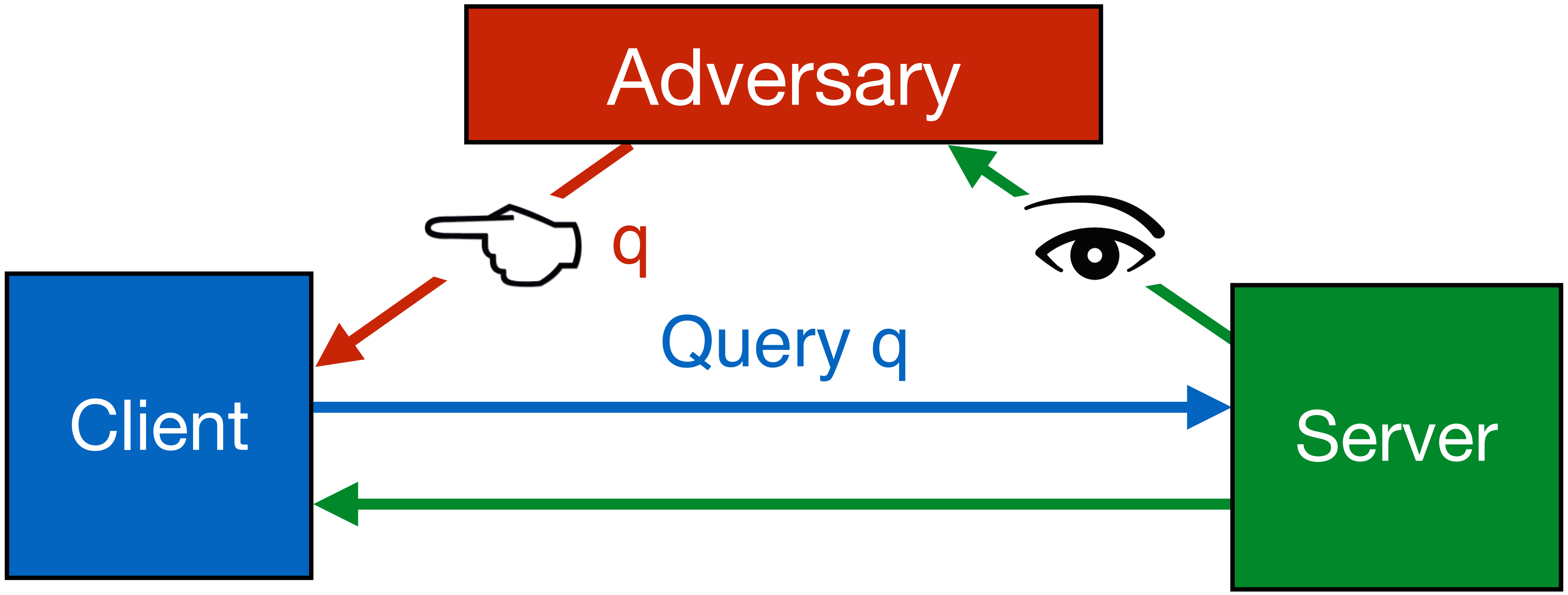
Generic solutions (FHE) are infeasible at scale → for efficiency reasons, some **leakage** is allowed.

Security model: parametrized by a **leakage function L**.

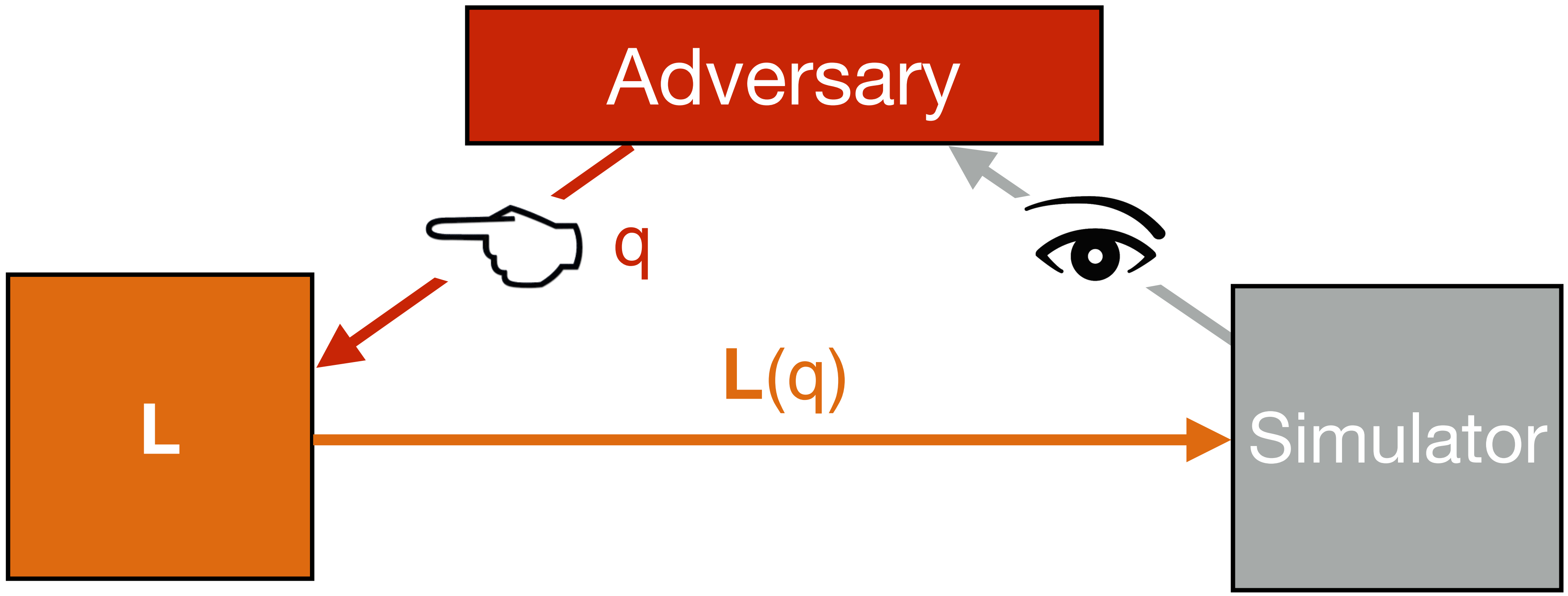
Server learns **nothing** except for the output of the leakage function.

Security Model

Real world



Ideal world



Keyword Search

Symmetric Searchable Encryption (SSE) = keyword search:

- Data = collection of documents. *e.g. messages.*
- Serch query = find documents containing given keyword(s).

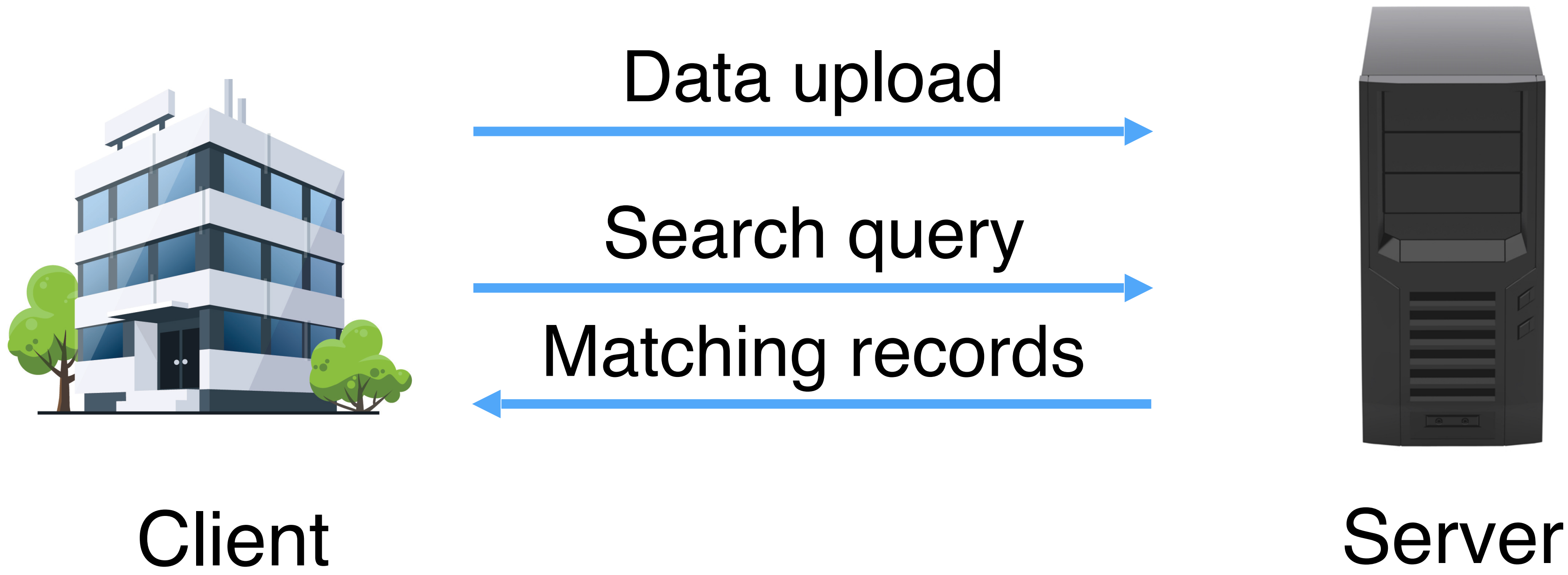
Efficient solutions for leakage = **search pattern** + **access pattern**.

