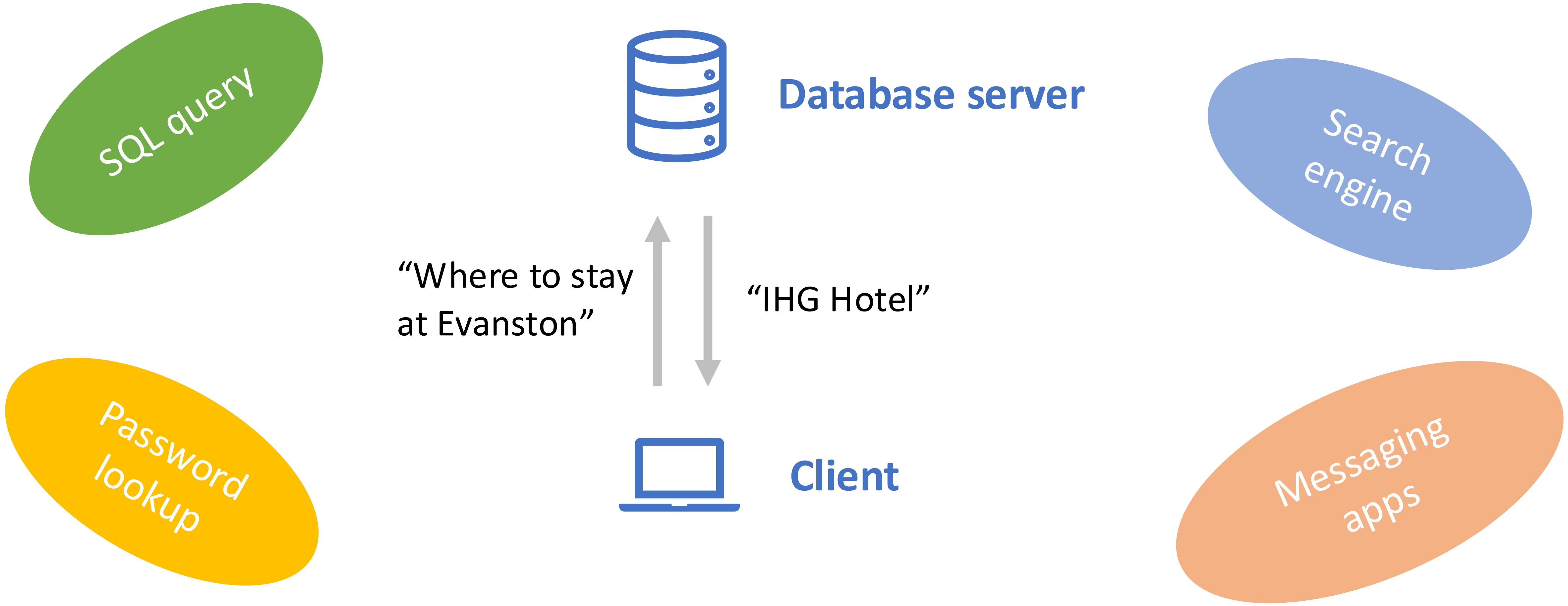
Single-Server Private Information Retrieval (PIR) in the Shuffle Model

Yiping Ma

Based on joint work with Adrià Gascón, Yuval Ishai,
Mahimna Kelkar, Daniel Lee, Baiyu Li, and Mariana Raykova

Google Technion Israel Institute of Technology Penn University of Pennsylvania Cornell Tech MIT

