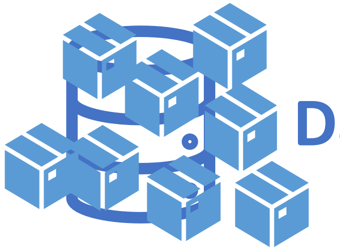
Single-Server Private Information Retrieval in the Shuffle Model

Yuval Ishai Mahimna Kelkar Daniel Lee Yiping Ma

Technion Israel Institute of Technology
Cornell Tech
Penn University of Pennsylvania

Private Information Retrieval (PIR) [CGKS95, KO97]



Database server has $x = \{0,1\}^n$

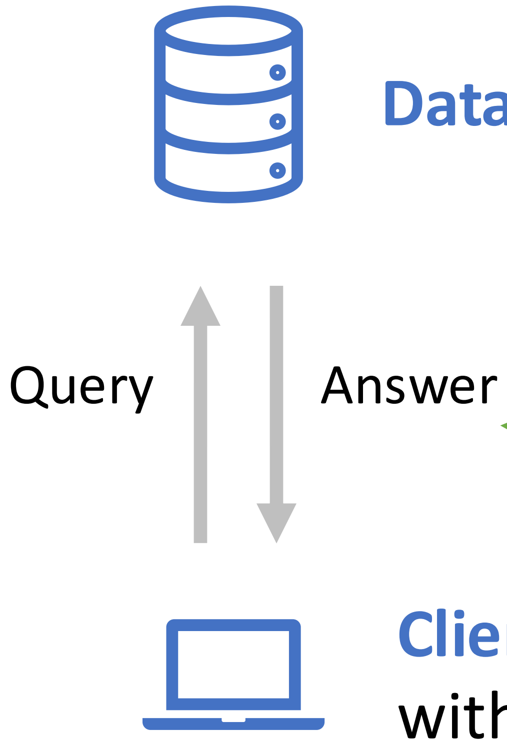
Query

Answer

A trivial solution:
download all entries

Client wants to get the entry x_i
without revealing the index i

Private Information Retrieval (PIR) [CGKS95, KO97]



Database server has $x = \{0,1\}^n$

Solutions that we are interested in:
communication sublinear in n

Client wants to get the entry x_i
without revealing the index i

PIR in two flavors

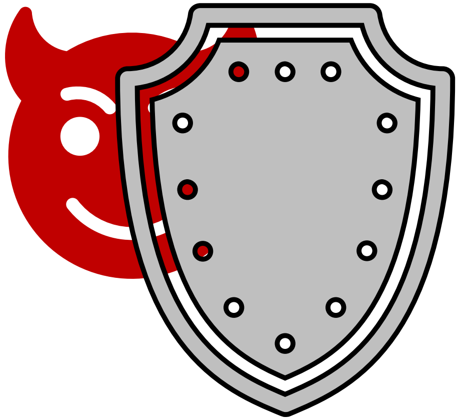
Information-theoretic

Computational

PIR in two flavors

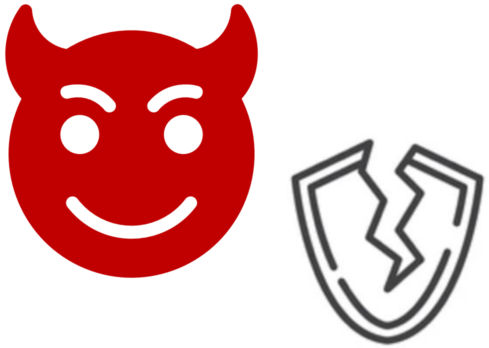
Information-theoretic

- Secure against unbounded adversaries



Computational

- Secure against polynomial-time adversaries

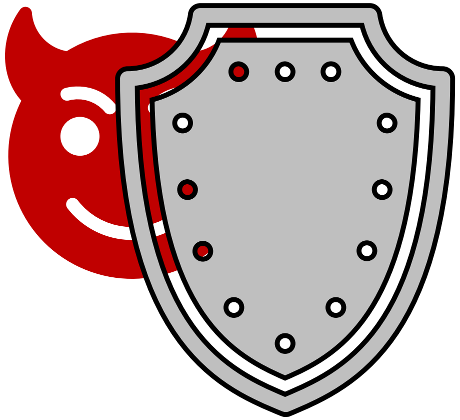


A weaker security notion

PIR in two flavors

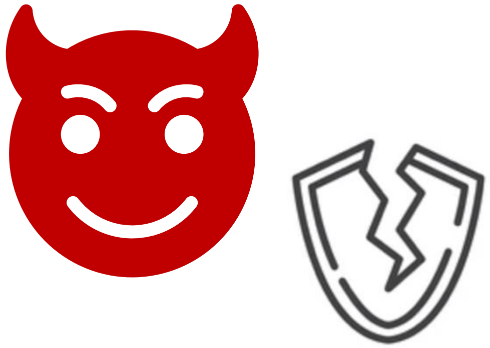
Information-theoretic

- Secure against unbounded adversaries



Computational

- Secure against polynomial-time adversaries



A weaker security notion

PIR in two flavors

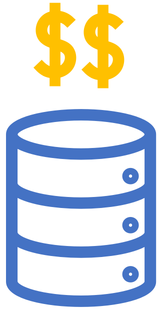
Information-theoretic

- Secure against unbounded adversaries
- Require database replication across multiple servers

Computational

- Secure against polynomial-time adversaries
- No database replication, a single server suffices

Managing multiple storage spots has high cost when databases are large



PIR in two flavors

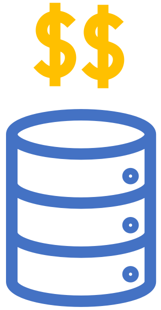
Information-theoretic

- Secure against unbounded adversaries
- Require database replication across multiple servers

Computational

- Secure against polynomial-time adversaries
- No database replication, a single server suffices

Managing multiple storage spots has high cost when databases are large



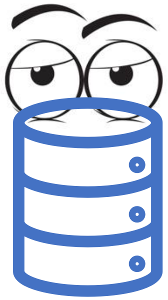
PIR in two flavors

Information-theoretic

- Secure against unbounded adversaries
- Require database replication across multiple servers
- Enforce non-collusion amongst the database servers

Computational

- Secure against polynomial-time adversaries
- No database replication, a single server suffices
- No need for non-colluding assumption on the database server



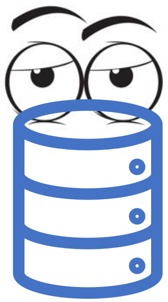
PIR in two flavors

Information-theoretic

- Secure against unbounded adversaries
- Require database replication across multiple servers
- Enforce non-collusion amongst the database servers

Computational

- Secure against polynomial-time adversaries
- No database replication, a single server suffices
- No need for non-colluding assumption on the database server



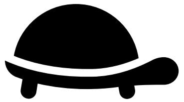
PIR in two flavors

Information-theoretic

- Secure against unbounded adversaries
- Require database replication across multiple servers
- Enforce non-collusion amongst the database servers
- Efficient in practice (no cryptographic operations)

Computational

- Secure against polynomial-time adversaries
- No database replication, a single server suffices
- No need for non-colluding assumption on the database server
- Expensive server cost because of cryptogaphic operations



Hard to scale to many clients

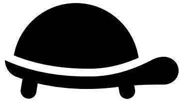
PIR in two flavors

Information-theoretic

- Secure against unbounded adversaries
- Require database replication across multiple servers
- Enforce non-collusion amongst the database servers
- Efficient in practice (no cryptographic operations)

Computational

- Secure against polynomial-time adversaries
- No database replication, a single server suffices
- No need for non-colluding assumption on the database server
- Expensive server cost because of cryptogaphic operations



Hard to scale to many clients

PIR in two flavors

Information-theoretic

- Secure against unbounded adversaries
- Require database replication across multiple servers
- Enforce non-collusion amongst the database servers
- Efficient in practice (no cryptographic operations)
- Schemes with short query size enable efficient preprocessing => sublinear server computation

Computational

- Secure against polynomial-time adversaries
- No database replication, a single server suffices
- No need for non-colluding assumption on the database server
- Expensive server cost because of cryptogaphic operations
- Query size depends on the computational security parameter
  - No "trivial" solution for efficient preprocessing
  - Exists efficient preprocessing in non-trivial ways

PIR in two flavors

Information-theoretic

- Secure against unbounded adversaries
- Require database replication across multiple servers
- Enforce non-collusion amongst the database servers
- Efficient in practice (no cryptographic operations)
- Schemes with short query siz
  efficient preprocessing => su
  server computation

Computational

- Secure against polynomial-time adversaries
- No database replication, a single server suffices
- No need for non-colluding assumption on the database server
- Expensive server cost because of
  nal
  sing
  - Exists efficient preprocessing in non-trivial ways

Existing single-server solutions with sublinear computation:
Either require per-client preprocessing [CHK22]; or utilize strong assumptions + VBB obfuscations [BIPW17, CHR17]

Best of both worlds?

Information-theoretic

- Secure against unbounded adversaries
- Require database replication across multiple servers
- Enforce non-collusion amongst the database servers
- Efficient in practice (no cryptographic operations)
- Schemes with short query size enable efficient preprocessing => sublinear server computation

Computational

- Secure against polynomial-time adversaries
- No database replication, a single server suffices
- No need for non-colluding assumption on the database server
- Expensive server cost because of cryptogaphic operations
- Query size depends on the computational security parameter
  - No "trivial" solution for efficient preprocessing
  - Exists efficient preprocessing in non-trivial ways

Best of both worlds?

- Security must hold for even **a single client**
  The only way out—requires n bits communication

"The standard model"

- New hope: relaxation by considering **multiple clients**

The shuffle model [IKOS06]
Component 1: Many clients make queries simultaneously
Component 2: The queries are shuffled before reaching the server

Best of both worlds? Yes, in the shuffle model

- Security must hold for even **a single client**
  The only way out—requires n bits communication

"The standard model"

- New hope: relaxation by considering **multiple clients**

The shuffle model [IKOS06, BEMM+17, BBGN20, …]

Component 1: Many cli… …s simultaneously

…efore reaching the server

- Construction based on a specific PIR protocol
- Nonstandard computational assumption

Best of both worlds? Yes, in the shuffle model

- Security must hold for even **a single client**
  The only way out—requires n bits communication

"The standard model"

- New hope: relaxation by considering **multiple clients**

This work: general constructions for single-server PIR in the shuffle model that has information-theoretic security and sublinear communication

Best of both worlds? Yes, in the shuffle model

- Security must hold for even **a single client**
  The only way out—requires n bits communication

"The standard model"

- New hope: relaxation by considering **multiple clients**

Theorem (Informal).
For every $\gamma > 0$, there is a single-server PIR in the shuffle model such that, on database size n, has $O(n^{\gamma})$ per-query communication and $1/\text{poly}(n)$ statistical security, assuming $\text{poly}(n)$ clients simultaneously accessing the database.
If further assuming one-time preprocessing, per-query computation is also $O(n^{\gamma})$.

Throughout this talk, we omit polylog n factors.

Best of both worlds? Yes, in the shuffle model

- Security must hold for even **a single client**
  The only way out—requires n bits communication

"The standard model"

- New hope: relaxation by considering **multiple clients**

Theorem (Informal).
For every $\gamma > 0$, there is a single-server PIR in the shuffle model such that, on database size n, has $O(n^\gamma)$ per-query communication and $1/\mathrm{poly}(n)$ statistical security, assuming $\mathrm{poly}(n)$ clients simultaneously accessing the database.
If further assuming one-time preprocessing, per-query computation is also $O(n^\gamma)$.

Throughout this talk, we omit $\mathrm{polylog}\, n$ factors.

Rest of this talk

- Background
  - The shuffle model
  - “Split and mix”
- Our results
  - General constructions
  - Lower bound: the security we get in the general constructions is “tight”
  - An interesting orthogonal problem: hiding record size without padding
- Discussion and open questions

The shuffle model

- Purpose: **anonymization**
- An existing notion in many literatures
  - Anonymous communication, e.g., [HLZZ15]
  - Differential privacy, e.g., [BBGN20]
  - Secure aggregation, e.g., [IKOS06]
- In our setting:
  assume a two-way anonymous channel

A shuffler

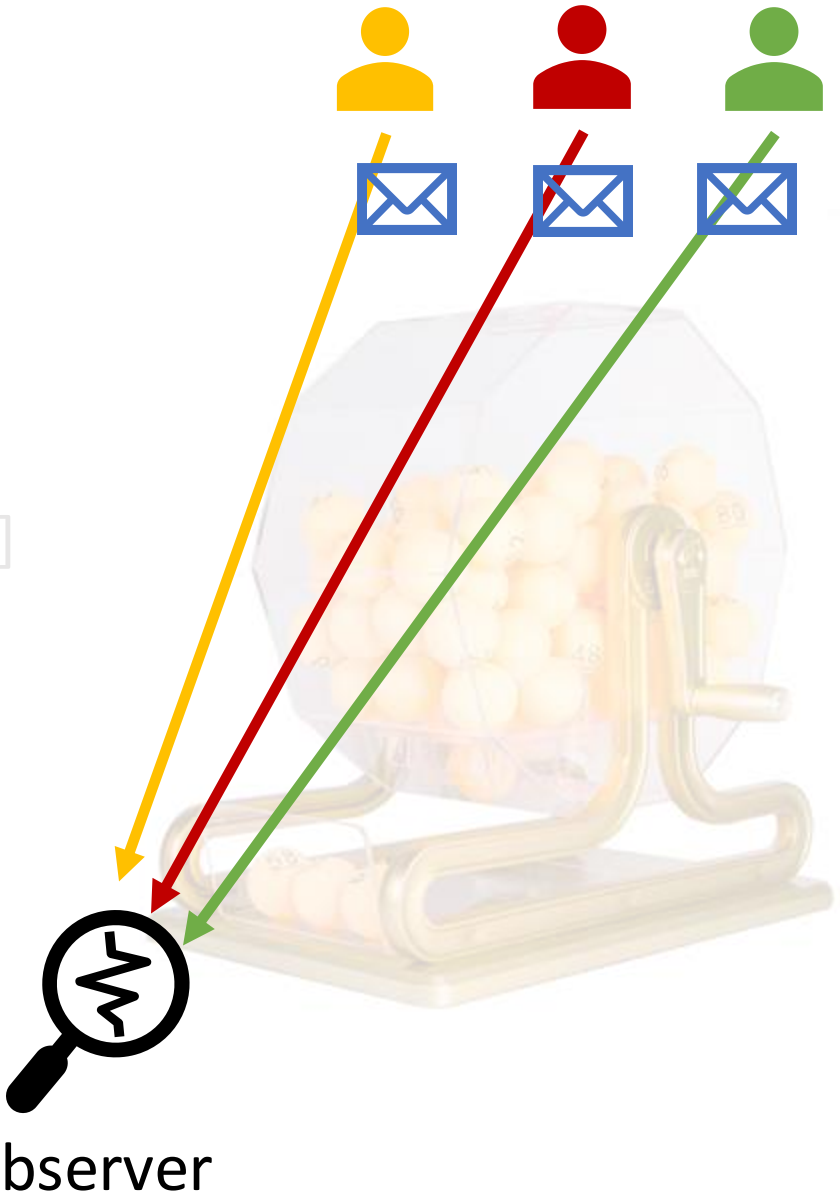
The shuffle model

- Purpose: **anonymization**
- An existing notion in many literatures
  - Anonymous communication, e.g., [HLZZ15]
  - Differential privacy, e.g., [BBGN20]
  - Secure aggregation, e.g., [IKOS06]
- In our setting:
  assume a two-way anonymous channel