Some active topics:

- **Forward** and **backward privacy** [B16][BMO17][CPPJ18][SYL+18]...

- **Locality** [CT14][ANSS16][DPP18]...

Beyond Keyword Search



For an **encrypted database management system**:

- Data = collection of records. *e.g. health records.*
- Basic query examples:
  - find records with given value. *e.g. patients aged 57.*
  - find records within a given range. *e.g. patients aged 55-65.*

Range queries

In this talk: **range queries.**

- Fundamental for any encrypted DB system.
- Many constructions out there.
- Simplest type of query that can't “just” be handled by an index.

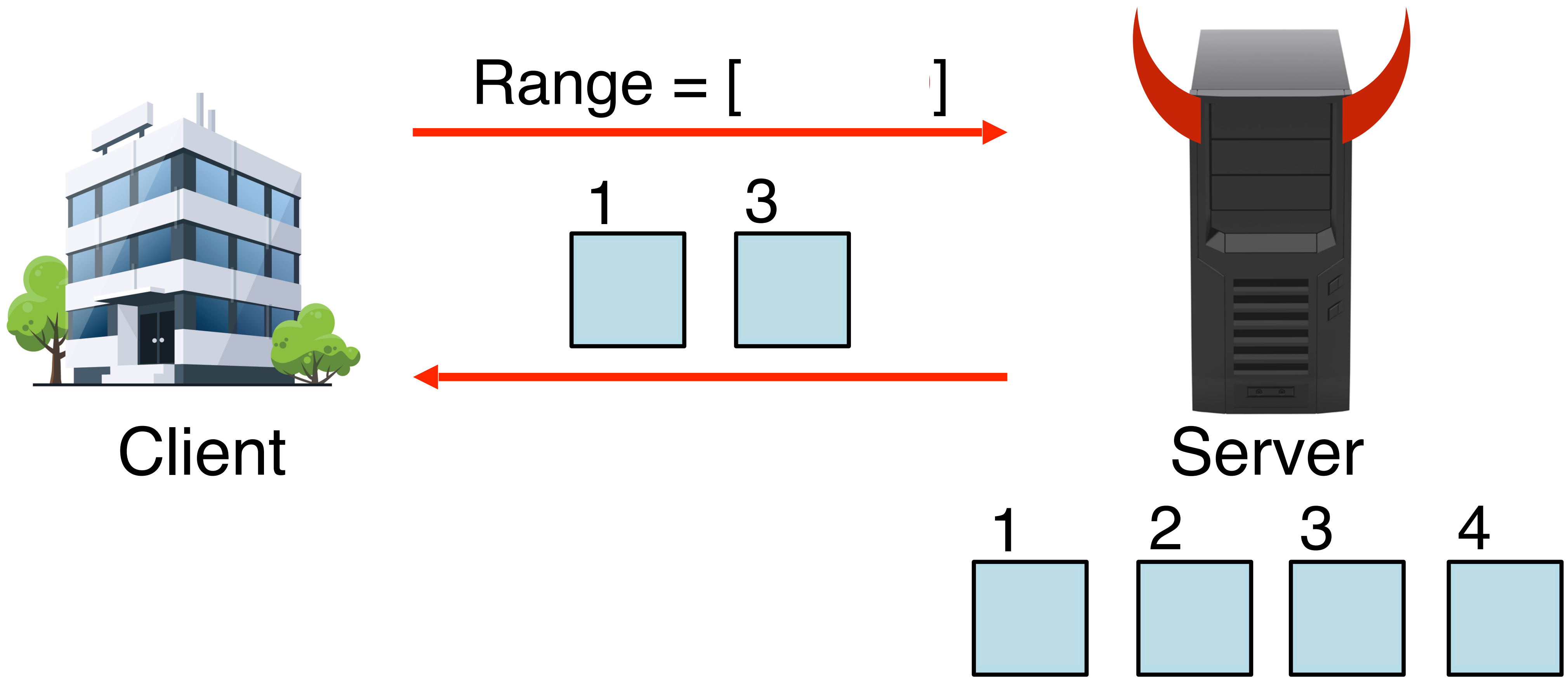
Initial solutions: **Order-Preserving, Order-Revealing Encryption**.

Leakage-abuse attacks: order information can be used to infer (approximate) values. **Leaking order is too revealing**.

→ **“Second-generation” schemes** enable range queries without relying on OPE/ORE.

Still leak **access pattern**.

Range Queries



What can the server learn from the above leakage?

Database Reconstruction

Let ***N*** = number of possible values for the target attribute.

Strongest goal: **full** database reconstruction = recovering the exact value of every record.

More general: **approximate** database reconstruction = recovering all values within εN.

$\varepsilon = 0.05$ is recovery within 5%. $\varepsilon = 1/N$ is full recovery.

(“Sacrificial” recovery: values very close to 1 and N are excluded.)

[KKNO16]: full reconstruction in $O(N^4 \log N)$ queries, assuming i.i.d. uniform queries!

Database Reconstruction

[KKNO16]: full reconstruction in $O(N^4 \log N)$ queries!

recovers

This talk ([GLMP19], [LMP18]):

	Full. Rec.	**Lower Bound**
‣ $O(\varepsilon^{-4} \log \varepsilon^{-1})$ for approx. reconstruction.	$O(N^4 \log N)$	$\Omega(\varepsilon^{-4})$
‣ $O(\varepsilon^{-2} \log \varepsilon^{-1})$ with very mild hypothesis.	$O(N^2 \log N)$	$\Omega(\varepsilon^{-2})$
‣ $O(\varepsilon^{-1} \log \varepsilon^{-1})$ for approx. *order* rec.	$O(N \log N)$	$\Omega(\varepsilon^{-1} \log \varepsilon^{-1})$

implies

Full reconstruction in $O(N \log N)$ for *dense* DBs.

Scale-free: does not depend on size of DB or number of possible values.

→ Recovering all values in DB within 5% costs O(1) queries!

Database Reconstruction

[KKNO16]: full reconstruction in $O(N^4 \log N)$ queries!

This talk ([GLMP19], subsuming [LMP18]):

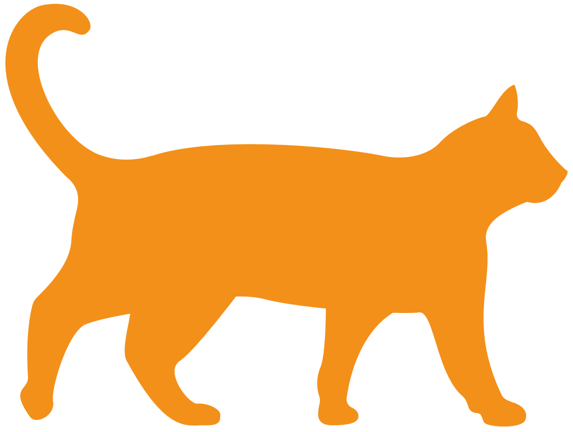
	Full. Rec.	**Lower Bound**
‣ $O(\varepsilon^{-4} \log \varepsilon^{-1})$ for approx. reconstruction.	$O(N^4 \log N)$	$\Omega(\varepsilon^{-4})$
‣ $O(\varepsilon^{-2} \log \varepsilon^{-1})$ with very mild hypothesis.	$O(N^2 \log N)$	$\Omega(\varepsilon^{-2})$
‣ $O(\varepsilon^{-1} \log \varepsilon^{-1})$ for approx. *order* rec.	$O(N \log N)$	$\Omega(\varepsilon^{-1} \log \varepsilon^{-1})$

This talk.

Main tool:

- connection with **statistical learning theory**;
- especially, **VC theory**.

VC Theory



C

VC Theory

Foundational paper: **Vapnik and Chervonenkis, 1971**.

Uniform convergence result.

Now a foundation of learning theory, especially PAC (*probably approximately correct*) learning.