Information retrieval nowadays



Google Feud

Top 10 queries
on Google starting with

how do you get x

?	10,000	?	5,000
?	9,000	?	4,000
?	8,000	?	3,000
?	7,000	?	2,000
?	6,000	?	1,000

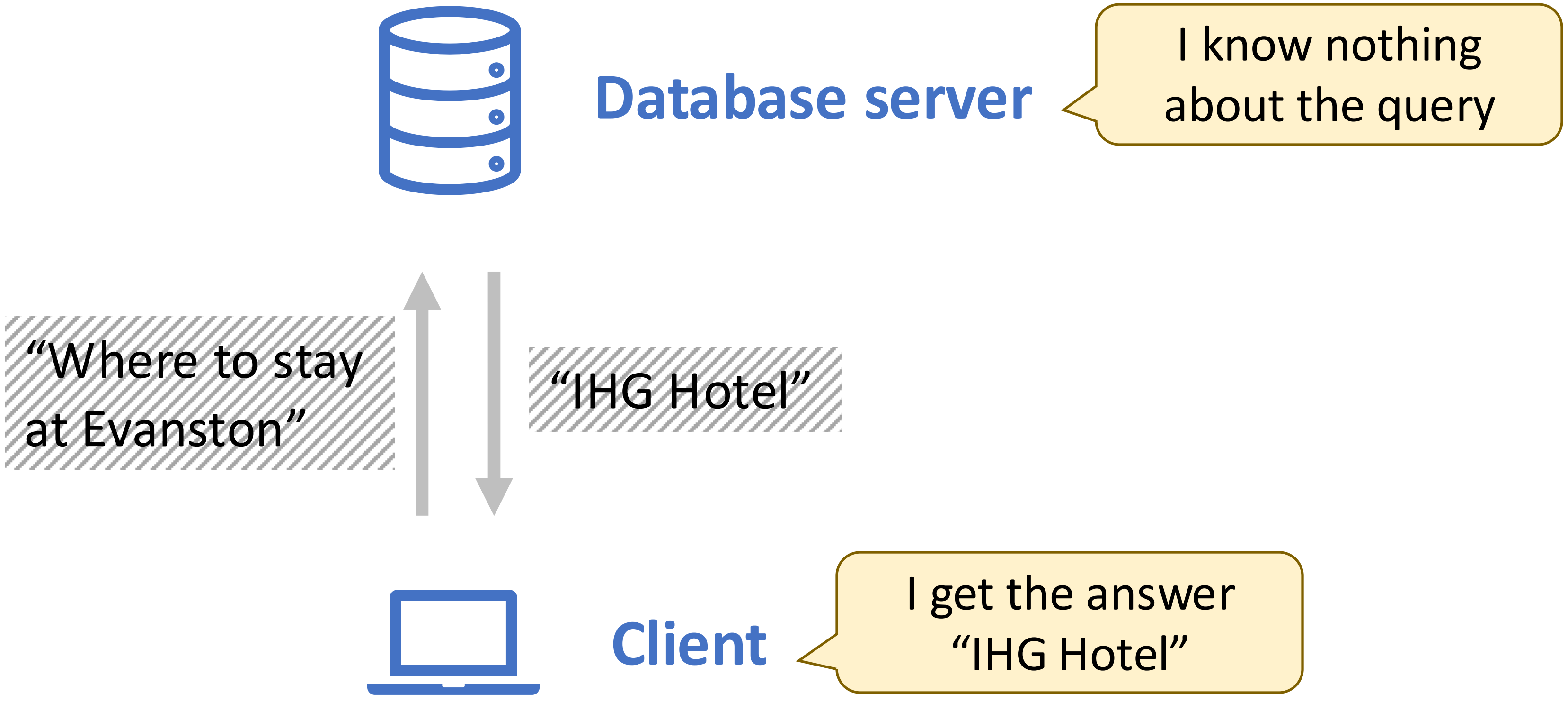
Google Feud

Top 10 queries
on Google starting with

how do you get x

pink eye	10,000	pneumonia	5,000
ringworm	9,000	shingles	4,000
a uti	8,000	rid of fruit flies	3,000
strep throat	7,000	mono	2,000
bed bugs	6,000	a yeast infection	1,000

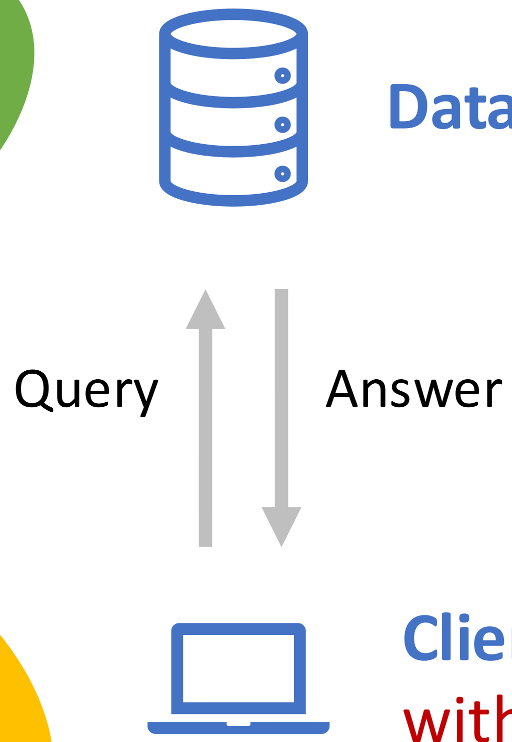
Private information retrieval (PIR) [CGKS95, KO97]



Private information retrieval (PIR) [CGKS95, KO97]

Encrypted search [Dory20]

Private search engine [Tiptoe23]



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

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

Secure password lookup [Checklist21]

Meta-data private messaging [Pung16]