A shuffler

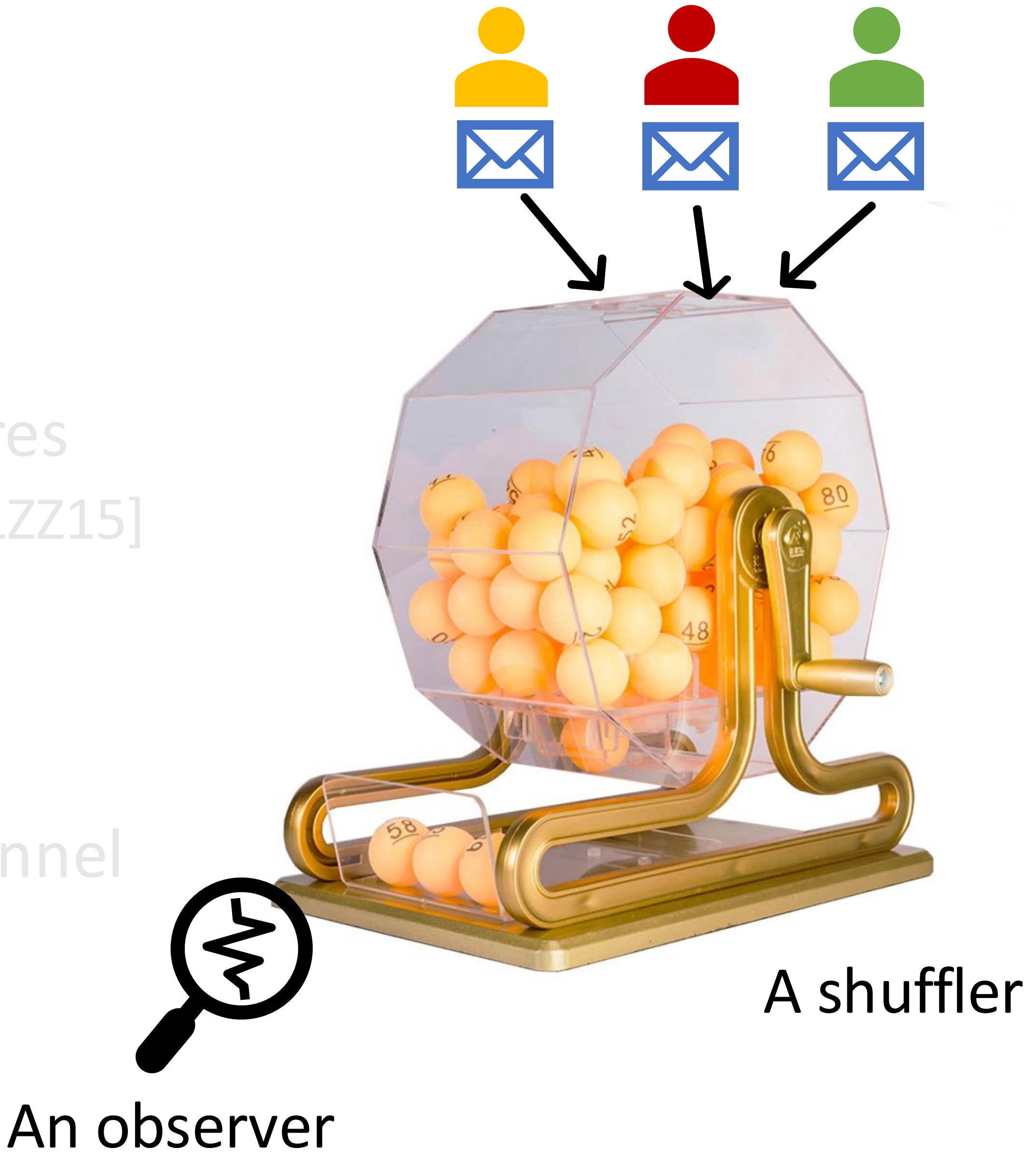
The shuffle model

- Purpose: **anonymization**
- An existing notion in many literatures
  - Anonymous communication, e.g., [HLZZ15]
  - Differential privacy, e.g., [BBGN20]
  - Secure aggregation, e.g., [IKOS06]
- In our setting:
  assume a two-way anonymous channel



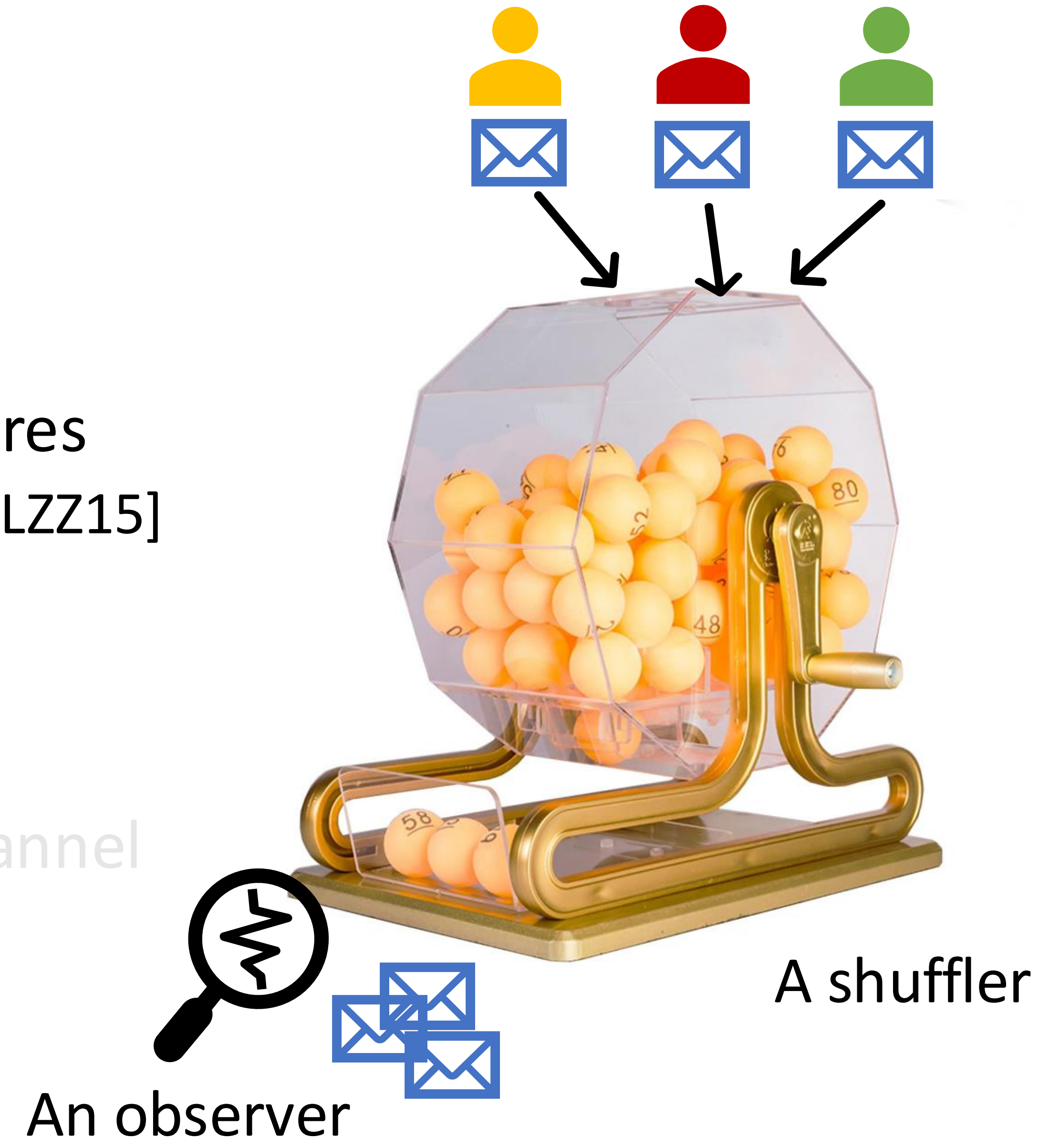
The shuffle model

- Purpose: **anonymization**
- An existing notion in many literatures
  - Anonymous communication, e.g., [HLZZ15]
  - Differential privacy, e.g., [BBGN20]
  - Secure aggregation, e.g., [IKOS06]
- In our setting:
  assume a two-way anonymous channel



The shuffle model

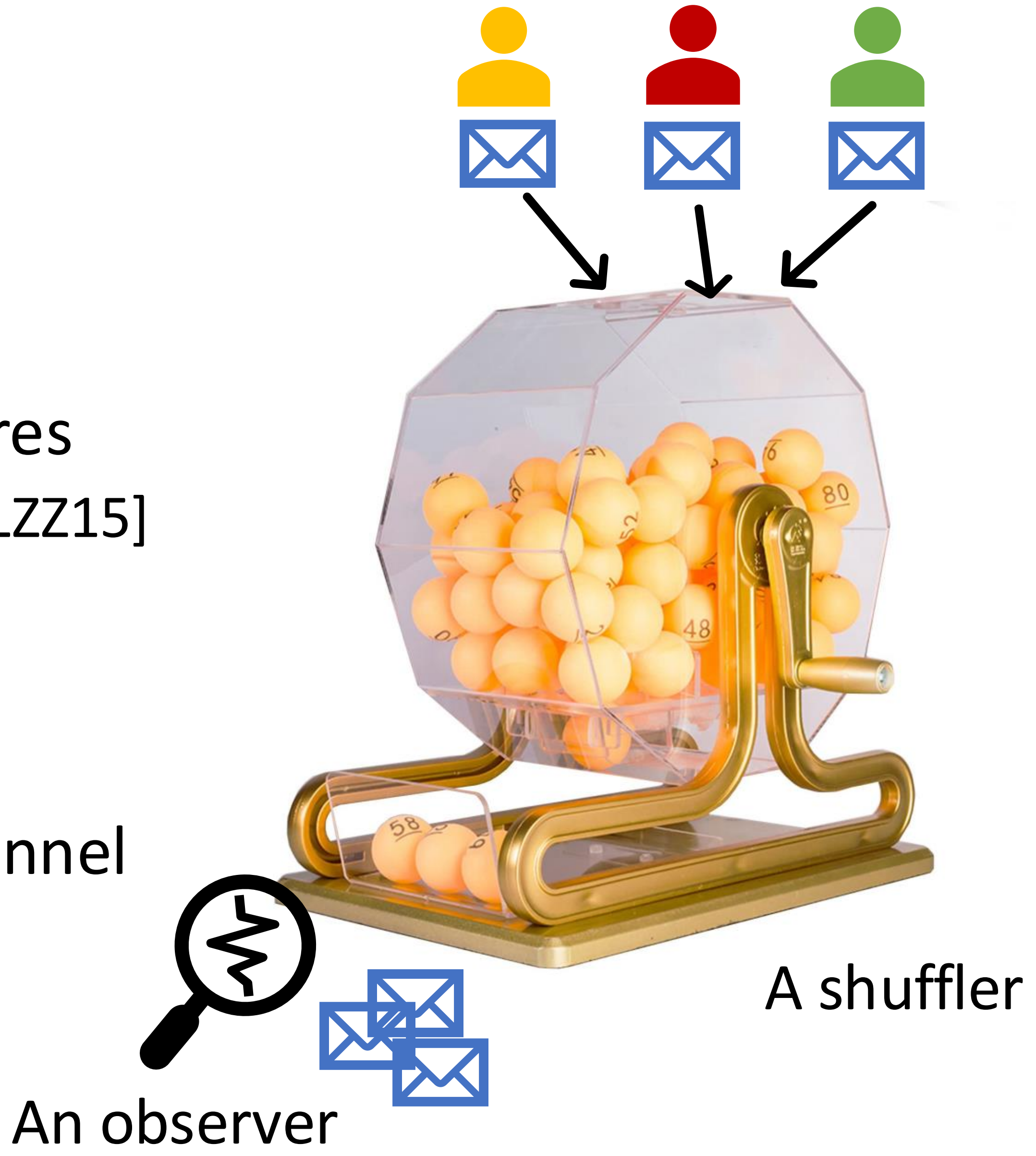
- Purpose: **anonymization**
- An existing notion in many literatures
  - Anonymous communication, e.g., [HLZZ15]
  - Differential privacy, e.g., [BBGN20]
  - Secure aggregation, e.g., [IKOS06]
- In our setting:
  assume a two-way anonymous channel



The shuffle model

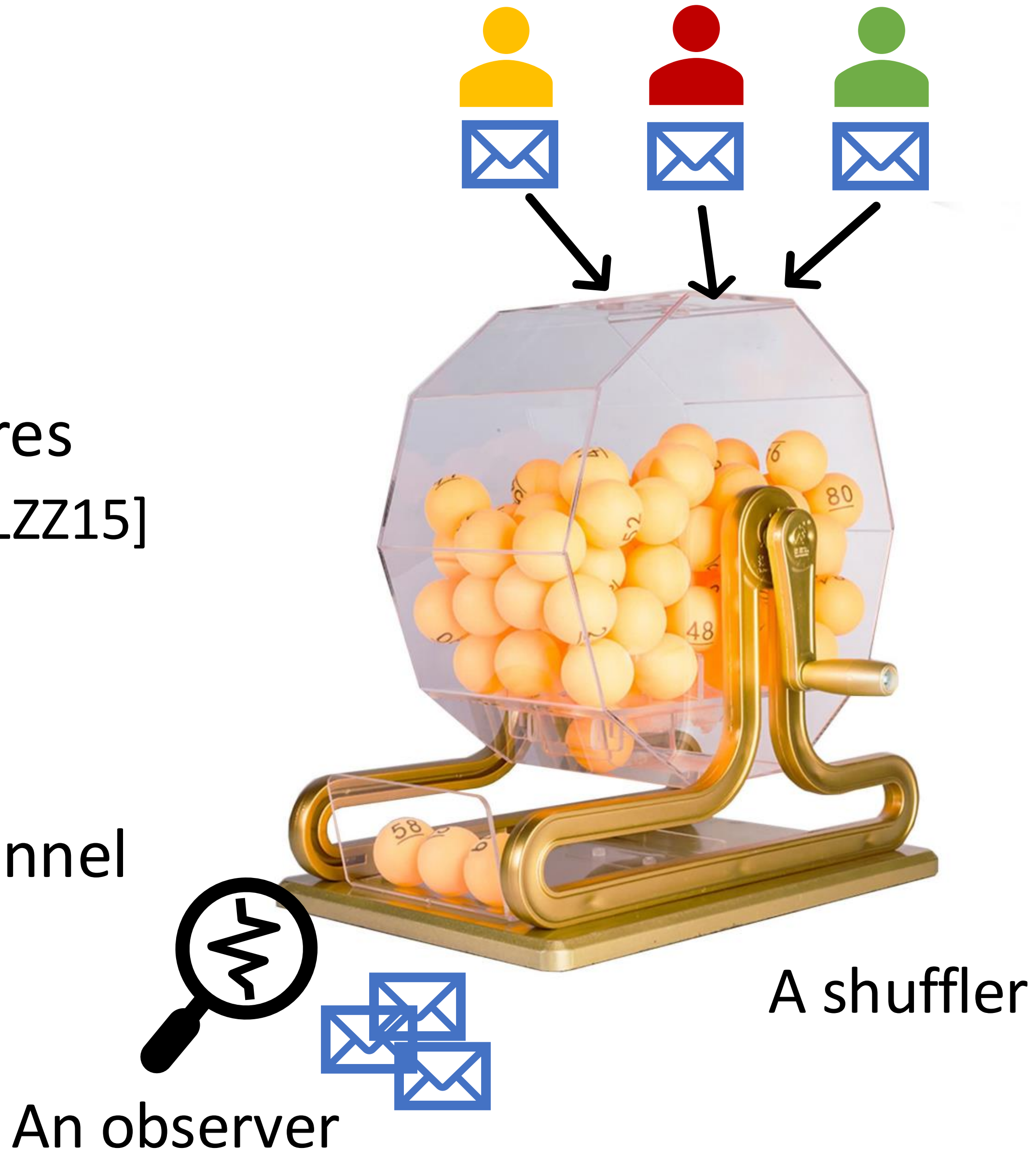
- Purpose: **anonymization**
- An existing notion in many literatures
  - Anonymous communication, e.g., [HLZZ15]
  - Differential privacy, e.g., [BBGN20]
  - Secure aggregation, e.g., [IKOS06]
- In our setting:
  assume a two-way anonymous channel

Strong assumption?