Wide applicability.

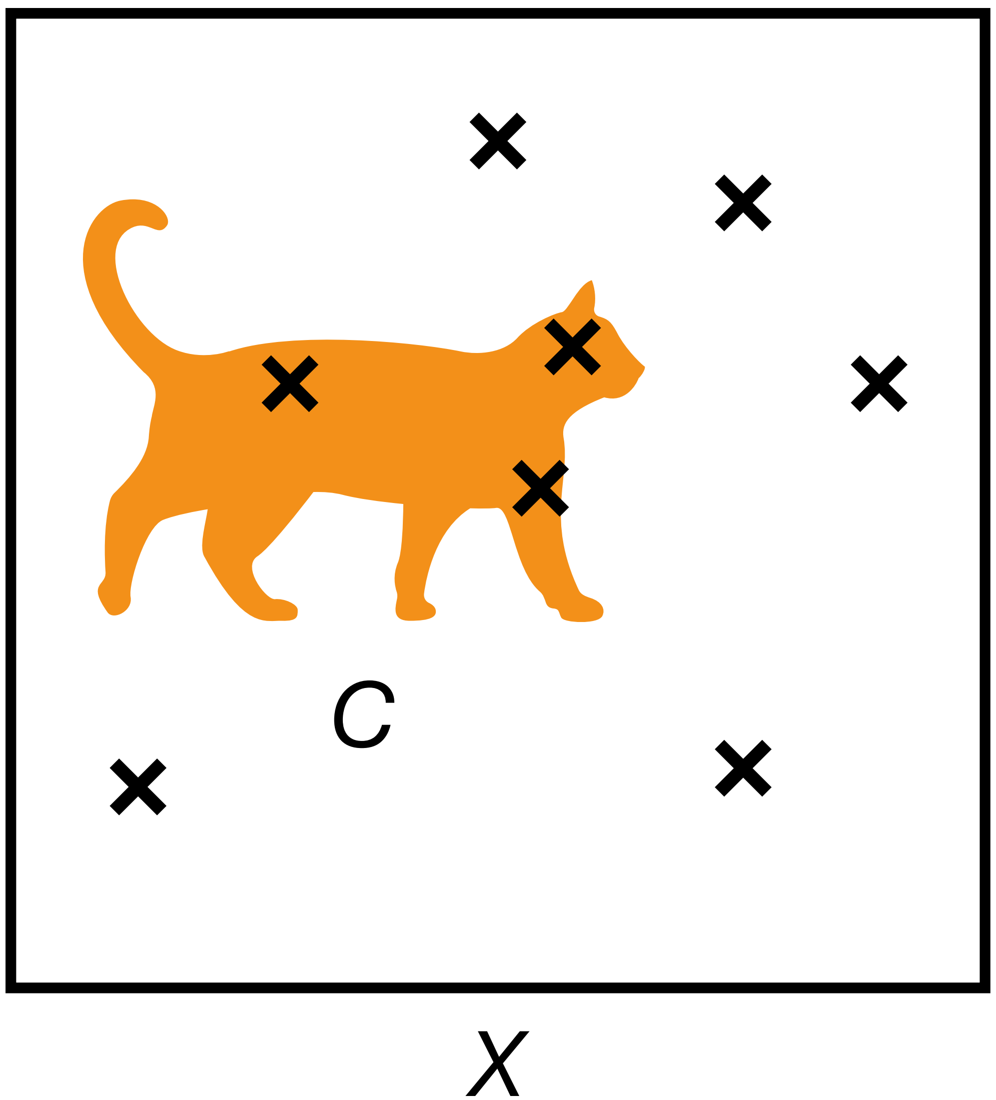
Fairly easy to state/use.

(You don't have to read the original article in Russian.)

Warm-up

Set X with probability distribution D.

Let $C \subseteq X$. Call it a *concept*.



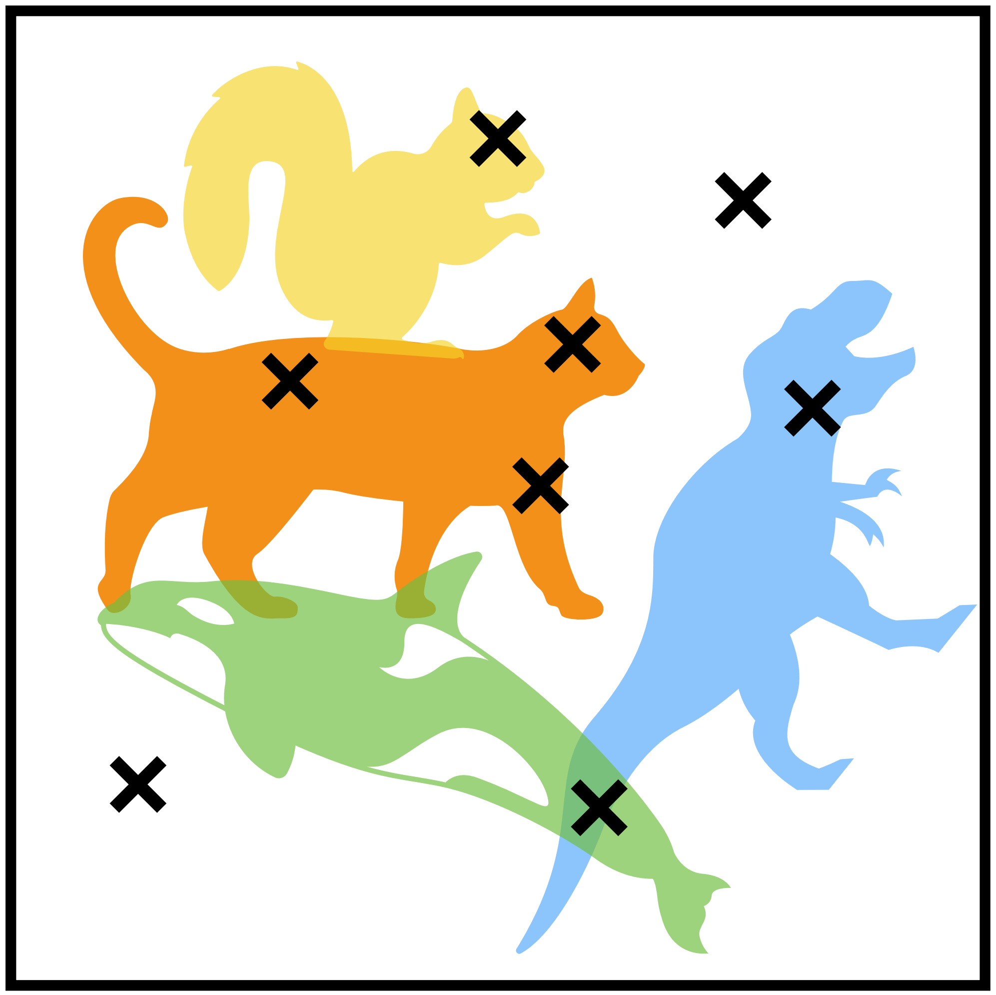
$$\Pr(C) \approx \frac{\#\text{points in } C}{\#\text{points total}}$$

Sample complexity:
to measure Pr(C) within ε,
you need O(1/ε^2) samples.

Approximating a Concept Set

Now: set $\mathscr{C}$ of concepts.

Goal: approximate their probabilities *simultaneously*.