The trivial solution is expensive

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

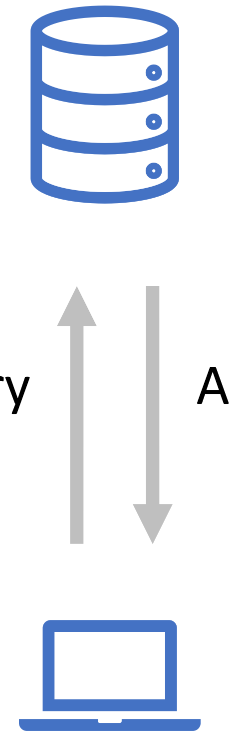
Download the whole database

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

Our efficiency goals

We want fast server computation

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



Query

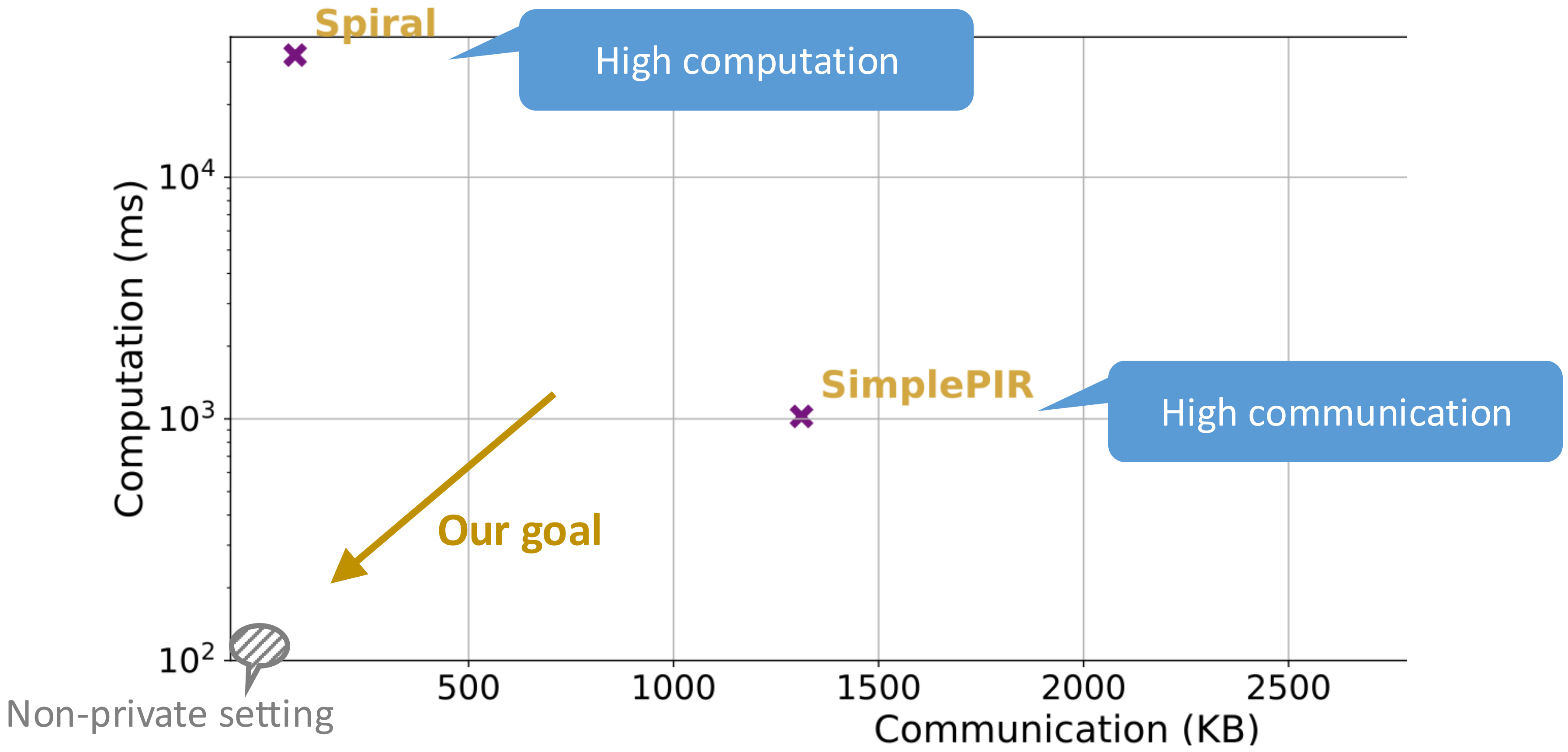
Answer

We want small communication

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

How far are we from the goals?

8GB database: 2^{18} entries of 32KB (size of a pdf document of a few pages)



Background: PIR in two flavors

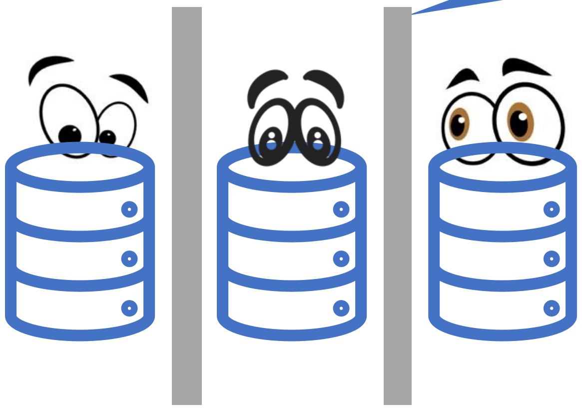
	Information-theoretic	Computational
Security	• Secure against unbounded adversaries	• Secure against poly-time adversaries

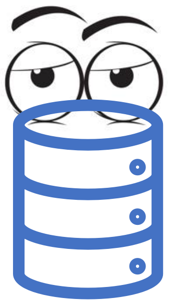
Single-server IT PIR is only possible when we allow n bits of communication [CGKS95]

Background: PIR in two flavors

	Information-theoretic	Computational
Security	• Secure against unbounded adversaries	• Secure against poly-time adversaries
System assumption	• Enforce non-collusion among the database servers	• No need for non-colluding assumption on the database server

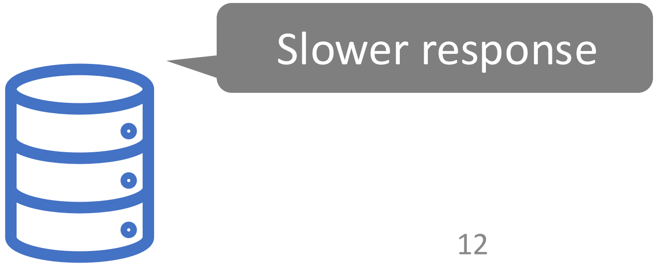
Hard to ensure when data is held by a single entity