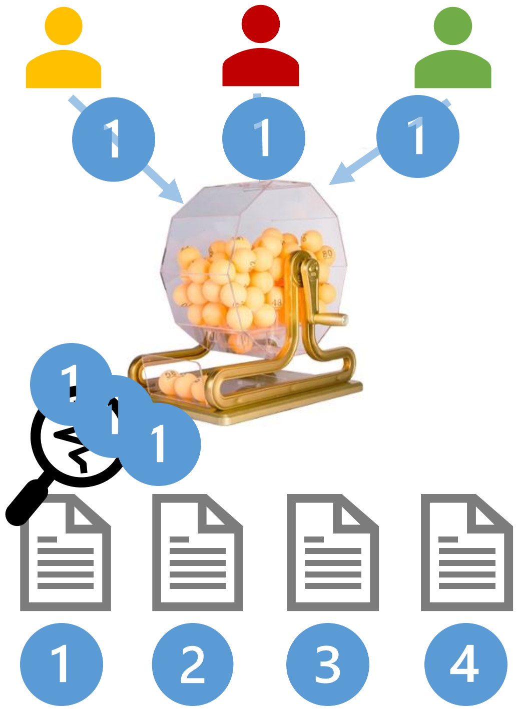


The shuffle model

- Purpose: **anonymization**
- An existing notion in many literatures
  - Anonymous communication, e.g., [HLZZ15]
  - Differential privacy, e.g., [BBGN20]
  - Secure aggregation, e.g., [IKOS06]
- In our setting:
  assume a two-way anonymous channel
- Instantiation:
  stay tuned for discussion!

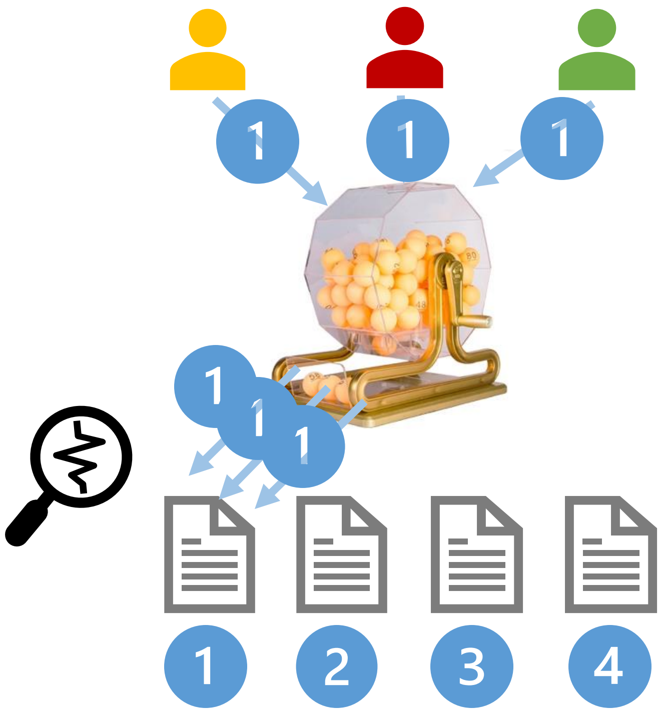


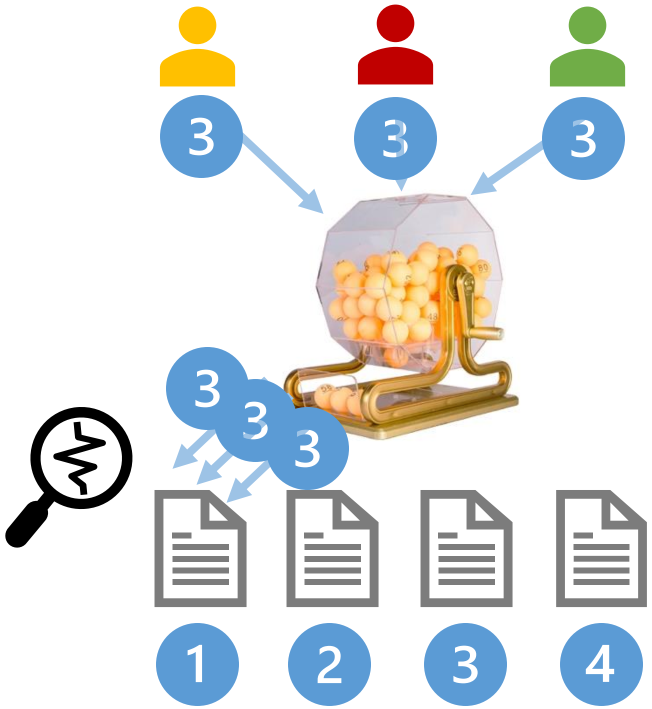
PIR in the shuffle model



PIR in the shuffle model

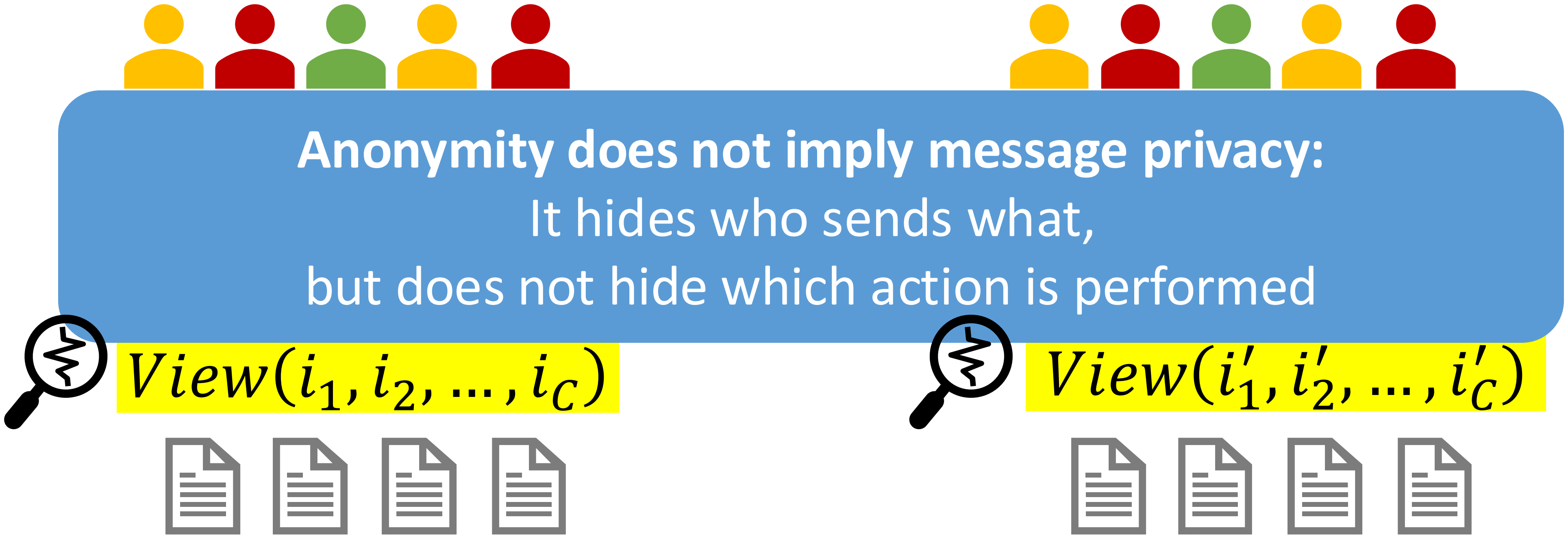
- Anonymization does not trivialize the PIR problem!





PIR in the shuffle model

- Anonymization does not trivialize the PIR problem!



PIR in the shuffle model

- Privacy from anonymity [IKOS06]: Secure sum from **"split and mix"**

5 1 0 3 8

Take a large enough p, each client splits its inputs into k shares in $\mathbb{Z}_p$

PIR in the shuffle model

- Privacy from anonymity [IKOS06]: Secure sum from **"split and mix"**

5 1 0 3 8

4+10+11 6+14+1 16+2+2 14+2+7 17+2+9

$p = 20, k = 3$

Shuffle the shares

Get the sum without learning any individual's input

PIR in the shuffle model

- Privacy from anonymity [IKOS06]: Secure sum from **"split and mix"**

10 2 2 1 1 4 4 4 4 0

Each input is split to k shares

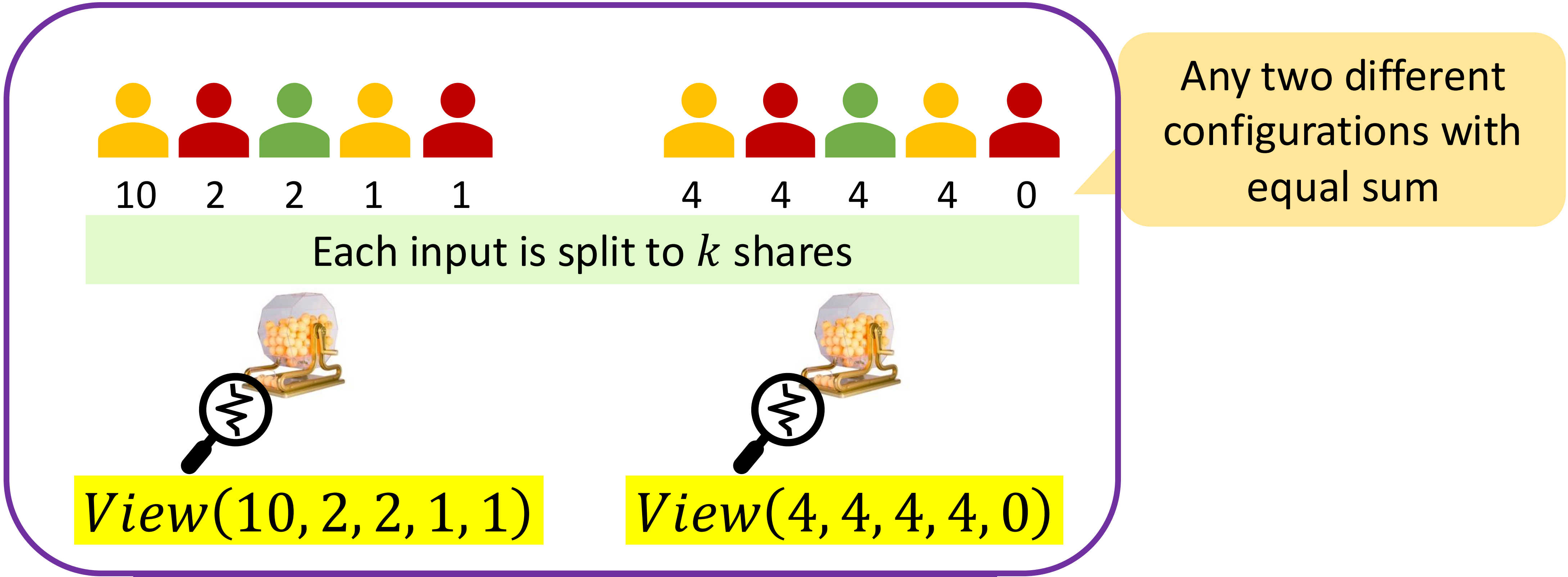
Any two different configurations with equal sum

Split and mix can provide statistical security against the observer

$View(10, 2, 2, 1, 1)$ $View(4, 4, 4, 4, 0)$

PIR in the shuffle model

- Privacy from anonymity [IKOS06]: Secure sum from **"split and mix"**



Can "split and mix" help in the PIR problem?

Split and mix in PIR

- Privacy from anonymity [IKOS06]: **"split and mix"**

i_1 i_2 i_3 i_4 i_5

4+10+11 6+14+1 16+2+2 14+2+7 17+2+9

Split each index into additive shares?

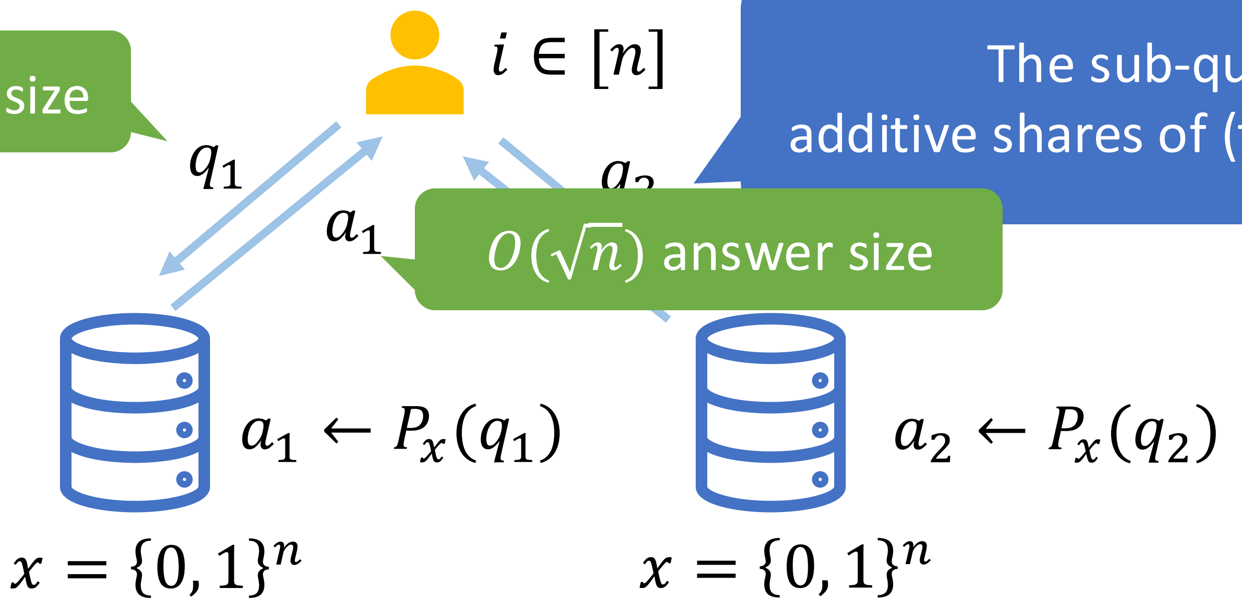
Answer to each share

Split and mix in PIR

- A two-server “additive PIR” [BIK04]