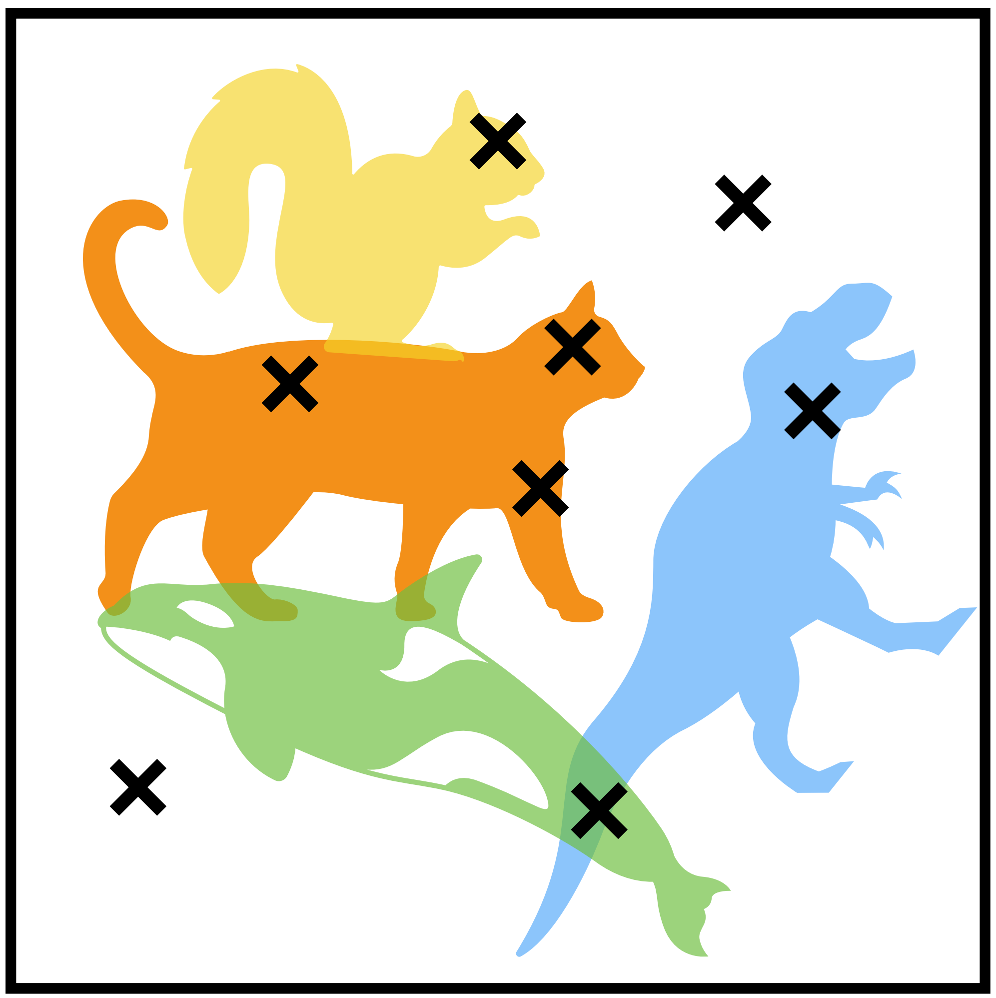
X

The set of samples drawn from X is an **ε-sample** iff for **all** C in $\mathscr{C}$:

$$\left| \Pr(C) - \frac{\#\text{points in } C}{\#\text{points total}} \right| \leq \epsilon$$

ε-sample Theorem

How many samples do we need to get an ε-sample whp?



X

Union bound: yields a sample complexity that depends on $|\mathscr{C}|$.

V & C 1971:

If $\mathscr{C}$ has **VC dimension** d, then the number of points to get an ε-sample whp is

$$O\left(\frac{d}{\epsilon^2} \log \frac{d}{\epsilon}\right).$$

Does not depend on $|\mathscr{C}|$!

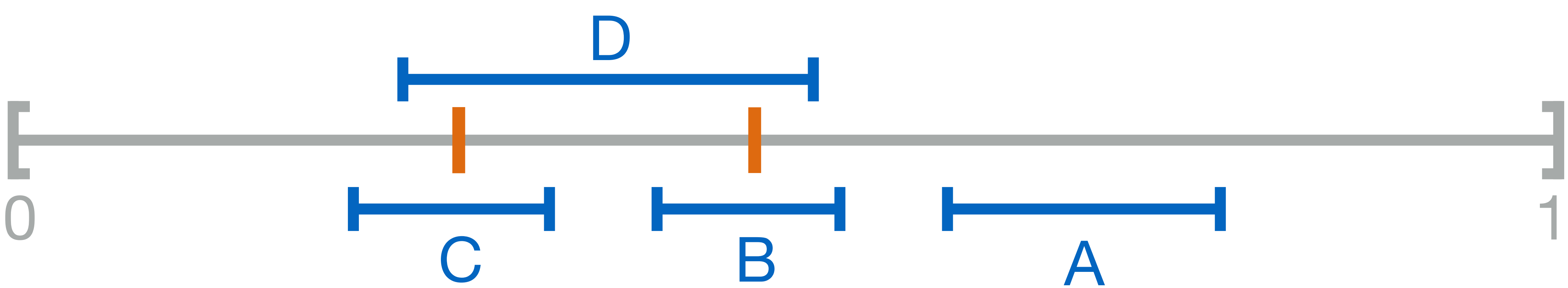
VC Dimension

Remaining Q: *what is the VC dimension?*

A set of points is **shattered** by $\mathscr{C}$ iff:

every subset of S is equal to $C \cap S$ for some C in $\mathscr{C}$.

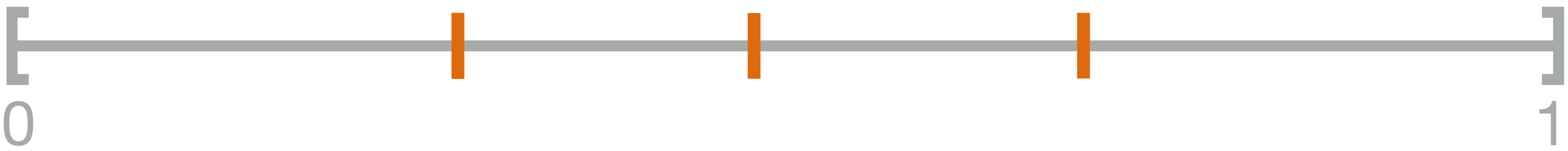
Example. Take **2 points** in X=[0,1]. Concepts $\mathscr{C}$ = all ranges.



Subsets:			
	✗	✗	**OK**. Range A.
	✗	O	**OK**. Range B.
2 points =	O	✗	**OK**. Range C.
SHATTERED	O	O	**OK**. Range D.

VC Dimension

Example. Take **3 points** in X=[0,1]. Concepts $\mathscr{C}$ = all ranges.



Subset: ○ × ○ Problem.

3 points = NOT SHATTERED

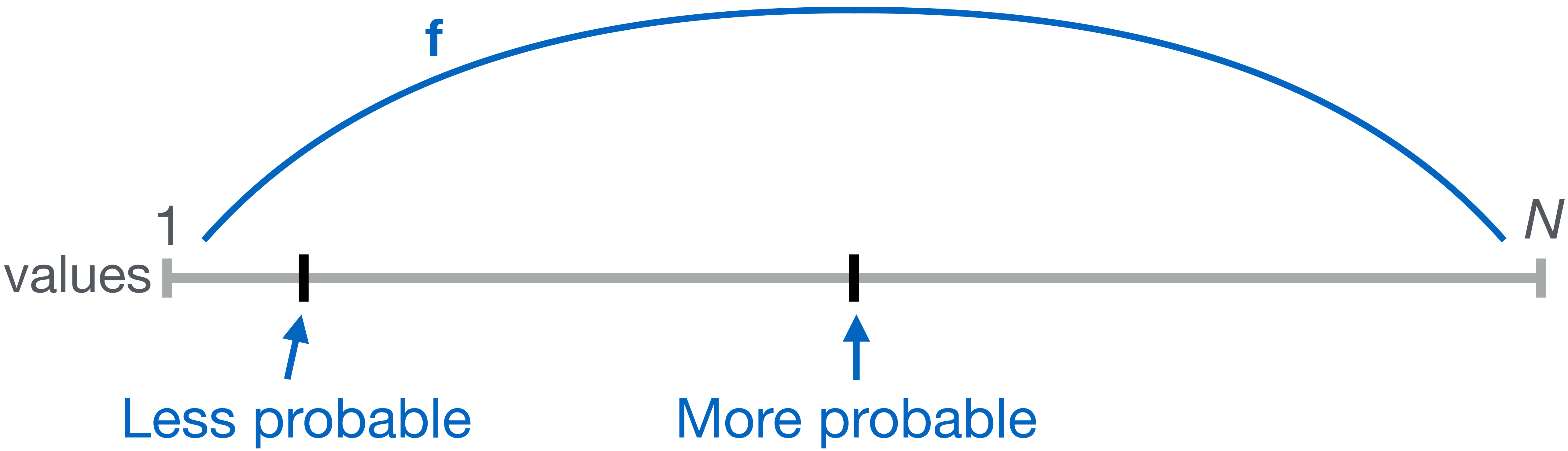
> **VC dimension** of $\mathscr{C}$ = largest cardinality of a set of points in X that is shattered by $\mathscr{C}$.

E.g. VC dimension of ranges is 2.

What typically matters is just that VC dim is finite.

Database Reconstruction

KKNO16-like Attack



Assume a **uniform distribution** on range queries.