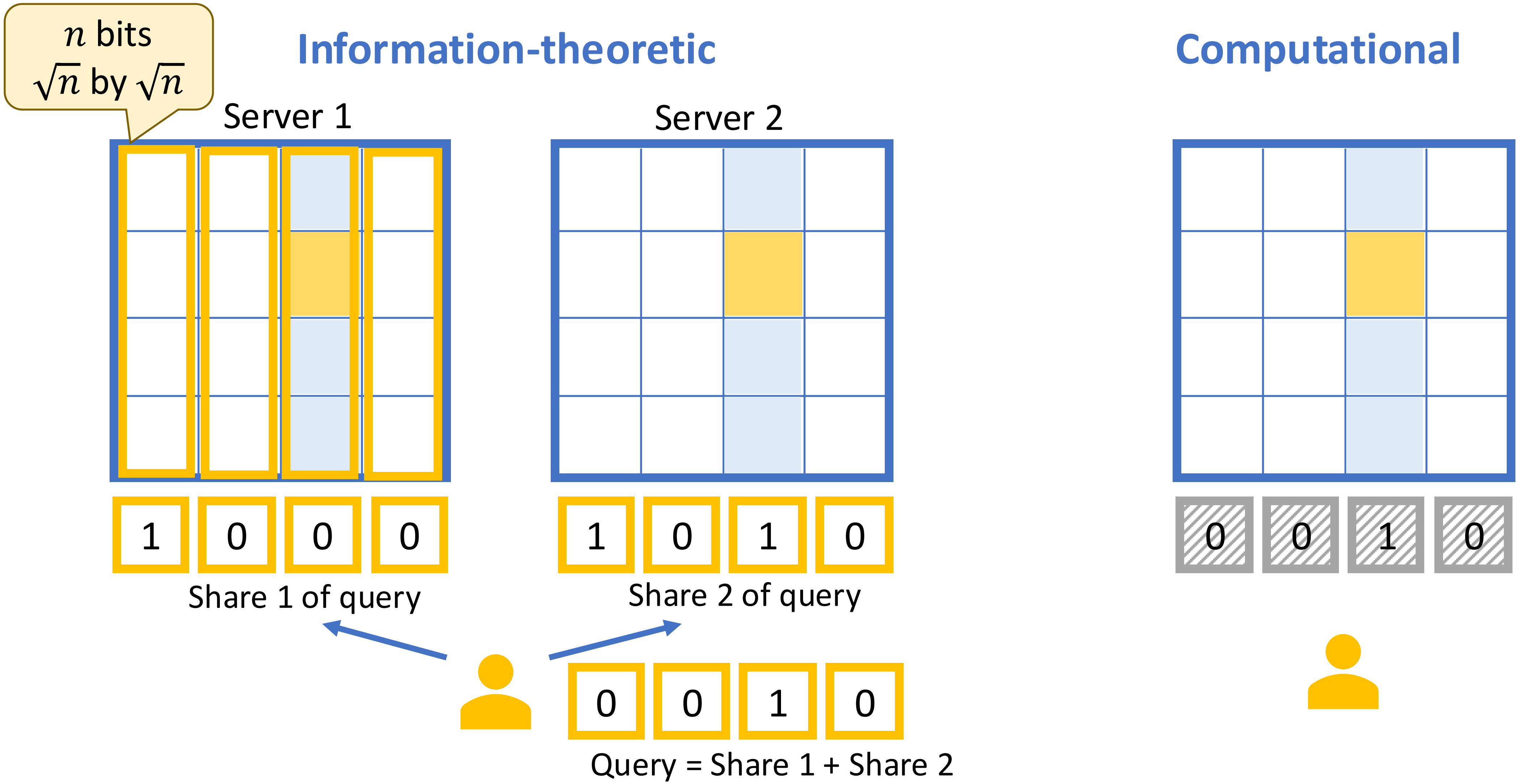


Background: PIR in two flavors

	Information-theoretic	Computational
Security	Secure against unbounded adversaries	Secure against poly-time adversaries
System assumption	Enforce non-collusion among the database servers	No need for non-colluding assumption on the database server
Efficiency (storage)	Require database replication across multiple servers	No database replication, a single server suffices
Efficiency (comp.)	Often efficient in practice (no cryptographic operations)	Expensive server cost because of cryptogaphic operations