$O(\log n)$ query size

The sub-queries q_1, q_2 are additive shares of (the encoding of) index i

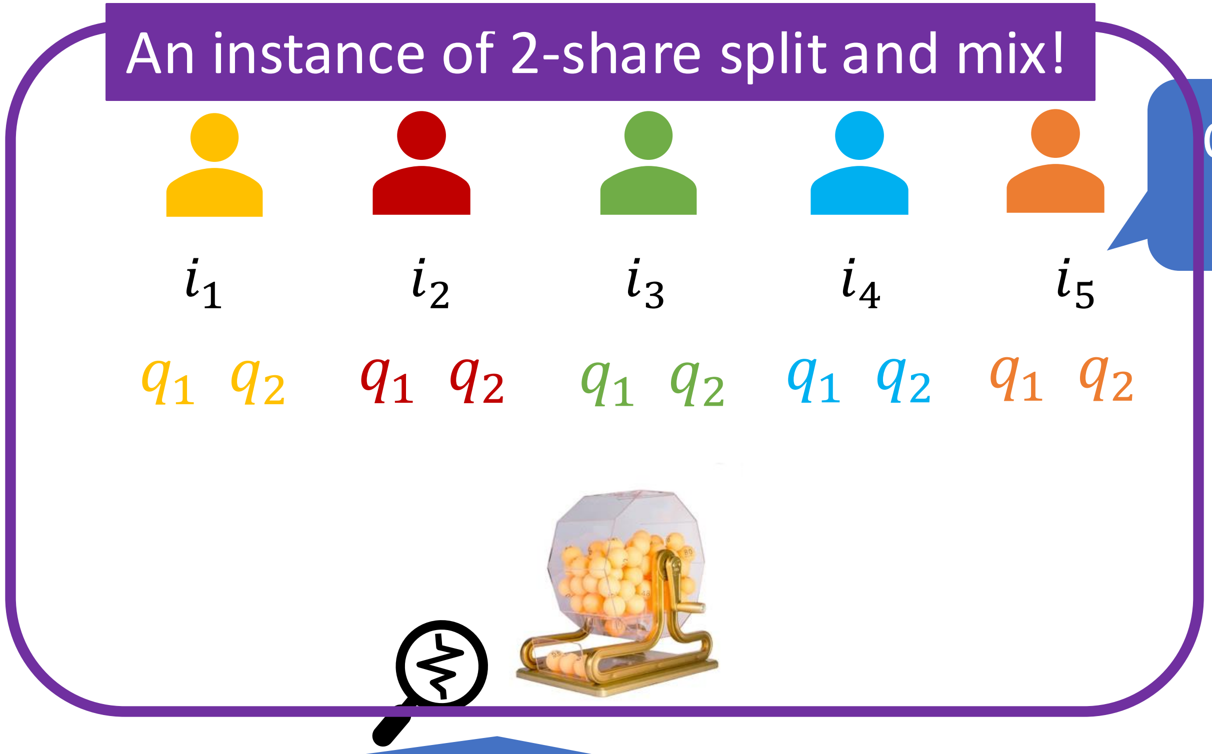


$O(\sqrt{n})$ answer size

Takeways: 1. Sub-queries are additive shares
2. Answer algorithm is simply P_x(share)

Split and mix in PIR

- A construction from the two-server "additive PIR"



Query using the two-server "additive PIR" protocol

Are we done?

Only learns the sum of all sub-queries but nothing else

Split and mix in PIR

Similar attack also generalizes to $\mathbb{Z}_p$

- 2-share is not enough to provide privacy: a simple example in $\mathbb{Z}_2$

All clients with input 0 v.s. All clients with input 1

0 can be split to 0+0 or 1+1

#0s and #1s may not be exactly equal

1 can only be split to 0+1

Exactly equal #0s and #1s in the shares!

Split and mix in PIR

- **Can we do more share?** Yes, but worse efficiency:

  The k-server "additive PIR" gives communication $O(n^{\frac{k-1}{k}})$

Our technique:
Randomize the query index for the "additive PIR"
using an outer layer of PIR

↓

Communication $O(n^{\frac{1}{2}}\,\mathrm{polylog}(n))$

General constructions: an "inner-outer" paradigm

Given any set of query indices

i_1 i_2 i_3 $\in [n]$

Learns nothing (except the sum)

Recall the problem

When $i_1, i_2, \ldots, i_C$ and $i'_1, i'_2, \ldots, i'_C$ are far apart, e.g., 1 1 1 1 1 v.s. 2 2 2 2 2

$View(i_1, i_2, \ldots, i_C)$ and $View(i'_1, i'_2, \ldots, i'_C)$ are also far apart

General constructions: an "inner-outer" paradigm

Given any set of query indices

i_1 i_2 i_3 $\in [n]$

Learns nothing (except the sum)

Our construction technique

A step forward

1 1 1 1 1 2 2 2 2 2

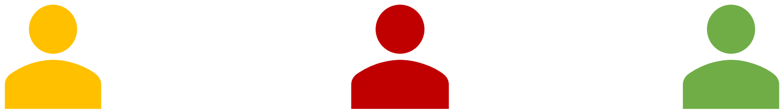
$

If we can make $i_1, i_2, \dots, i_C$ and $i'_1, i'_2, \dots, i'_C$ closer, e.g., 1 2 3 4 4 v.s. 1 2 3 4 5

Would $View(i_1, i_2, \dots, i_C)$ and $View(i'_1, i'_2, \dots, i'_C)$ be close?

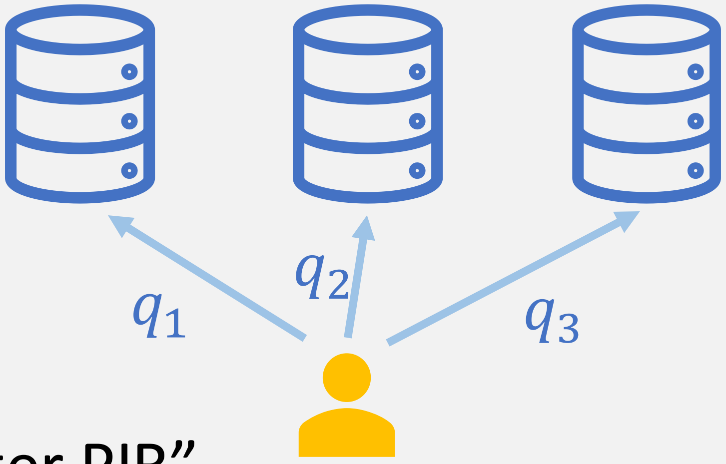
Our proof technique

General constructions: an “inner-outer” paradigm



How to randomize the indices? i_1 i_2 i_3 $\in [n]$

An important observation



“Outer PIR”

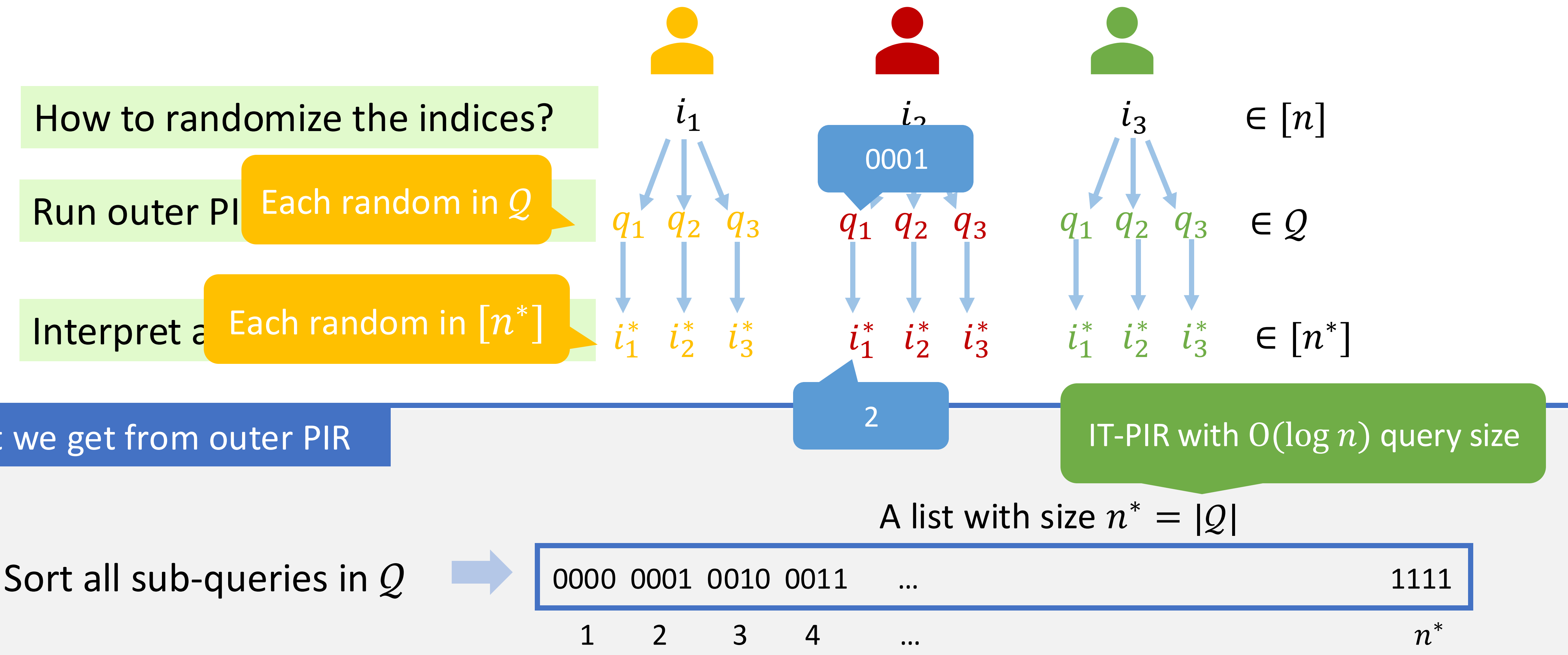
Consider PIR query algorithm:

$$(q_1, q_2, q_3) \leftarrow Query(i; r)$$

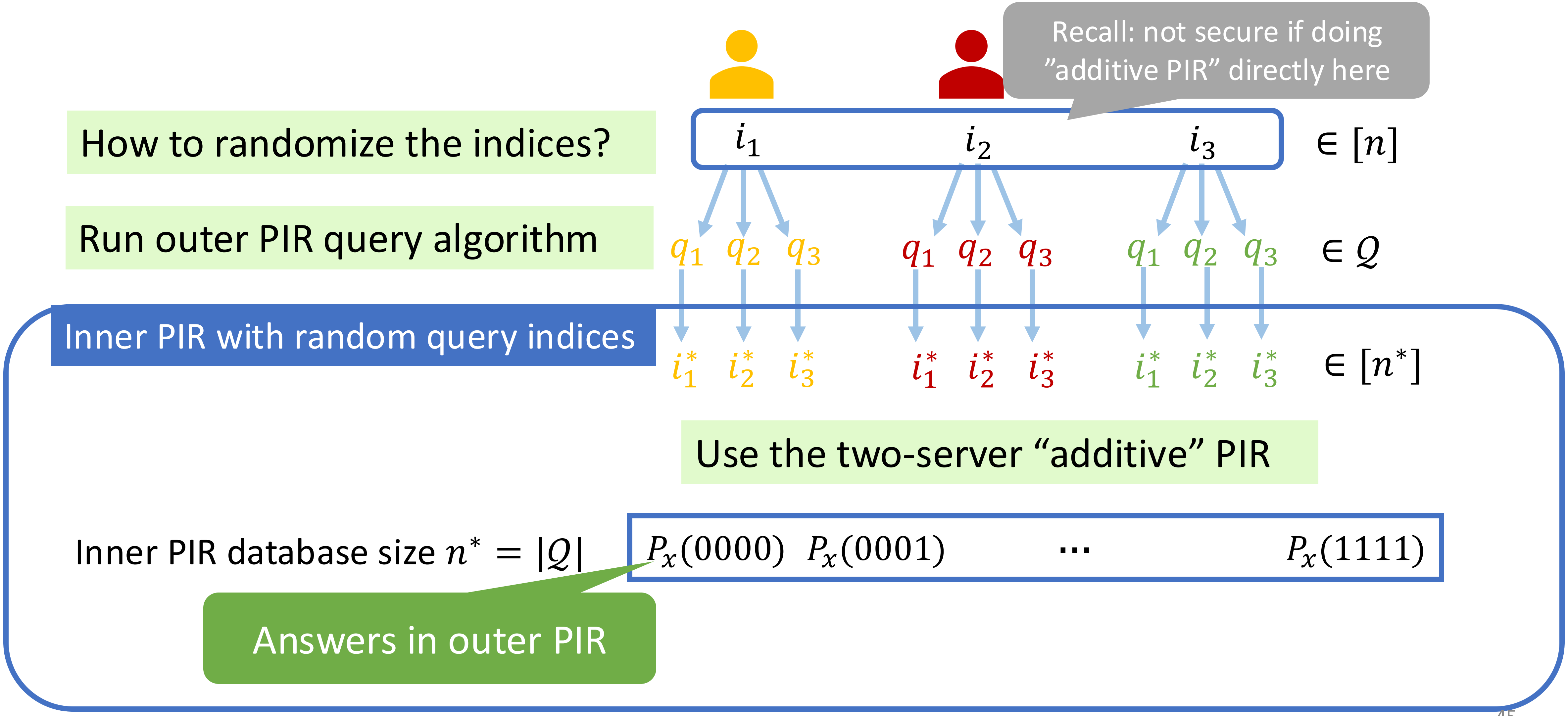
Let Q be the space that consists of all possible sub-queries

For any given $i \in [n]$, each sub-query q is uniformly random over Q

General constructions: an "inner-outer" paradigm



General constructions: an “inner-outer” paradigm



General constructions: an “inner-outer” paradigm

How to randomize the indices?

i_1 i_2 i_3 $\in [n]$

Run outer PIR query algorithm

q_1 q_2 q_3 q_1 q_2 q_3 q_1 q_2 q_3 $\in \mathcal{Q}$

Inner PIR with random query indices

i_1^* i_2^* i_3^* i_1^* i_2^* i_3^* i_1^* i_2^* i_3^* $\in [n^*]$

The distributions of the shuffled additive shares
from any index configurations are close
(with some tweaks)

General constructions: an “inner-outer” paradigm

A brief summary

On any query indices — i_1

Outer PIR: Any k-server protocol (k>2)

Run outer PIR query algorithm — q_1 q_2 q_3 q_1 q_2 q_3 q_1 q_2 q_3 $\in \mathcal{Q}$

Interpret as indices for inner PIR — i_1^* i_2^* i_3^* i_1^* i_2^* i_3^* i_1^* i_2^* i_3^* $\in [n^*]$

Use inner PIR for re

Inner PIR sub-queri

Inner PIR: The two-server “additive PIR”

The server prepares this in advance — $P_x(0000)$ $P_x(0001)$ ⋯ $P_x(1111)$ — Size n^*

A single server!

General constructions: an "inner-outer" paradigm

Theorem (Informal).
On any database size n, the "inner-outer" construction with any outer PIR and the two-server additive inner PIR, gives a single-server PIR in the shuffle model that has $1/\mathrm{poly}(n)$ statistical security and $O(\sqrt{n})$ per-query communication, assuming $\mathrm{poly}(n)$ clients simultaneously accessing the database.