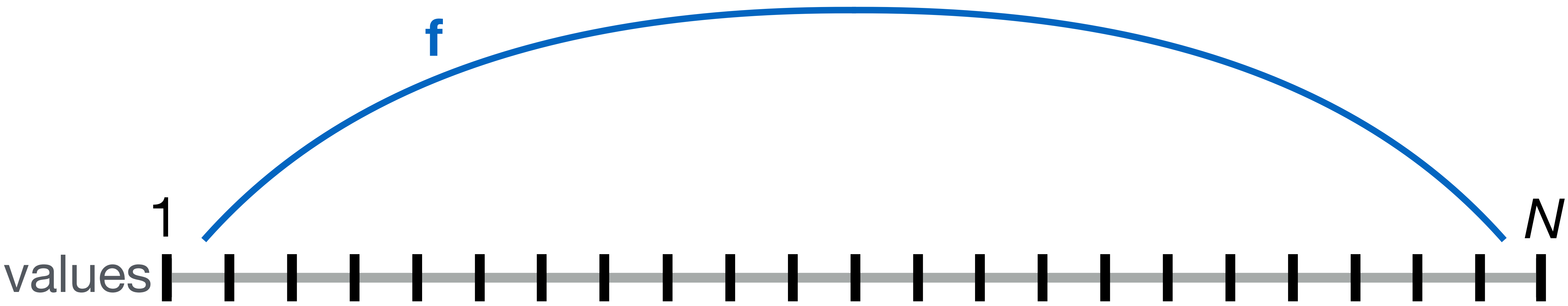
Induces a distribution **f** on the prob. that a given value is hit.

Idea: for each record...

1. Count frequency at which the record is hit.
→ gives estimate of probability it's hit by uniform query.

2. deduce estimate of its value by "inverting" **f**.

KKNO16-like Attack



Step 1: for **all** records, estimate prob of the record being hit.

This is an ε-sample!

X = ranges $\mathscr{C}$ =$\{\{$ranges $\ni x\}: x \in [1,N]\}$

so we need $O(\varepsilon^{-2} \log \varepsilon^{-1})$ queries.

Step 2: because **f** is quadratic, “inverting” **f** adds a square.

After $O(\varepsilon^{-4} \log \varepsilon^{-1})$ queries, the value of all records is recovered within εN.

On the i.i.d. Assumption

We are assuming **uniformly distributed** queries.

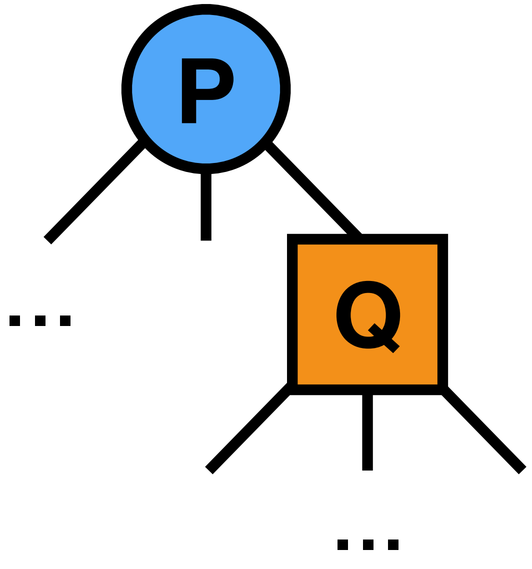
In reality we are assuming:

- The advesary knows the query distribution.
- Queries are uniform.
- More fundamentally, queries are **independent and identically distributed** (i.i.d.).

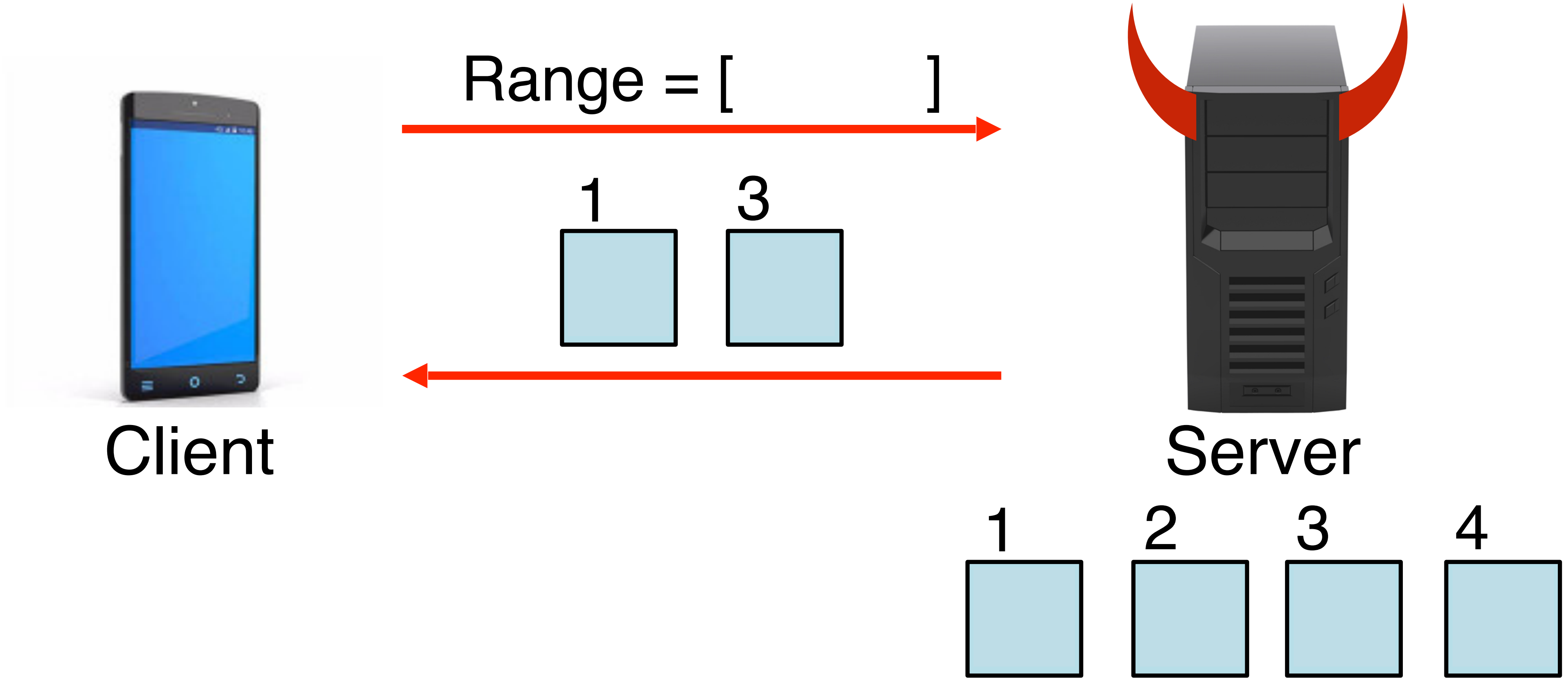
This is not realistic.

What can we learn without that hypothesis?

Order Reconstruction



Problem Statement



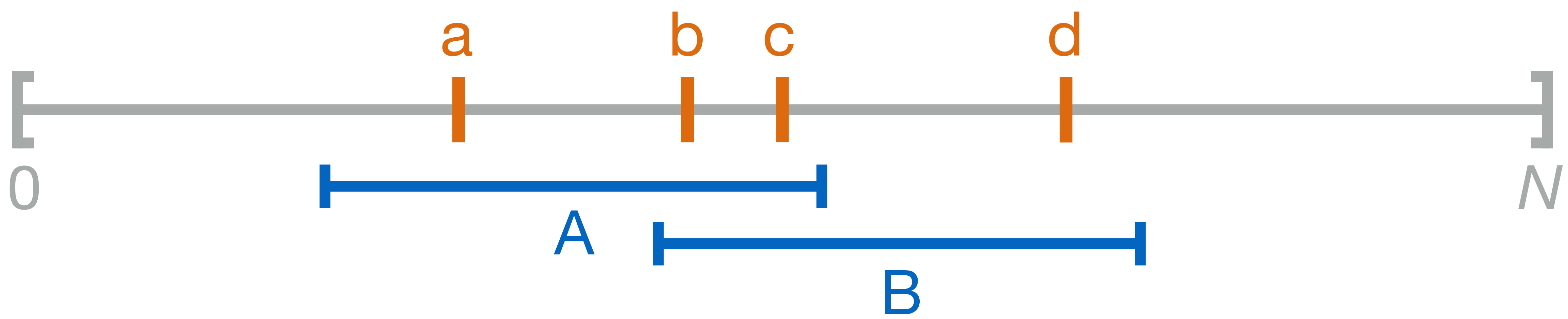
What can the server learn from the above leakage?

This time we **don't assume** i.i.d. queries, or knowledge of their distribution.

Range Query Leakage

Query A matches records a, b, c.

Query B matches records b, c, d.



Then this is the only configuration (up to symmetry)!