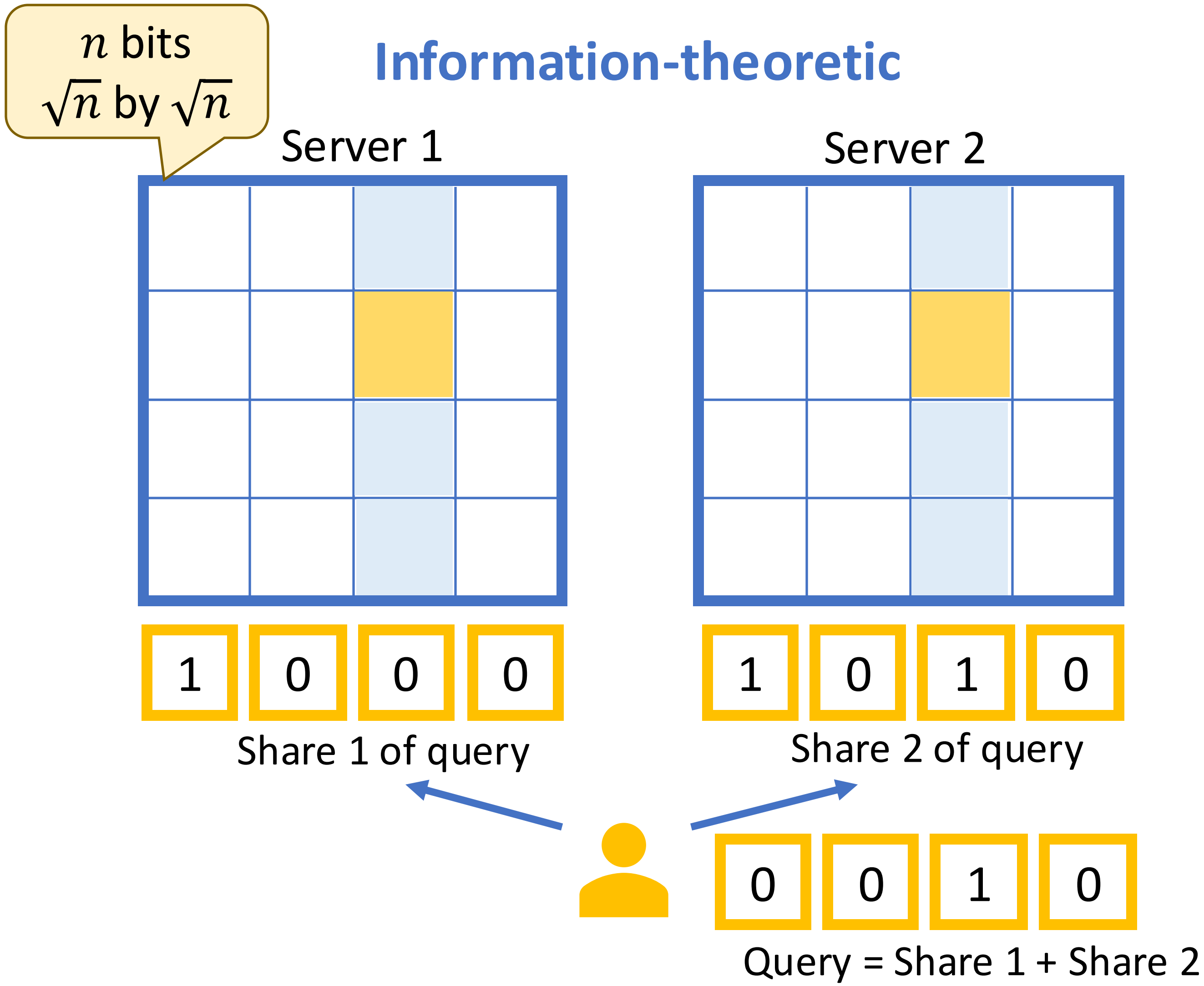


Background: PIR in two flavors

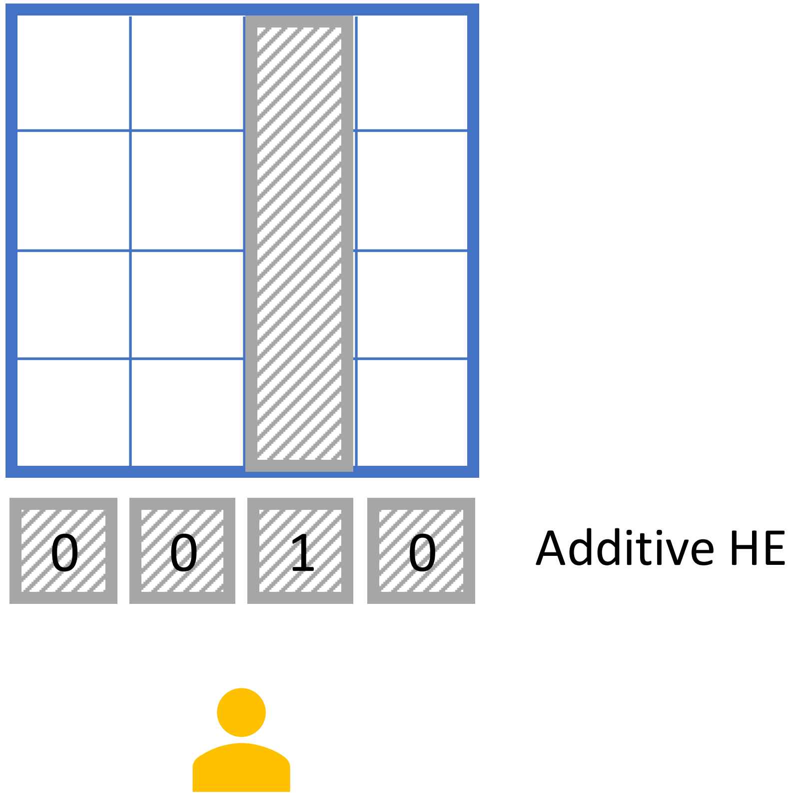


Background: PIR in two flavors

Information-theoretic



Computational



Best of both worlds?

	Information-theoretic	Computational
Security	Secure against unbounded adversaries	Secure against poly-time adversaries
System assumption	Enforce non-collusion among the database servers	No need for non-colluding assumption on the database server
Efficiency (storage)	Require database replication across multiple servers	No database replication, a single server suffices
Efficiency (comp.)	Often efficient in practice (no cryptographic operations)	Expensive server cost because of cryptogaphic operations

Do we have a sweet spot between security, efficiency and system assumption?