Corollary (Informal).
Using fancier inner PIR ("CNF PIR"), on any database size n, for every constant γ, there is a PIR construction that has

- Per-query communication and computation $O(n^{\gamma})$,
- Server storage $O(n^{1+\gamma})$,

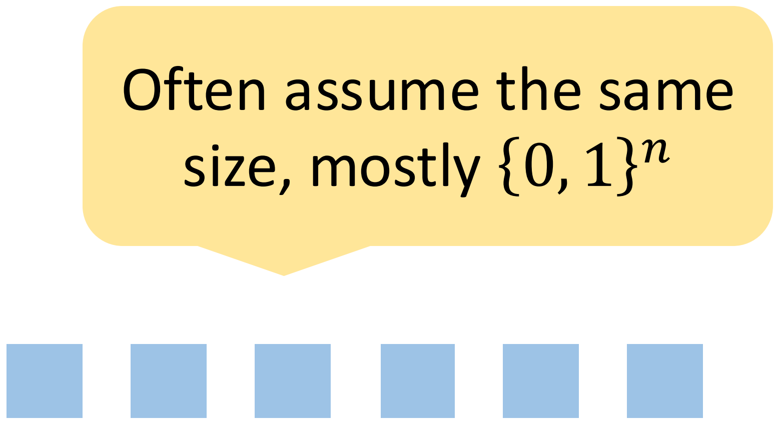
assuming one-time preprocessing.

Rest of this talk

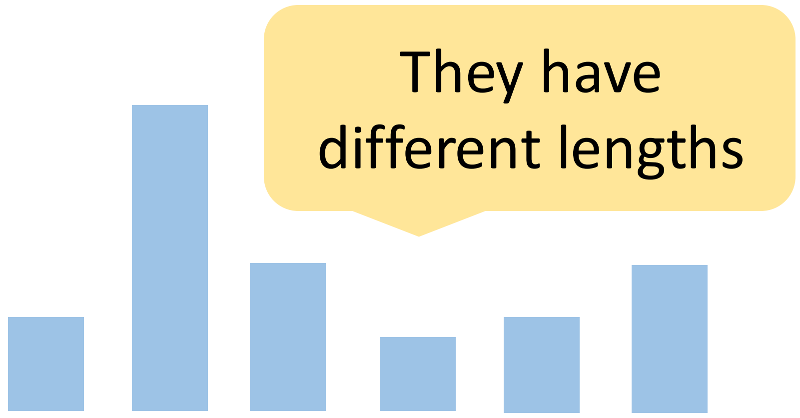
- Background
  - The shuffle model
  - “Split and mix”
- Our results
  - General constructions
  - Lower bound: the security we get in the general constructions is “tight”
  - An interesting orthogonal problem: hiding record size without padding
- Discussion and open questions

PIR with variable-sized records

- To deploy PIR in real-world applications…



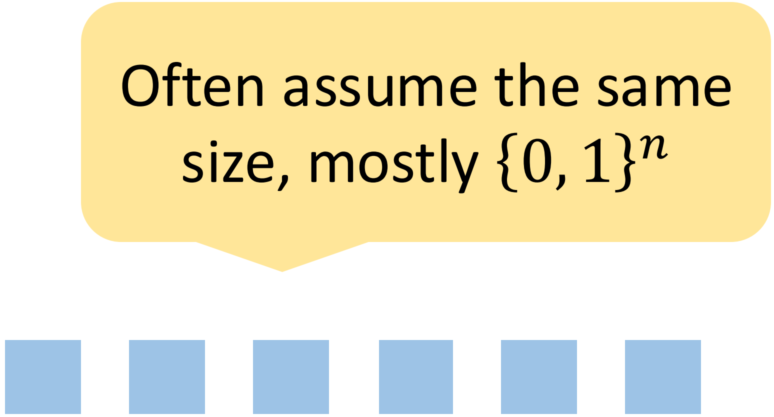
Database entries of PIR in theory



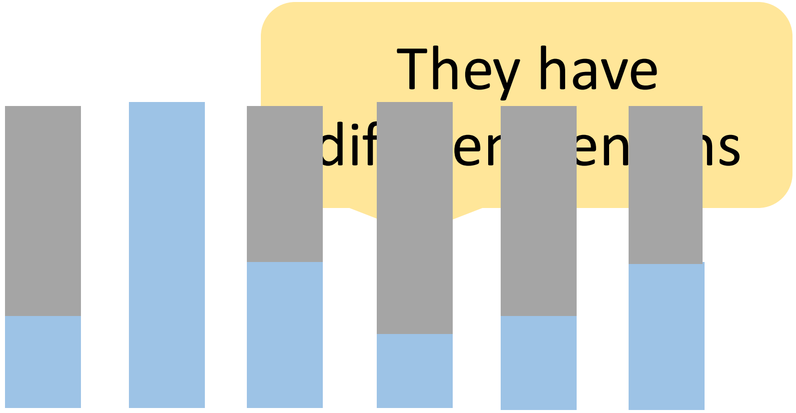
Database records in practice

PIR with variable-sized records

- To deploy PIR in real-world applications…



Database entries of PIR in theory

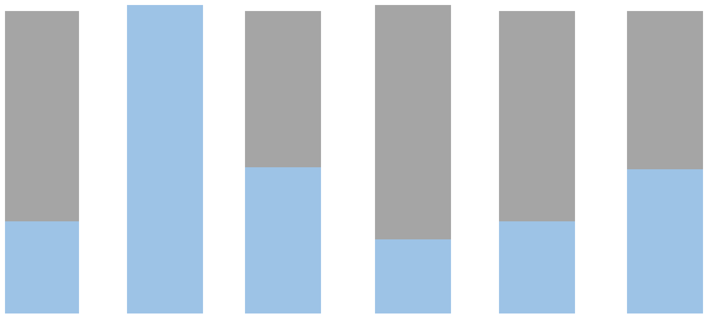


Database records in practice

To retrieve privately, it is necessary to hide record size

PIR with variable-sized records

- Padding solves the problem: how about efficiency?



Database records in practice

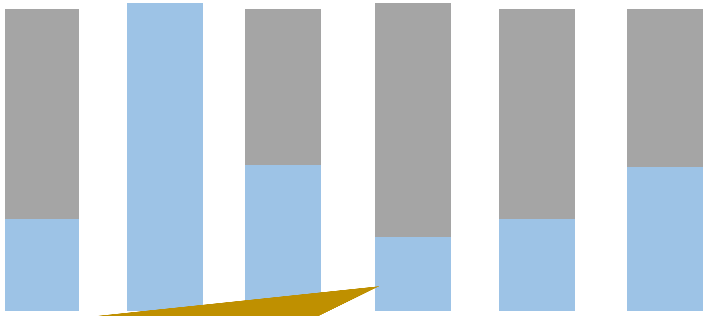
Features

The discrepancy between the smallest and the largest record can be huge
Majority of the records are small
Most users access the small records much more often than the large records

PIR with variable-sized records

- Padding solves the problem: how about e

Waste of server storage
(though can virtually store)



practice

Features

Client who retrieves the small record has to pay the cost of retrieving the largest record

The discrep an be huge

Majority of the records are small

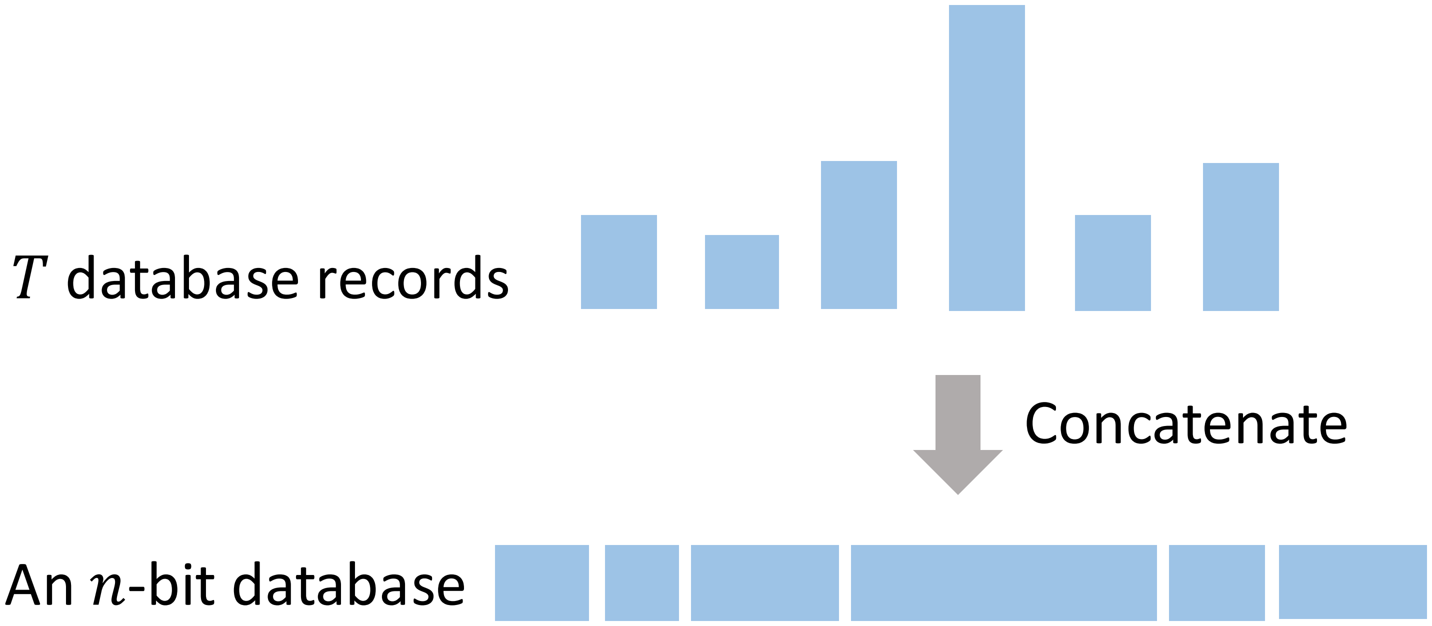
Most users access the small records much more often than the large records

PIR with variable-sized records

- In the “standard” model, there is no way out
- In the shuffle model: yes, we can
  - No server storage overhead
  - Client communication proportional to the length of the retrieved record
  - Leak only the total size of all queried records

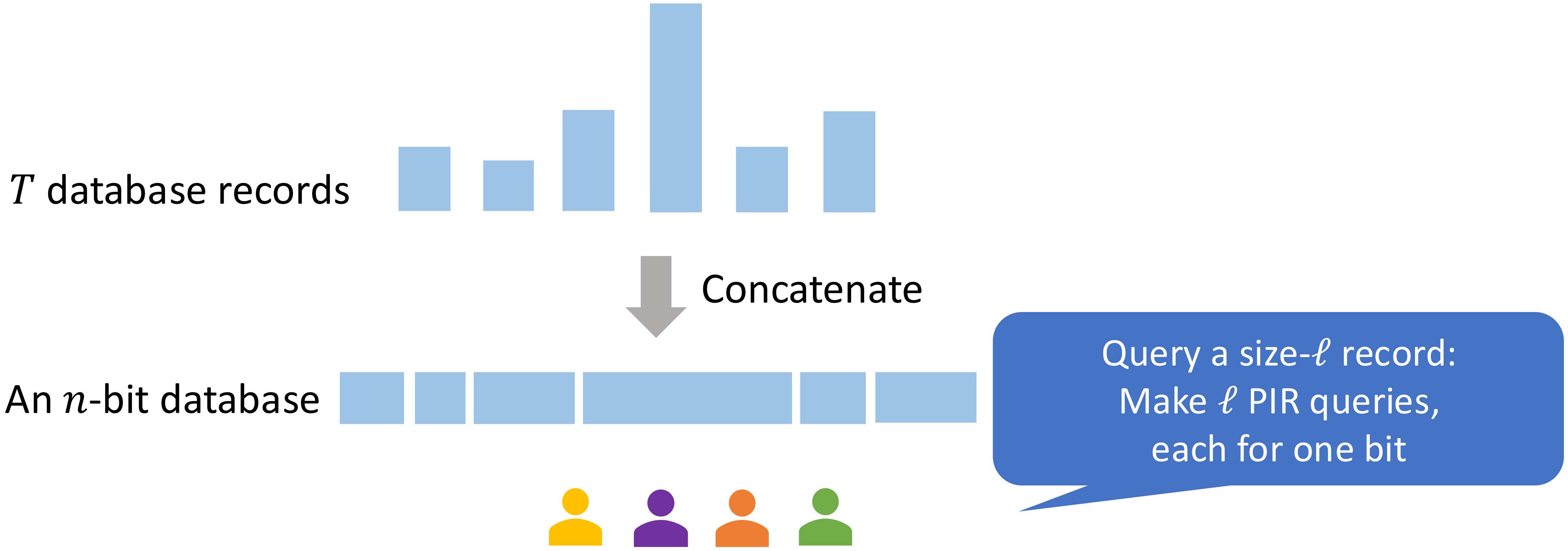
PIR with variable-sized records

- A toy protocol



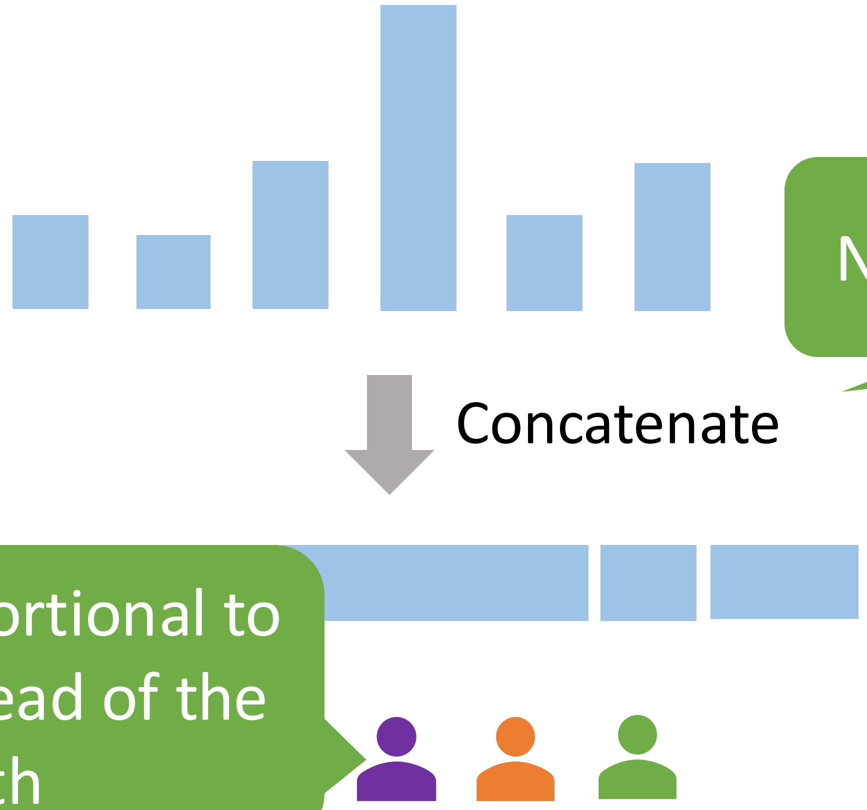
PIR with variable-sized records

- A toy protocol



PIR with variable-sized records

- A toy protocol



PIR with variable-sized records

- A toy protocol

T data

age overhead

Can we do better?