→ we learn that records b, c are *between* a and d.

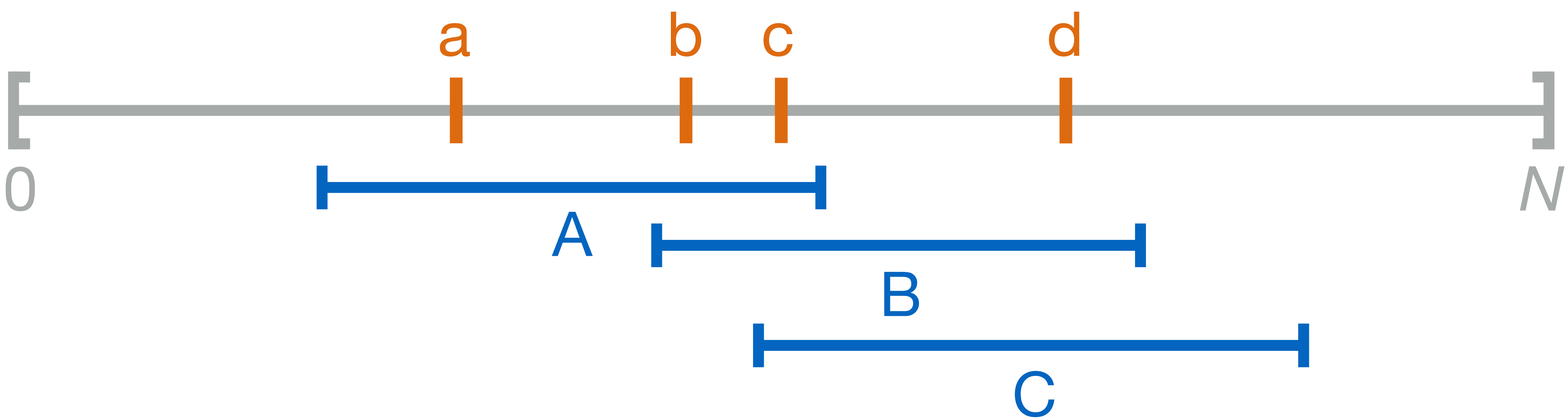
We learn something about the **order** of records.

Range Query Leakage

Query A matches records a, b, c.

Query B matches records b, c, d.

Query C matches records c, d.



Then the only possible order is a, b, c, d (or d, c, b, a)!

Challenges:

- How do we extract order information? (What **algorithm**?)
- How do we **quantify** and **analyze** how fast order is learned as more queries are observed?

Challenge 1: the Algorithm

Short answer: there is already an algorithm!

Long answer: **PQ-trees**.

X: linearly ordered set. Order is unknown.

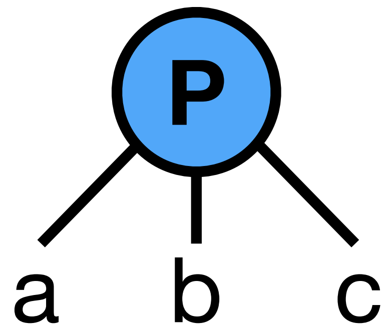
You are given a set S containing some intervals in X.

> A **PQ tree** is a compact (linear in $|X|$) representation of the set of all permutations of X that are compatible with S.

Can be updated in linear time.

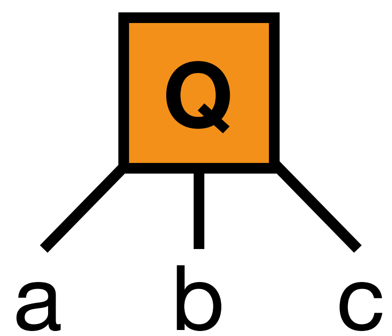
Note: was used in [DR13], didn't target reconstruction.

PQ Trees



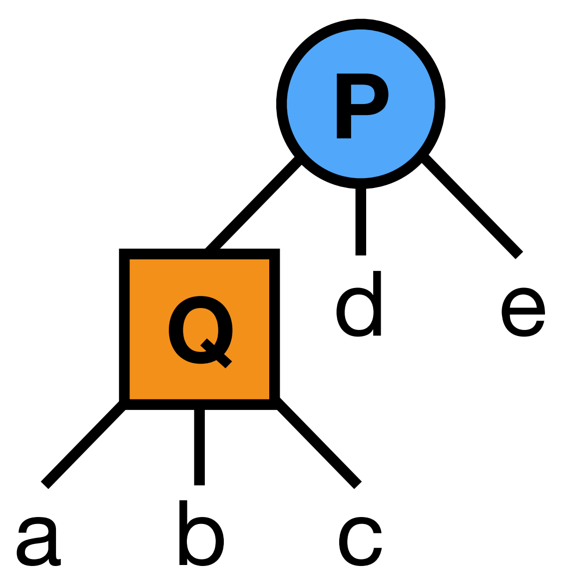
Order is completely **unknown**.

- any permutation of abc.



Order is completely **known** (up to reflection).

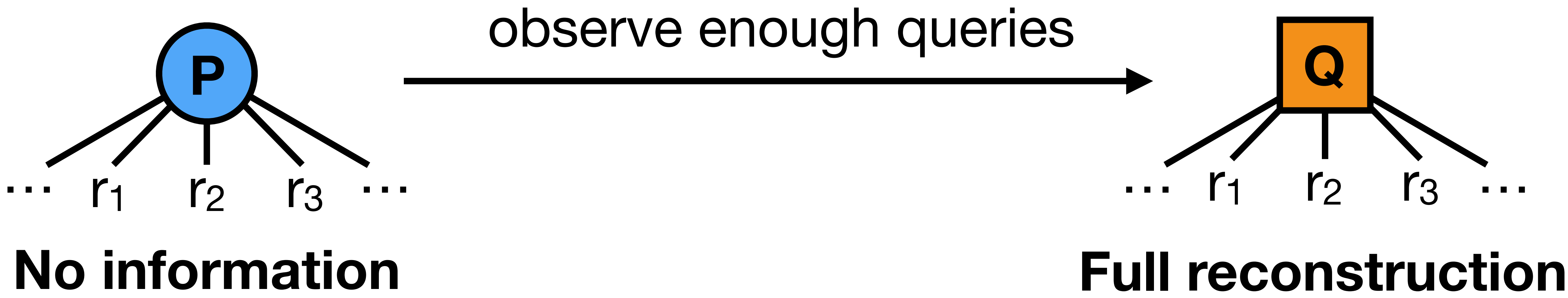
- abc'or 'cba'.



Combines in the natural way.

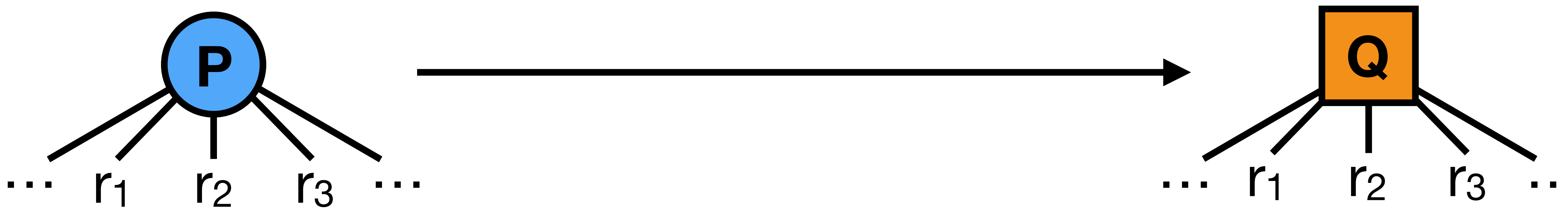
- 'abcde', 'abced', 'dabce', 'eabcd', 'deabc', 'edabc', 'cbade' etc.

Full Order Reconstruction



We want to **quantify** order learning...