PIR in the shuffle model

- Security must hold for even **a single client**

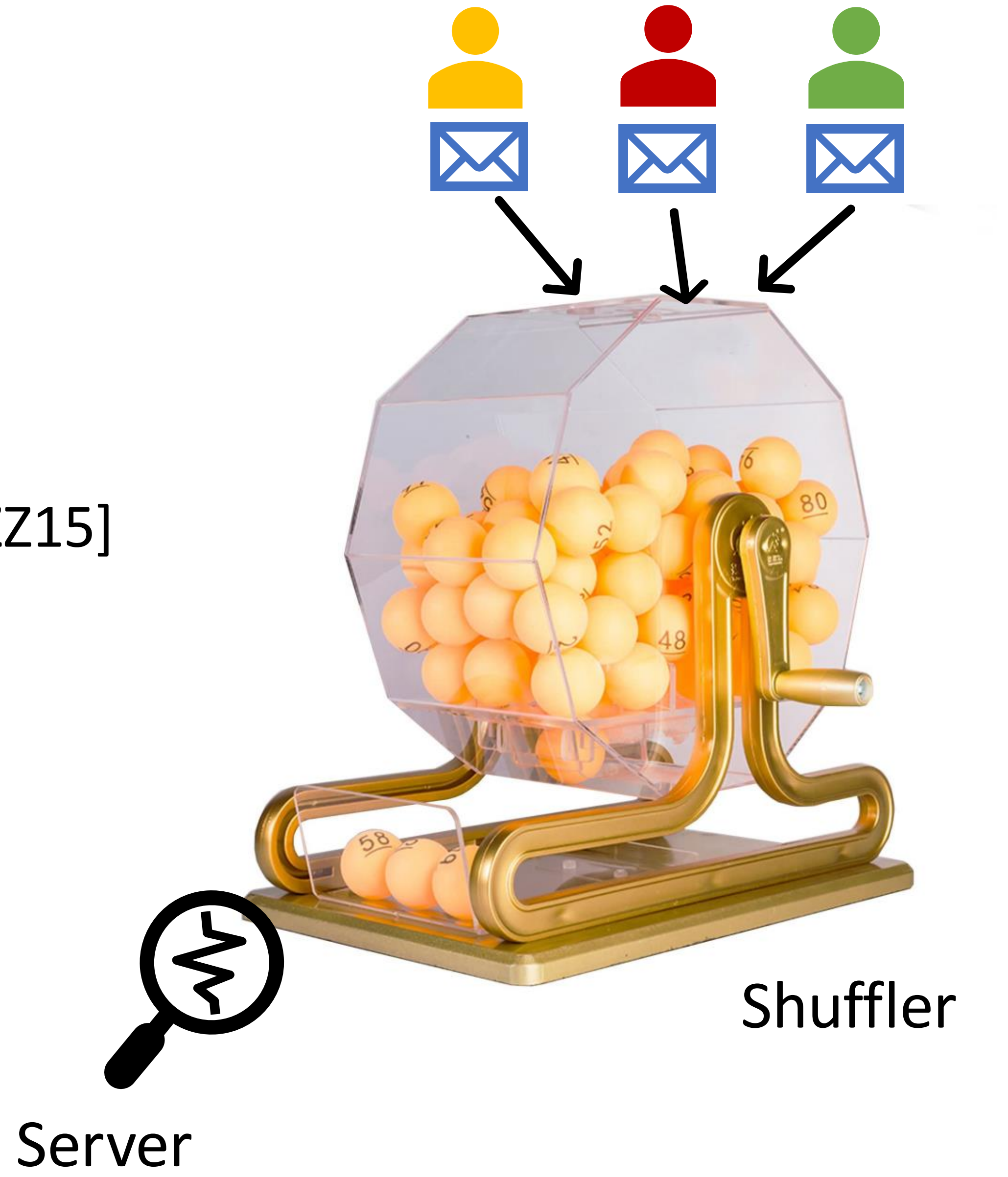
"The standard model"

- New hope: relaxed security 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