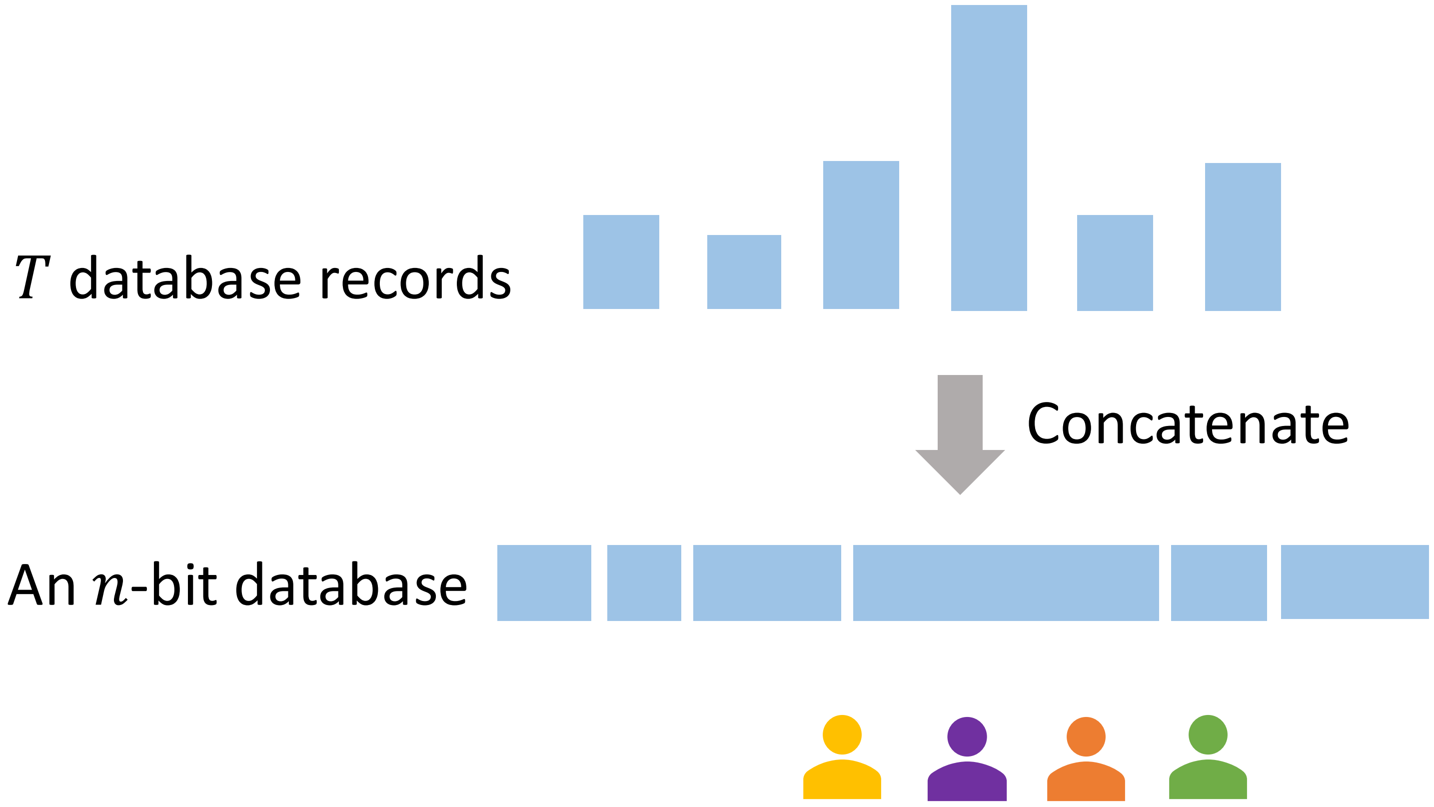
Yes, from ℓ PIR queries to polylogℓ PIR queries

Query a size-ℓ record:
Make ℓ PIR queries,
each for one bit

Communication is proportional to the queried length instead of the maximum length

PIR with variable-sized records

- Revisit the toy protocol



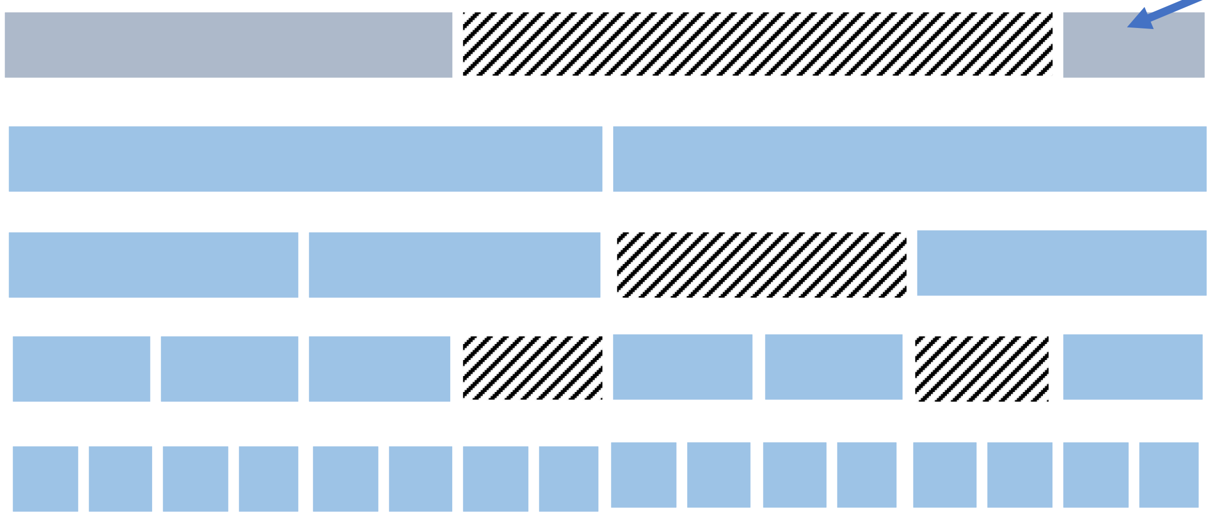
Why not retrieve more bits in each PIR query?

Query a size-ℓ record:
Make ℓ PIR queries,
each for one bit

PIR with variable-sized records

- Splitting records to the powers of two

The n-bits concatenated database



Server (logically) preprare $\log n$ databases:
the j-th database is partitioned to 2^j bits per entry

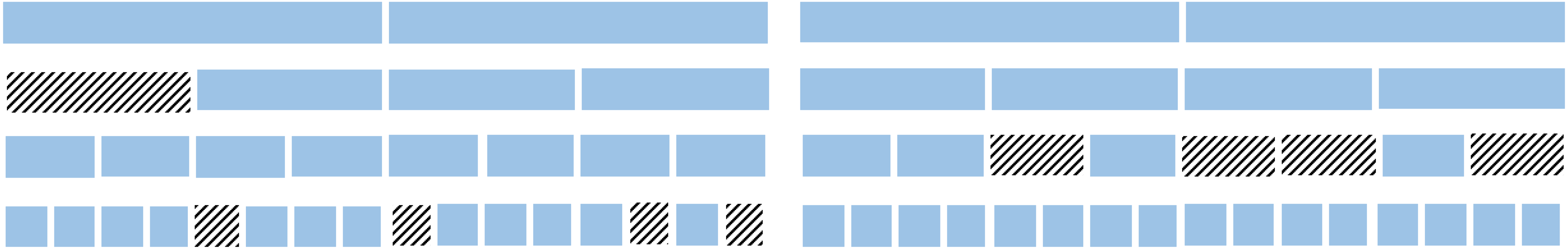
Secure or not?

Deterministic splitting is not secure (unless split down to 1)

PIR with variable-sized records

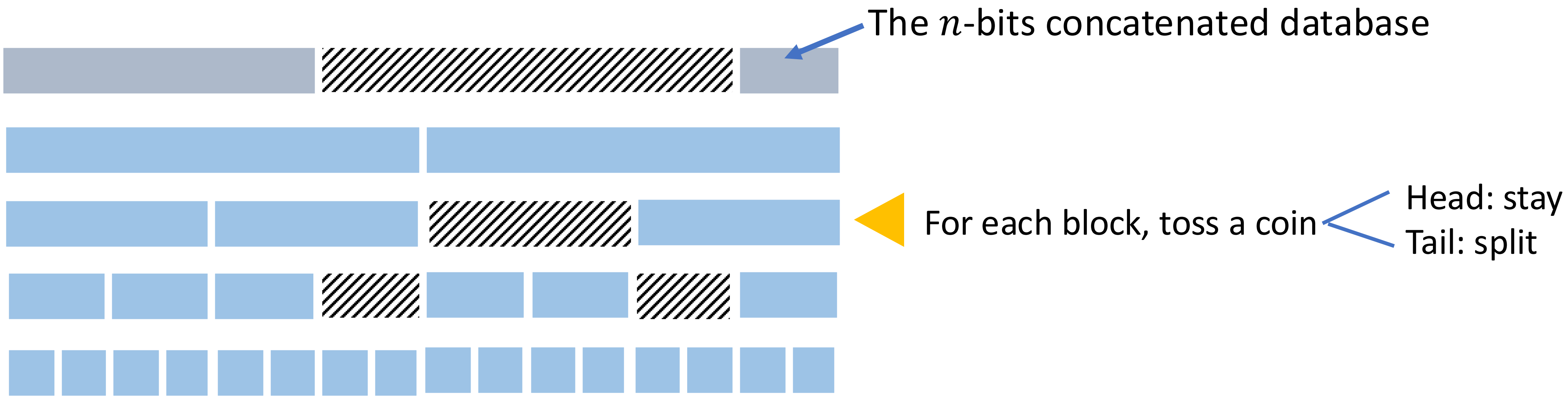
- Splitting records to the powers of two