Challenge 2a: Quantify Order Learning

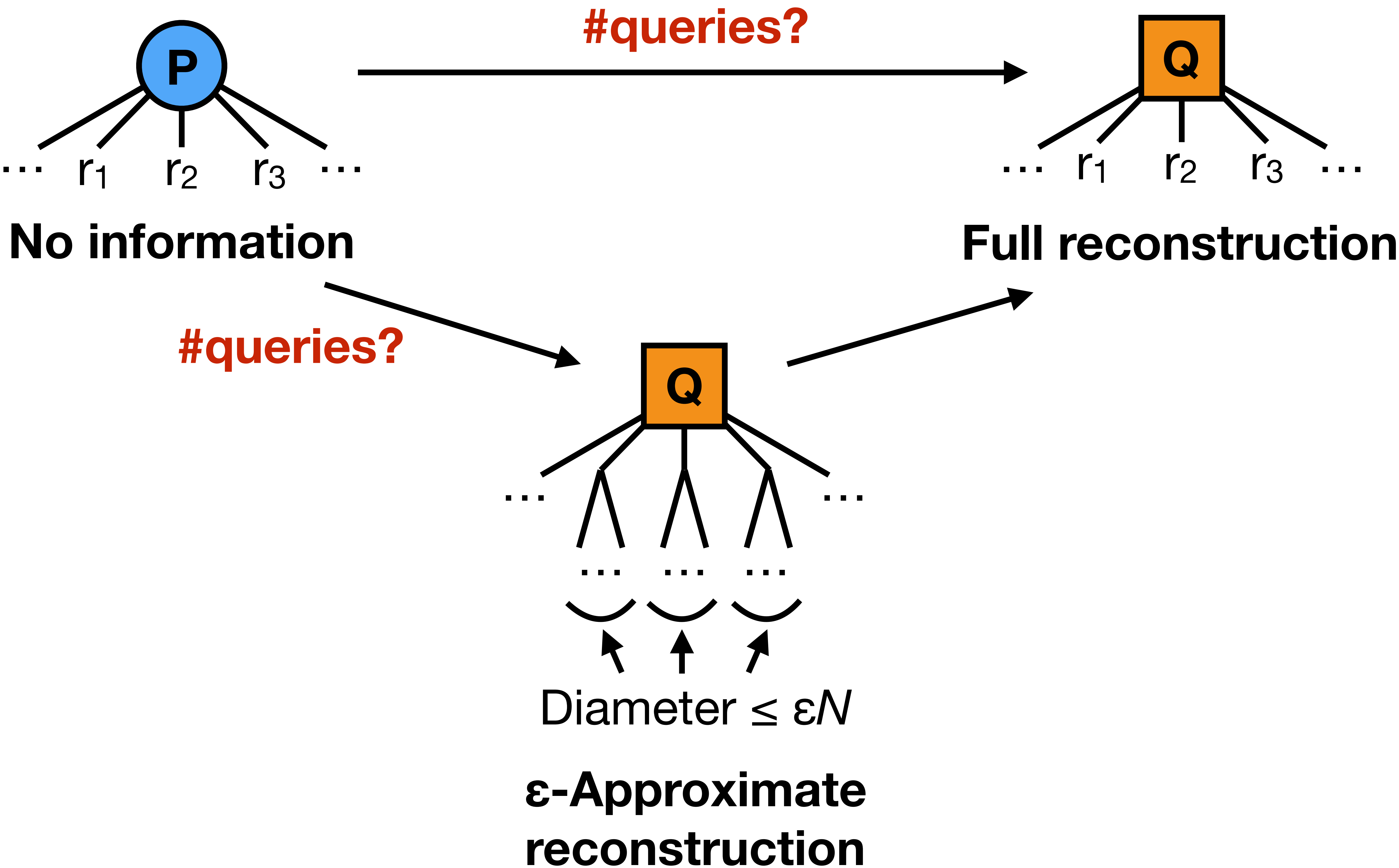


No information **Full reconstruction**

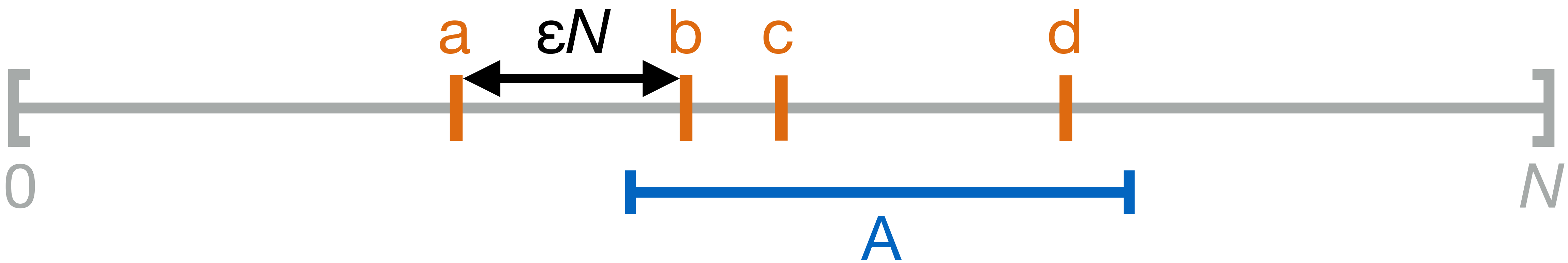
ε-Approximate order reconstruction.

Roughly: we learn the order between two records as soon as their values are $\geq \varepsilon N$ apart. ($\varepsilon = 1/N$ is full reconstruction)

Approximate Order Reconstruction



Challenge 2b: Analyze Query Complexity



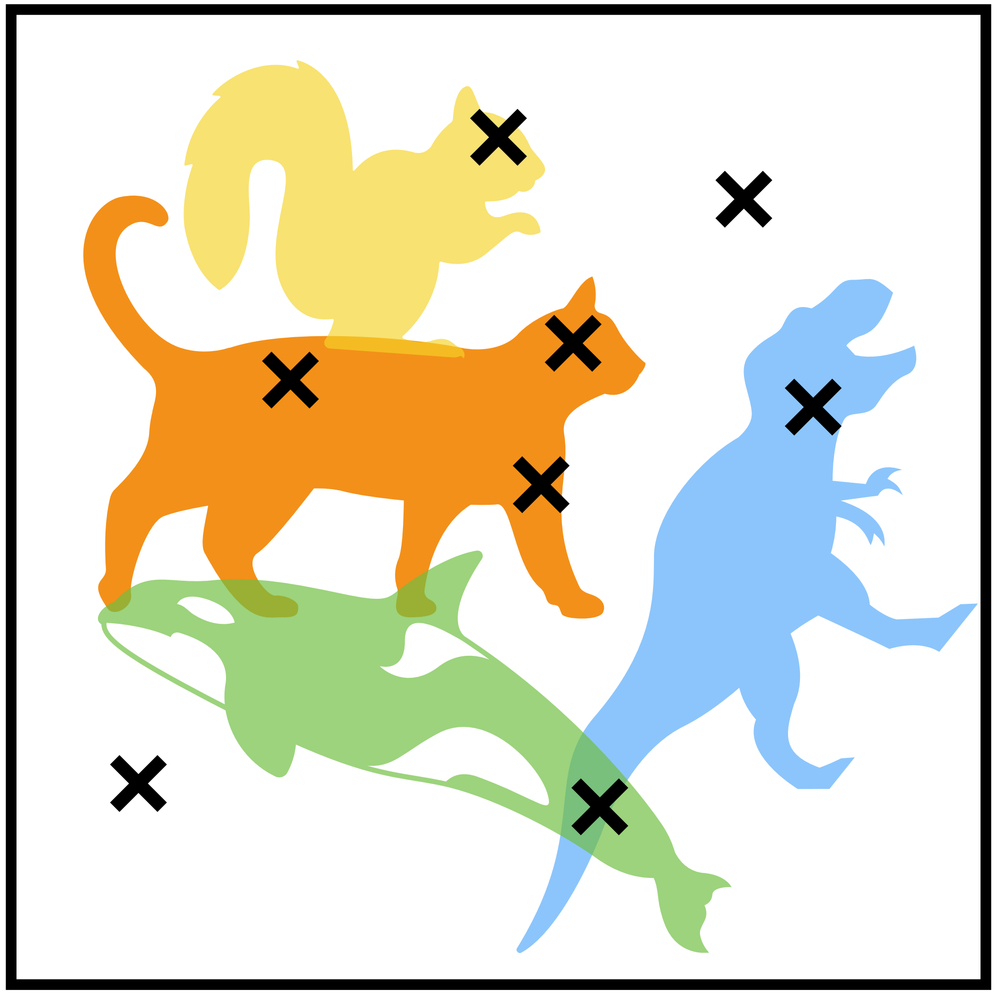
Intuition: if no query has an endpoint between a and b, then a and b can't be separated.

→ ε-approximate reconstruction is impossible.

You want a query endpoint to hit every interval ≥ ε*N*.
Conversely with some other conditions it's enough.

Heavy sweeping of details under rug.

VC Theory Saves the Day (again)



ε-samples: the ratio of points hitting each concept is close to its probability.