The shuffle model

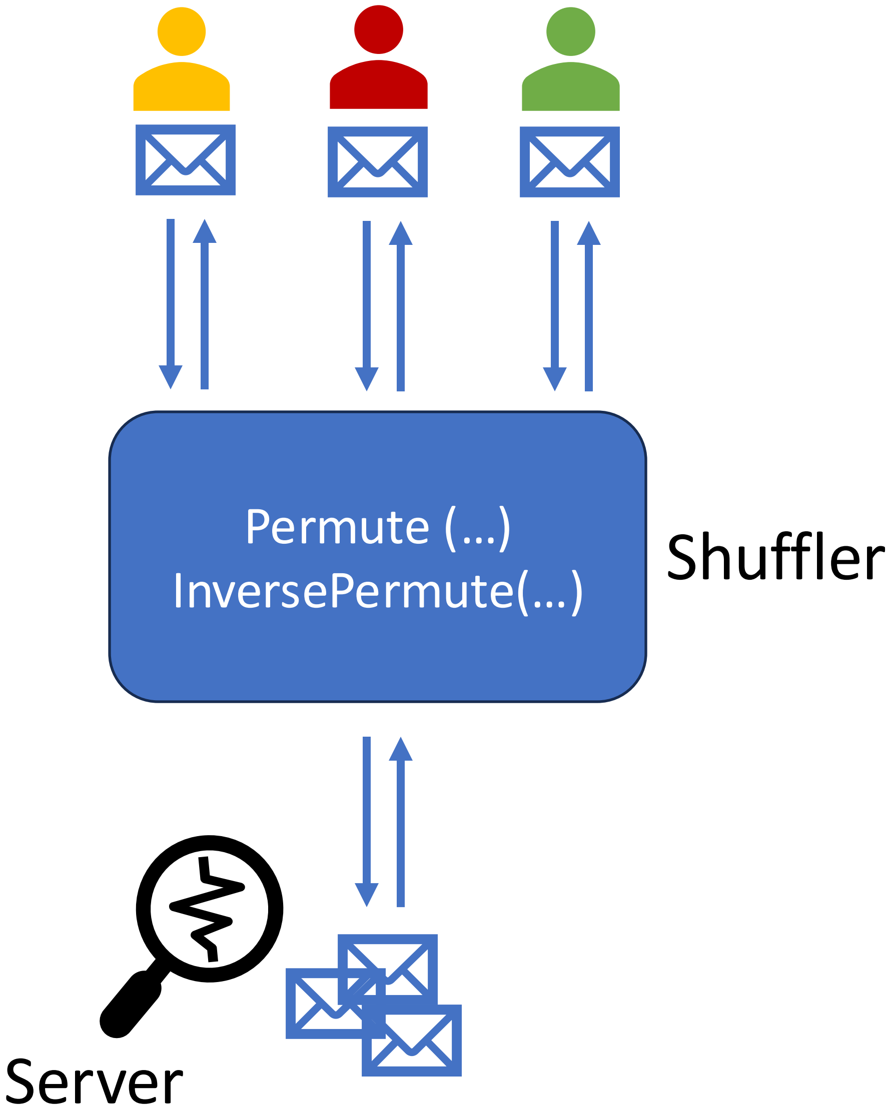
- Purpose: anonymization
- A popular model in
  - Anonymous communication, e.g., [HLZZ15]
  - Differential privacy, e.g., [BBGN20]



The shuffle model

- Purpose: anonymization
- A popular model in
  - Anonymous communication, e.g., [HLZZ15]
  - Differential privacy, e.g., [BBGN20]
- In our PIR setting:
  - We assume it is two-way
  - Can be instantiated by, e.g., Tor
  - Or can be viewed as a second shuffle server who does not hold the database

A hybrid model between single-server and two-server



PIR in the shuffle model: Our results

- Result 1: Single-server IT secure PIR with sublinear communication is theoretically feasible in the shuffle model
- Impossibility result [CGKS95]: For single-server IT-PIR in the standard model, the only way out is requiring n bits communication

PIR in the shuffle model: Our results

- Result 1

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 computation with $1/\mathrm{poly}(n)$ statistical security (assuming one-time preprocessing), as long as $\mathrm{poly}(n)$ clients simultaneously accessing the database.

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

PIR in the shuffle model: Our results

- Result 1: Single-server IT secure PIR with sublinear communication is theoretically feasible in the shuffle model

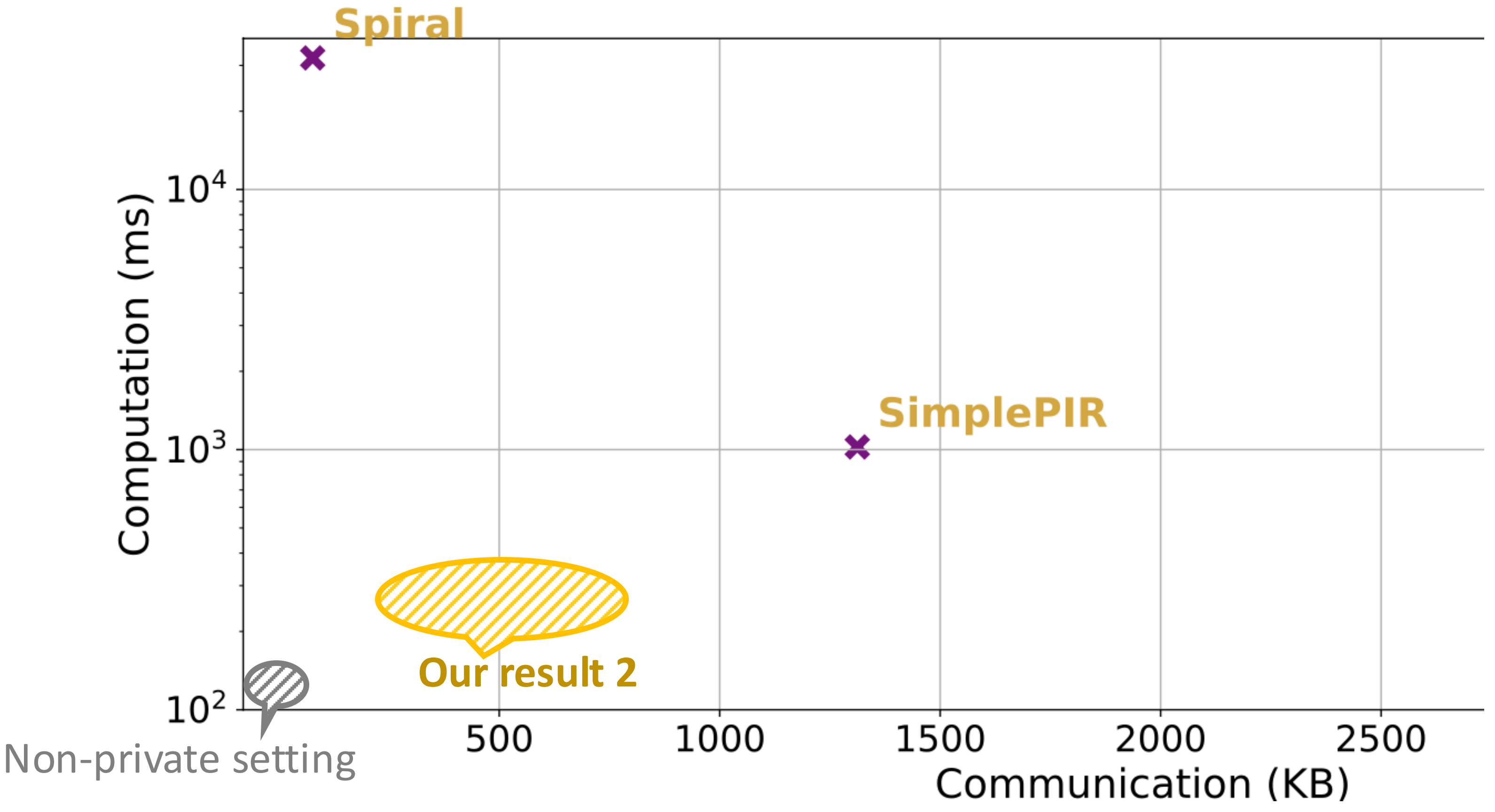
  Drawback of result 1: requiring too many clients querying at the same time