Consider 5 1 1 1 v.s. 2 2 2 2



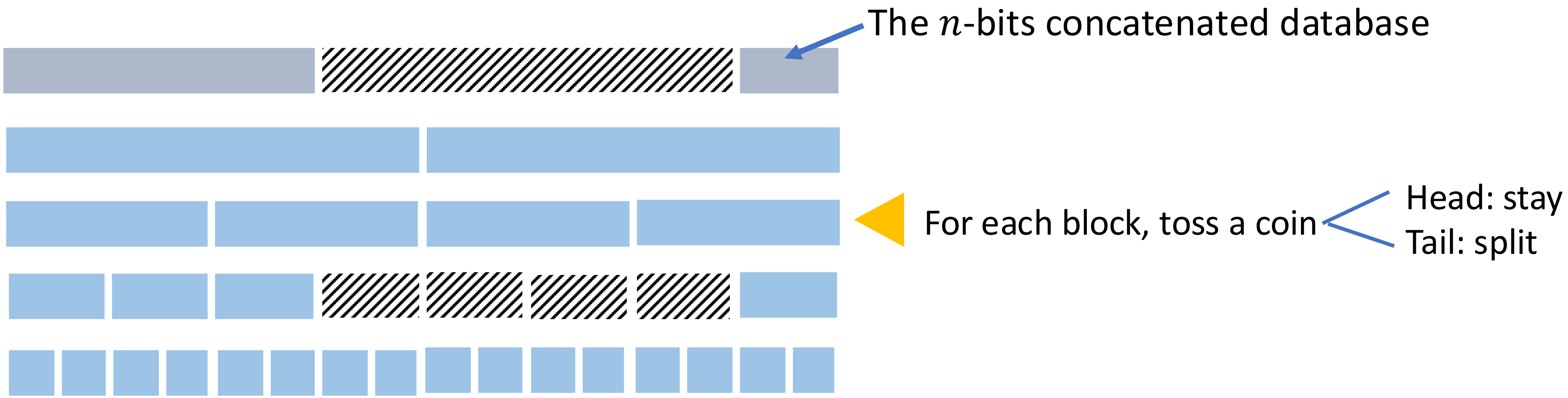
PIR with variable-sized records

- Our approach: recursive splitting



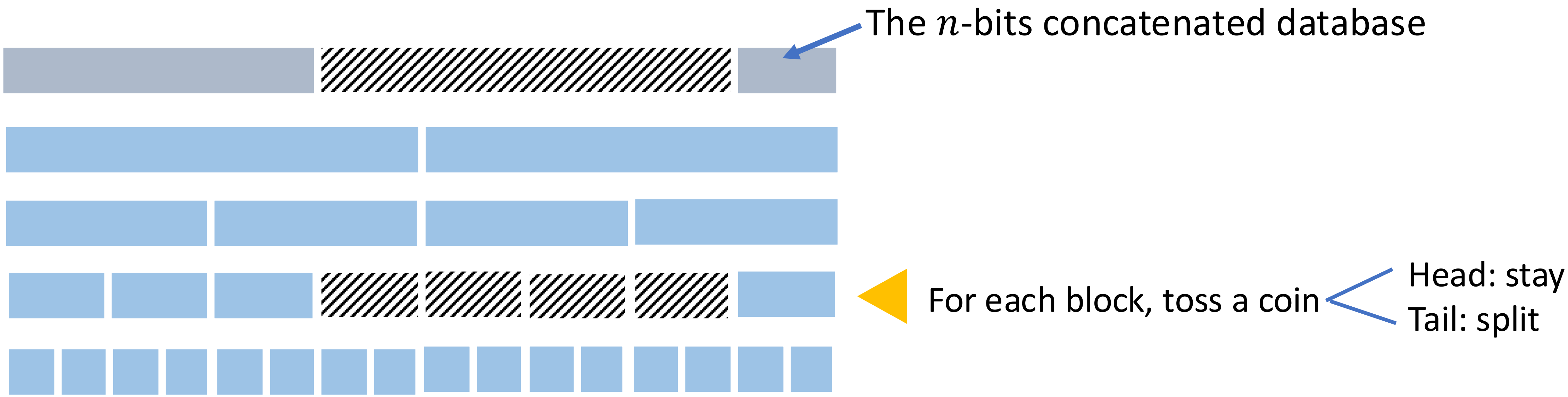
PIR with variable-sized records

- Our approach: recursive splitting



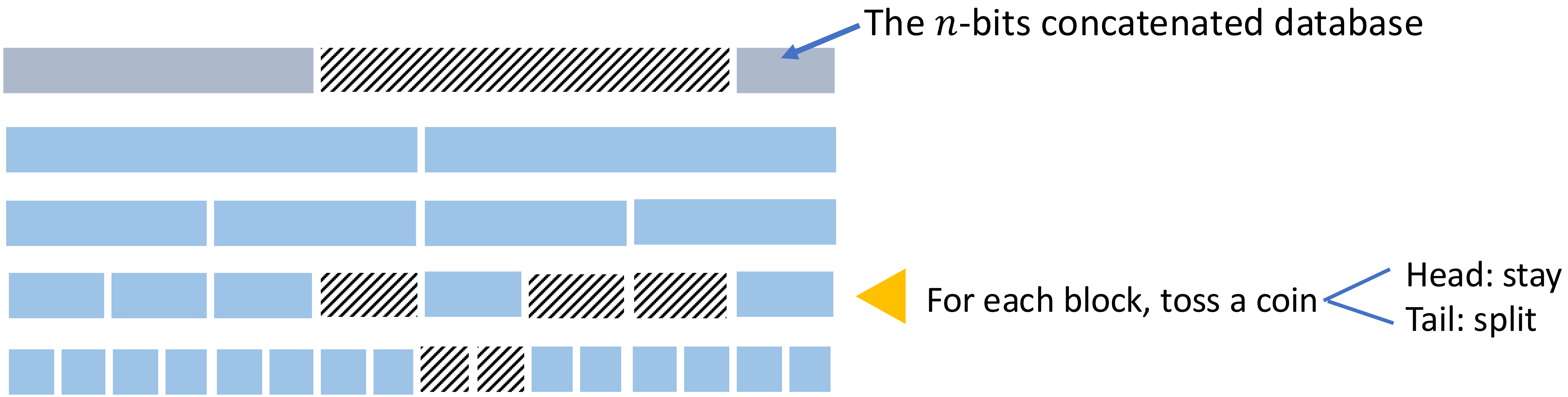
PIR with variable-sized records

- Our approach: recursive splitting



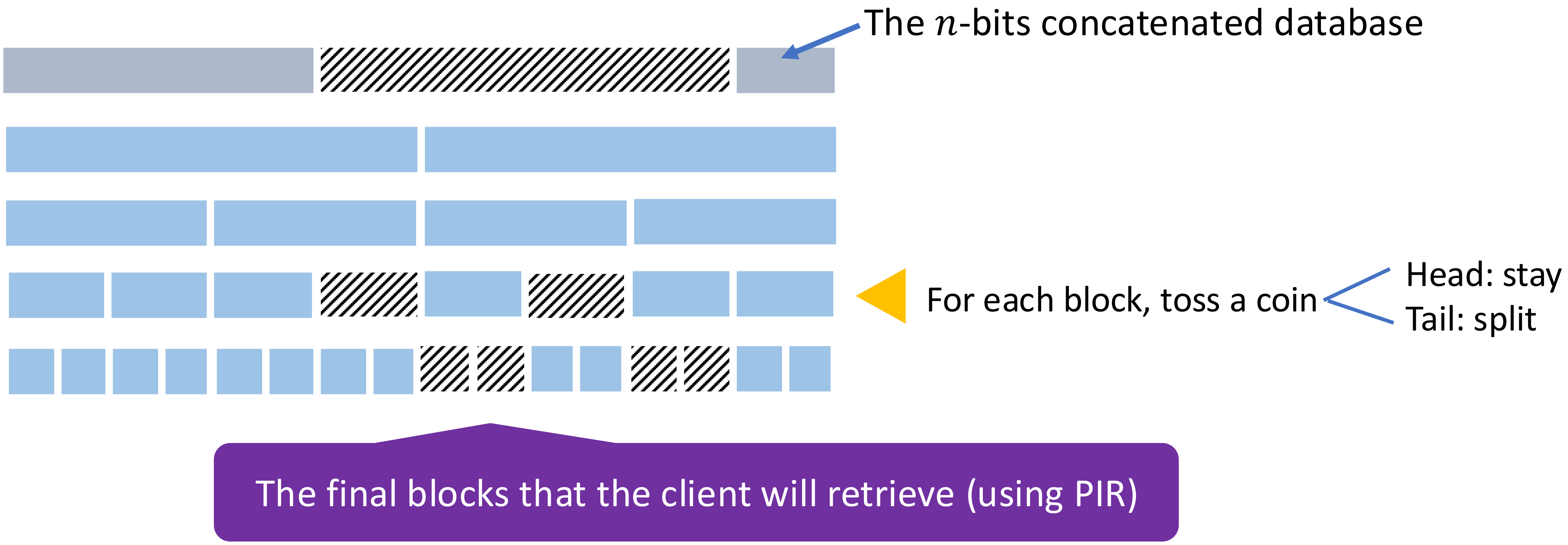
PIR with variable-sized records

- Our approach: recursive splitting



PIR with variable-sized records

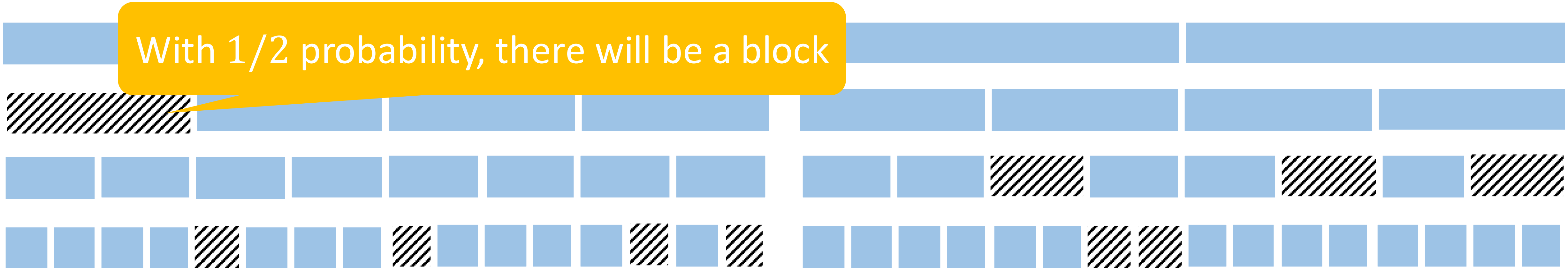
- Our approach: recursive splitting



PIR with variable-sized records

- A complication of recursive splitting: fully split the highest $\log C$ levels

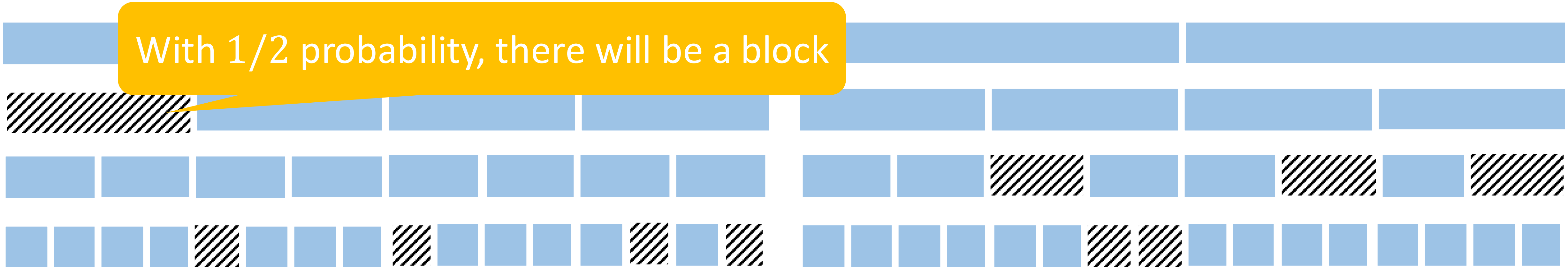
Consider 5 1 1 1 v.s. 2 2 2 2



PIR with variable-sized records

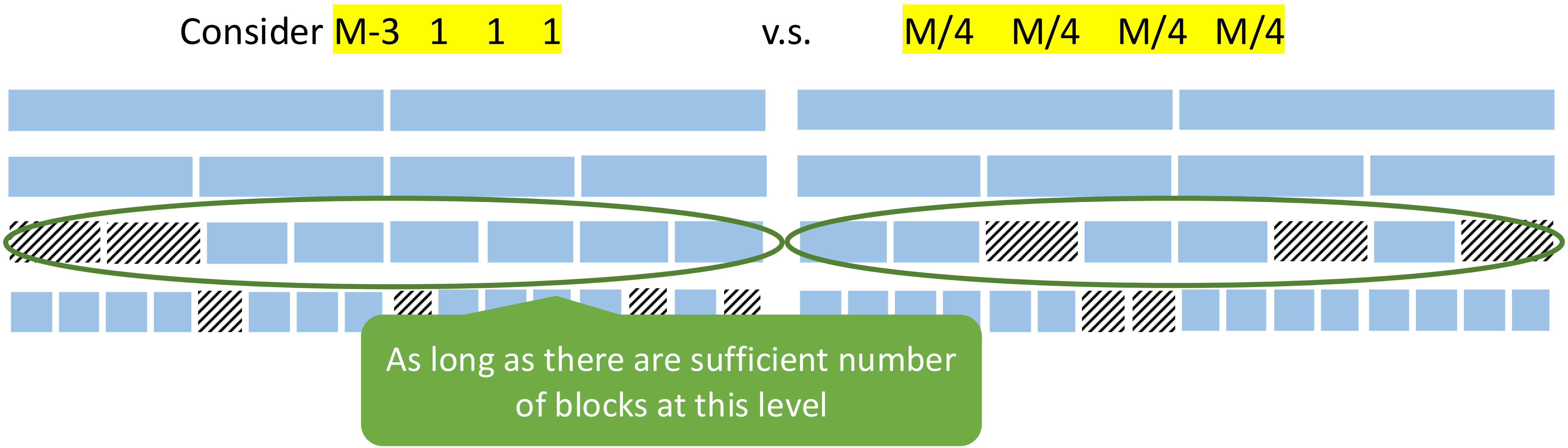
- A complication of recursive splitting: fully split the highest $\log C$ levels

Consider M-3 1 1 1 v.s. M/4 M/4 M/4 M/4



PIR with variable-sized records

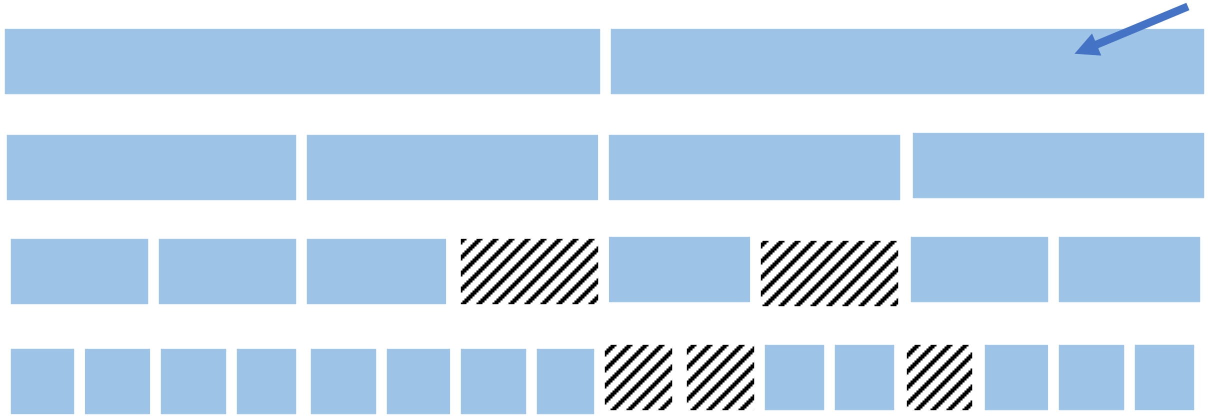
- A complication of recursive splitting: fully split the highest $\log C$ levels

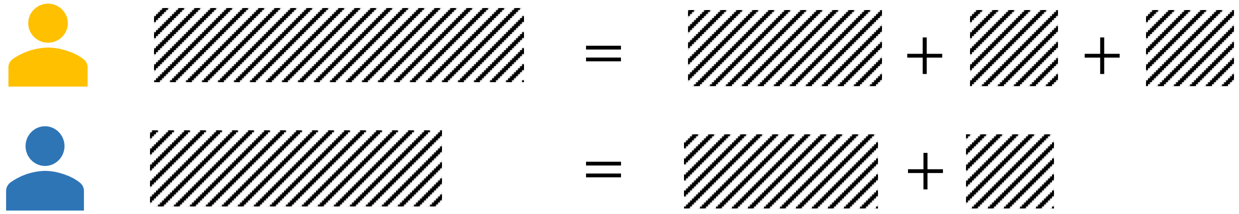


PIR with variable-sized records

- Splitting records to the power of two

The largest block ≥ maximum record size/2





The multi-set of record lengths from all clients will not leak any individual queried length

Rest of this talk

- Background
  - The shuffle model
  - “Split and mix”
- Our results
  - General constructions
  - Lower bound: the security we get in the general constructions is “tight”
  - An interesting orthogonal problem: hiding record size without padding
- Discussion and open questions

Discussion

- Two-way anonymous channel
  - A way given in DP literature: two or more non-colluding (network) servers holds a permutation

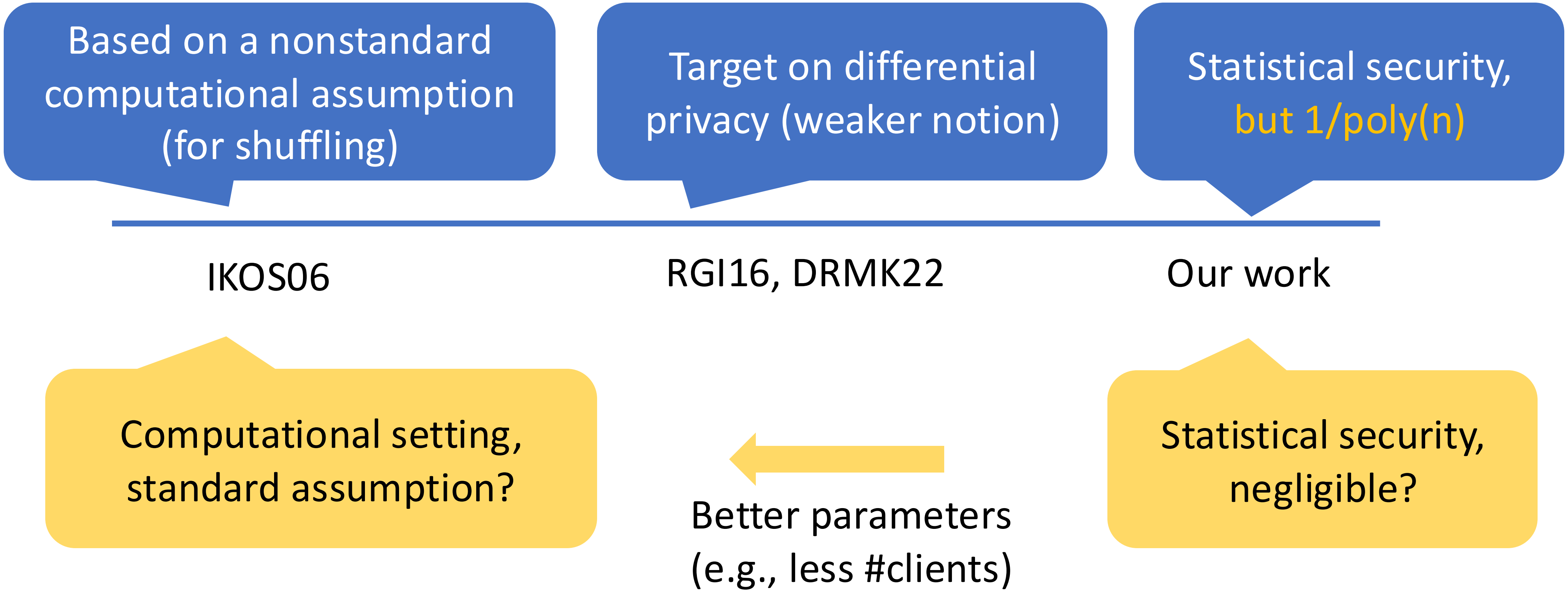
Discussion

- We want minimum assumptions
- Yet, in order to gain something (e.g., efficiency), you have to make assumptions, e.g.,
  - Hardness assumptions
  - Non-colluding assumptions

- Meanwhile, guaranteeing different assumptions does not require the same amount of effort: system efforts, law efforts, etc.
- The likelihood of assumptions being compromised in real-world scenarios may vary