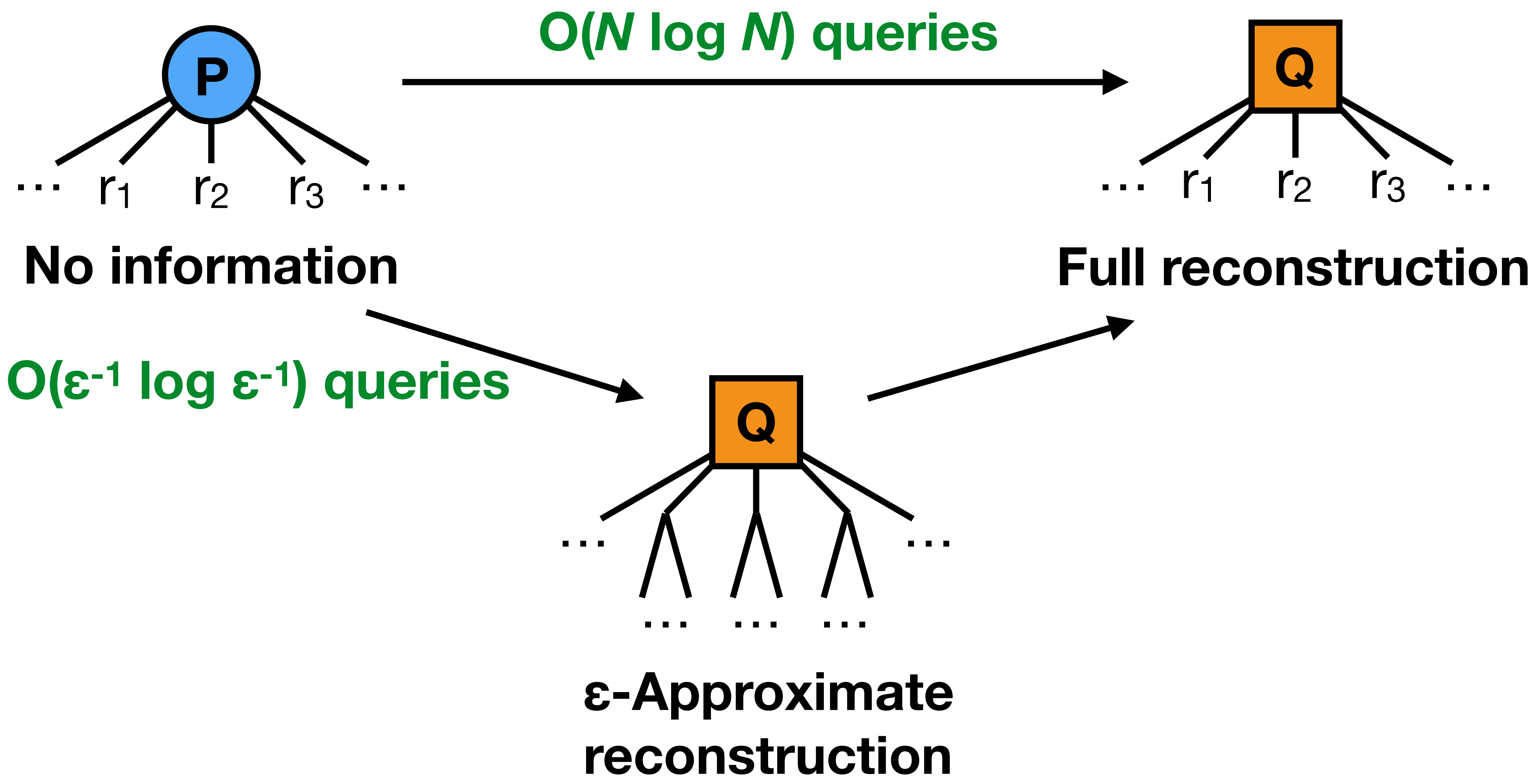
What we want now: if a concept has high enough probability, it is hit by at least one point.

The set of samples drawn from X is an **ε-net** iff for all C in $\mathscr{C}$:

$$\Pr(C) \geq \epsilon \Rightarrow C \text{ contains a sample}$$

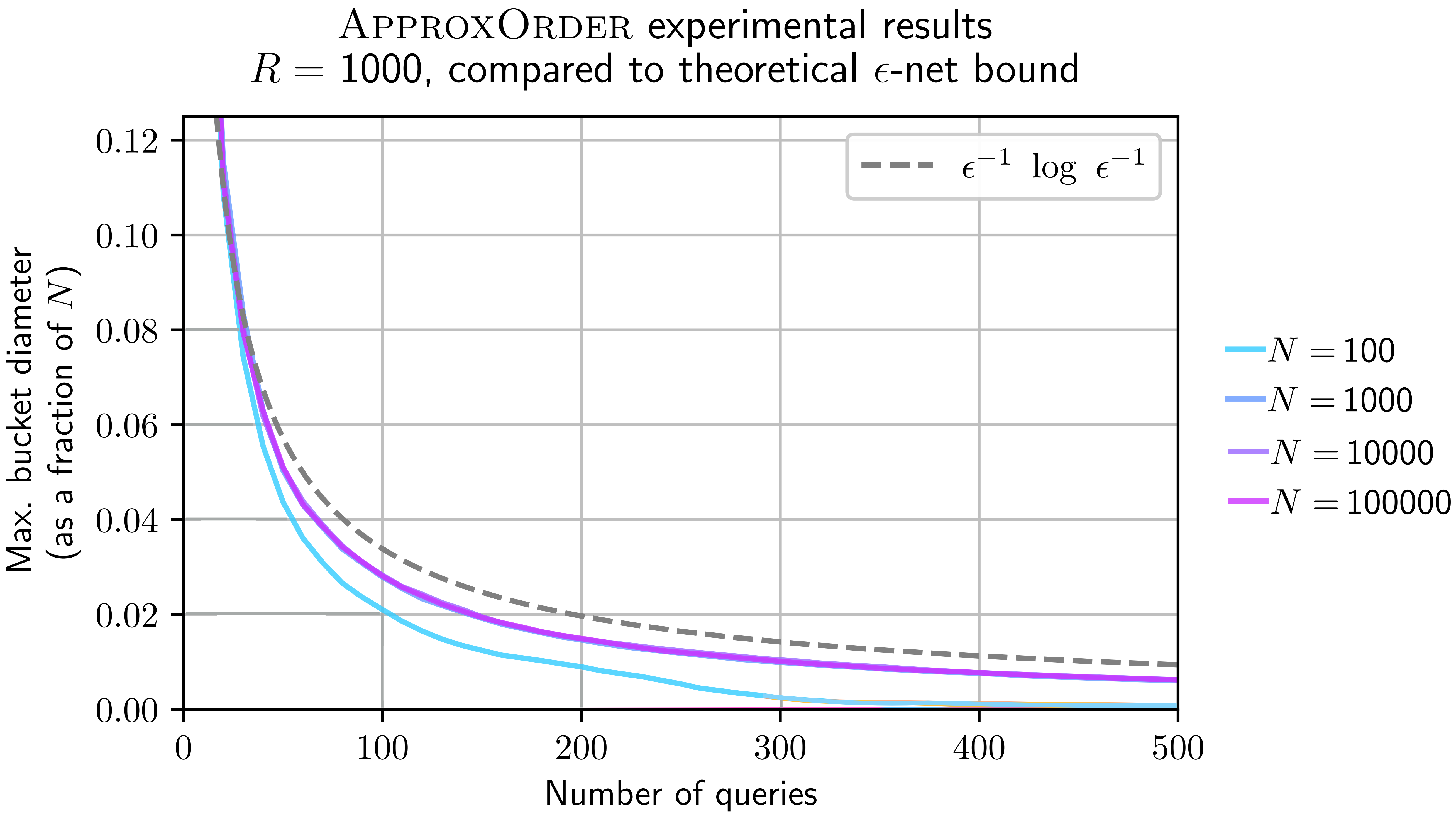
→ Number of points to get an ε-net whp: $O\left(\frac{d}{\epsilon}\log\frac{d}{\epsilon}\right)$

Approximate Order Reconstruction



Note: some (weak) assumptions are swept under the rug.

Experiments



Closing Remarks

On Range Queries

Severe attacks under minimal assumptions.

Analysis clarifies setting.

- Size of DB, or number of possible values, don't matter.
- What is really leaked is order of records.
- Various auxiliary info can get you from order to values.

Please don't use OPE/ORE.

Also avoid current encrypted DBs if you don't trust the server and care about privacy.

New solutions needed. E.g. efficient specialized ORAMs.

Connection to Machine Learning

- In this talk: VC theory.
- In the article: known query setting = PAC learning.
- Some results for general query classes.

Machine learning in crypto: also used for **side channel attacks**. Same general setting!

Natural connection between **reconstructing secret information from leakage** and **machine learning**.

Seems to be a powerful tool to understand the security implications of leakage. **In side channels** - use learning algorithms; **here** - use learning theory.