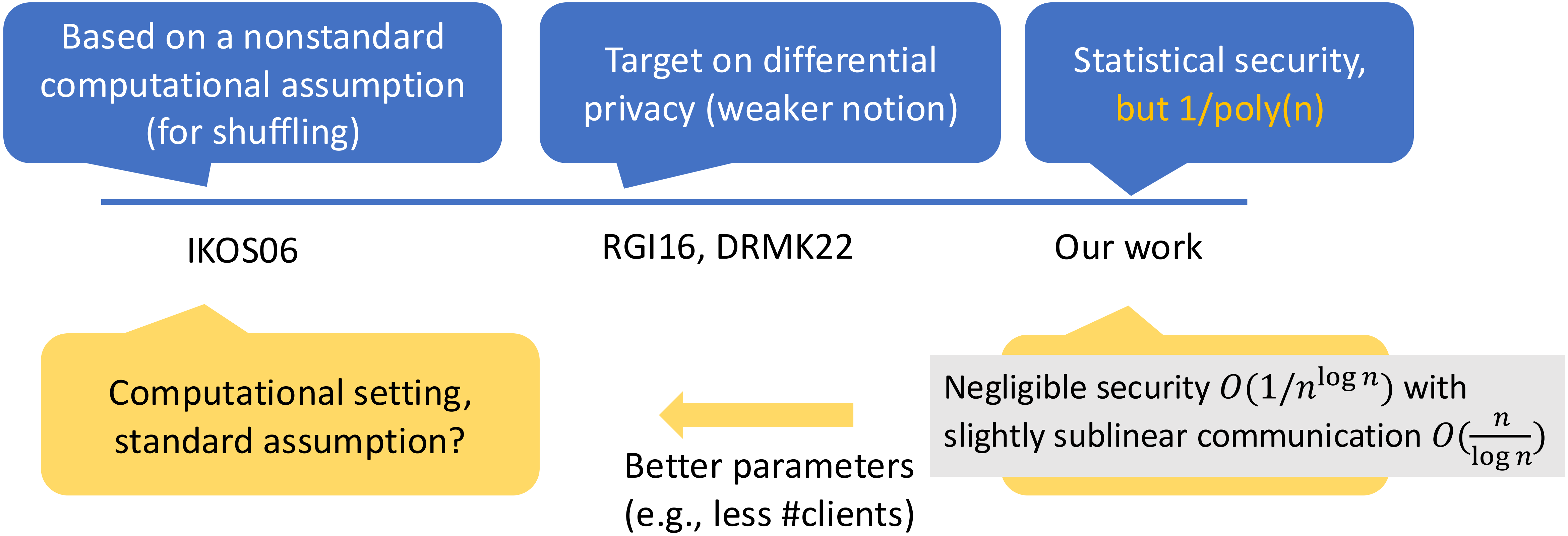
Open questions

- PIR in the shuffle model: where do we stand



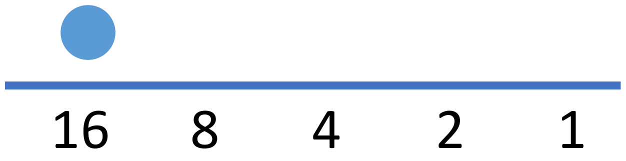
Open questions

- PIR in the shuffle model: where do we stand



Backup slides

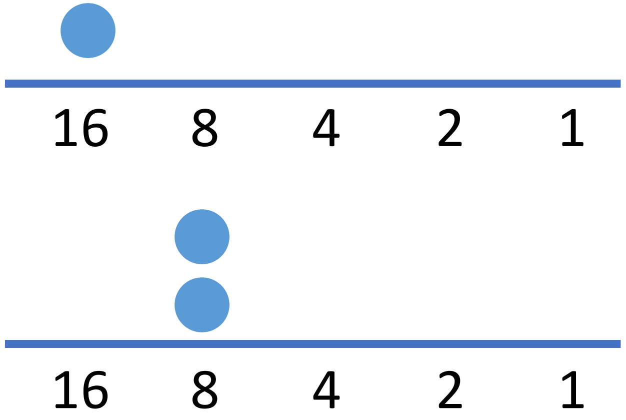
Proof idea for recursive splitting



Place the original length at the corresponding bin

Proof idea for recursive splitting

- Randomized splitting: a recursive approach

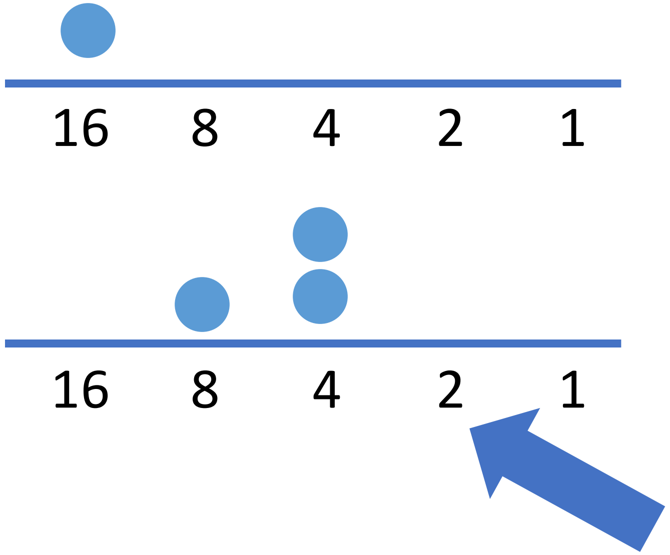


Place the original length at the corresponding bin

For each level:
    For each ball:
        Toss a coin and decide whether to split

Proof idea for recursive splitting

- Randomized splitting: a recursive approach



Place the original length at the corresponding bin

For each level:
    For each ball:
        Toss a coin and decide whether to split

Send PIR queries for each of these balls

Are we done?

Proof idea for recursive splitting

- Tweaks to the recursive approach

Queried lengths 16 1 1 1 1 8 8 2 1 1

The resulting multi-set

Proof idea for recursive splitting

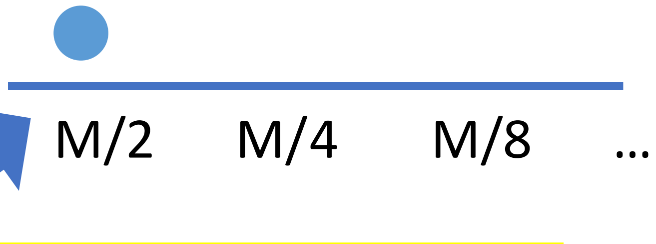
- Tweaks to the recursive approach

Queried lengths

M-4 1 1 1 1

M/5 M/5 … M/5

The resulting multi-set



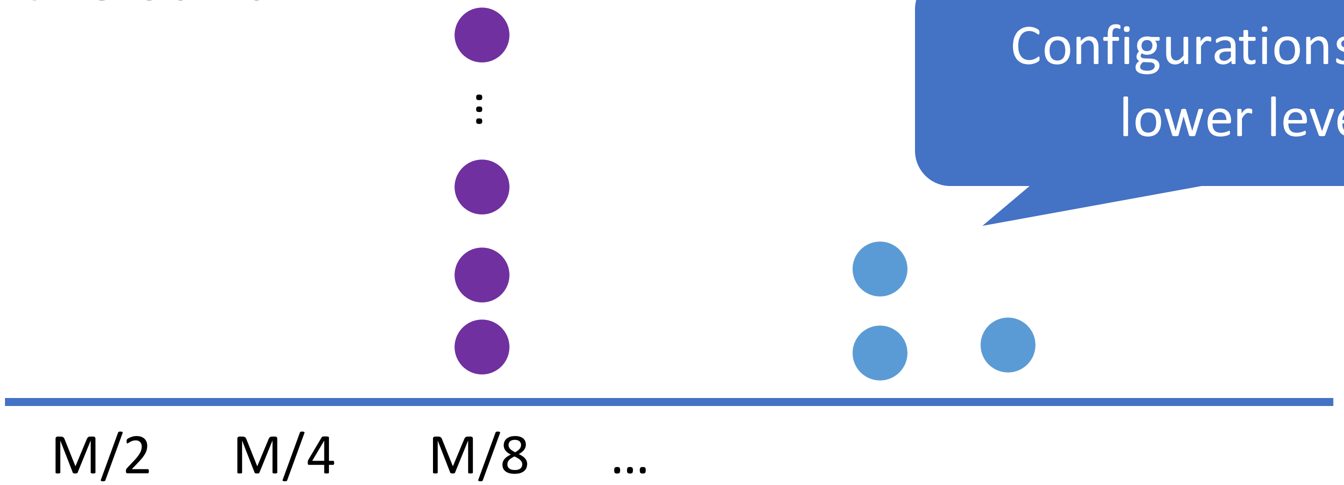
Fully split for at least $\log C$ levels

Proof idea for recursive splitting

- Analysis: “Toy in the sand”

Configurations at the lower levels

The resulting multi-set



Fully split for at least $\log C$ levels

As long as there are many balls at the “highest” level, then after the recursive splitting, any configuration at the lower levels will be smoothed out

Proof idea for the inner-outer construction

- Step 0. Understand shuffling: balls-and-bins formulation
- Step 1. A hammer for analysis: edit distance
- Step 2. Understand the histogram: outer PIR sub-queries, inner PIR sub-queries, and the relation between them
- Step 3. “Toy in sand” problem: hiding the shape of the toy

Proof idea for the inner-outer construction

- Step 0. Understand shuffling: balls-and-bins formulation

$|\mathcal{Q}_{\mathrm{OPIR}}|$ bins

$|\mathcal{Q}_{\mathrm{IPIR}}|$ bins

$\mathcal{Q}_{\mathrm{OPIR}}$: sub-query space of outer PIR

$\mathcal{Q}_{\mathrm{IPIR}}$: sub-query space of inner PIR

Proof idea for the inner-outer construction

- Step 0. Understand shuffling: balls-and-bins formulation
- Step 1. A hammer for analysis: edit distance
- Step 2. Understand the histogram: outer PIR sub-queries, inner PIR sub-queries, and the relation between them
- Step 3. “Toy in sand” problem: hiding the shape of the toy

Proof idea for the inner-outer construction

- Step 0. Understand shuffling: balls-and-bins formulation
- Step 1. A hammer for analysis: edit distance

Proof idea for the inner-outer construction

- Step 0. Understand shuffling: balls-and-bins formulation
- Step 1. A hammer for analysis: edit distance

Proof idea for the inner-outer construction

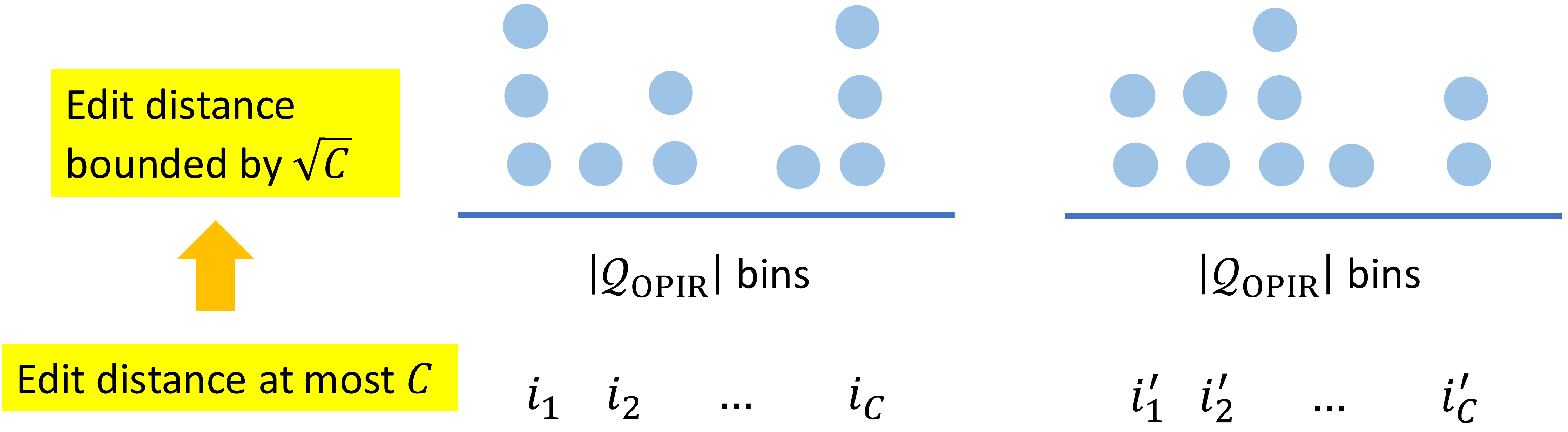
- Step 0. Understand shuffling: balls-and-bins formulation
- Step 1. A hammer for analysis: edit distance

Proof idea for the inner-outer construction

- Step 0. Understand shuffling: balls-and-bins formulation
- Step 1. A hammer for analysis: edit distance
- Step 2. Understand the histogram: outer PIR sub-queries, inner PIR sub-queries, and the relation between them
- Step 3. “Toy in sand” problem: hiding the shape of the toy

Proof idea for the inner-outer construction

- Step 2. Understand the histogram of **outer PIR sub-queries**



Proof idea for the inner-outer construction

- Step 2. **inner PIR sub-queries resultant from outer PIR sub-queries**

Plug in the previous result:
edit distance bounded by $C^{\frac{1}{4}}$

The 2-share histograms:
edit distance $\sqrt{\delta}$

If edit distance is δ

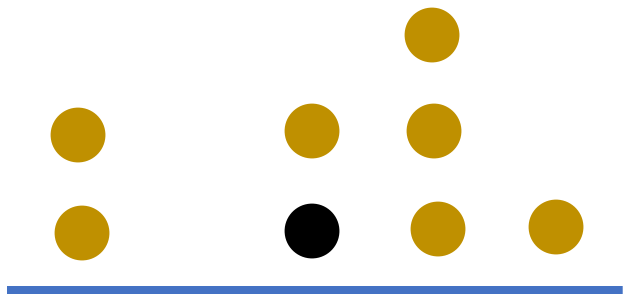
Resultant histograms after the 2-share

$|\mathcal{Q}_{\mathrm{IPIR}}|$ bins $|\mathcal{Q}_{\mathrm{IPIR}}|$ bins

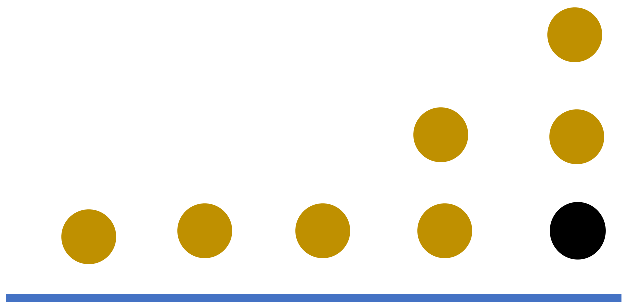
Proof idea for the inner-outer construction

- Step 0. Understand shuffling: balls-and-bins formulation
- Step 1. A hammer for analysis: edit distance
- Step 2. Understand the histogram: the relation between outer PIR sub-queries and inner PIR sub-queries
- Step 3. “Toy in sand” problem: hiding the shape of the toy

$$\mathrm{SD}(\mathcal{D}_i, \mathcal{D}_j) \leq \sqrt{\frac{\#bins}{\#balls}}$$



$\mathcal{D}_i$



$\mathcal{D}_j$

Proof idea for the inner-outer construction

- Step 0. Understand shuffling: balls-and-bins formulation
- Step 1. A hammer for analysis: edit distance
- Step 2. Understand the histogram: the relation between outer PIR sub-queries and inner PIR sub-queries
- Step 3. “Toy in sar shape of the toy

Let inner PIR sub-query space be Q

$$\mathrm{SD}(\mathcal{D}_i, \mathcal{D}_j) \le \sqrt{\frac{\#bins}{\#balls}} = \sqrt{\frac{Q}{C}} \Rightarrow \mathrm{SD}(\mathcal{D}, \mathcal{D}') \le C^{\frac{1}{4}} \cdot \sqrt{\frac{Q}{C}} = \frac{Q^{\frac{1}{2}}}{C^{\frac{1}{4}}}$$

Edit distance $C^{\frac{1}{4}}$