- Result 2: Single-server computationally secure PIR in the shuffle model that has concretely small communication and computation, and requires a reasonable number of simultaneously querying clients

[IKOS06] initialized the study of PIR from anonymity, but their construction relies on non-standard computational assumptions and is not concretely efficient.

Our result 2: a new design space

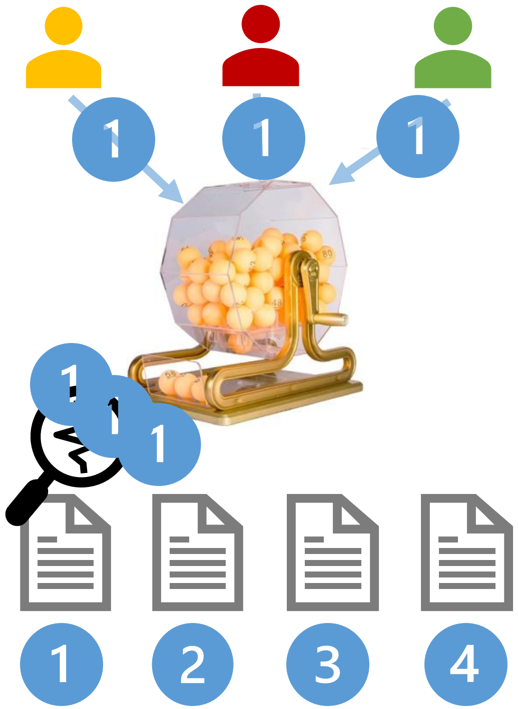
8GB database: 2^{18} entries of 32KB (size of a pdf document of a few pages)



Rest of this talk

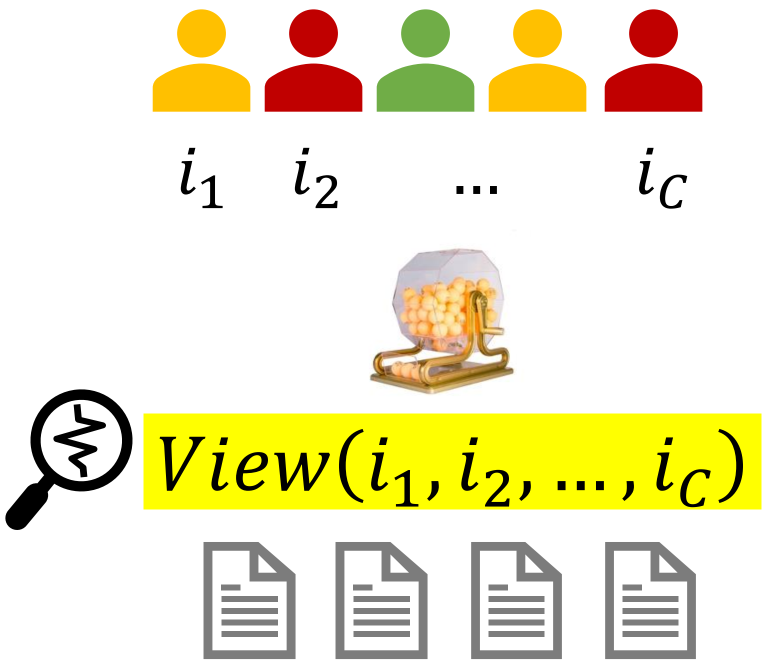
- Construction backbone: “Split and mix”
- Result 1: A generic construction of IT-PIR in the shuffle model
- Result 2: PIR from computationally secure split-and-mix
- An interesting orthogonal problem: hiding record size without padding
- Discussion and open questions

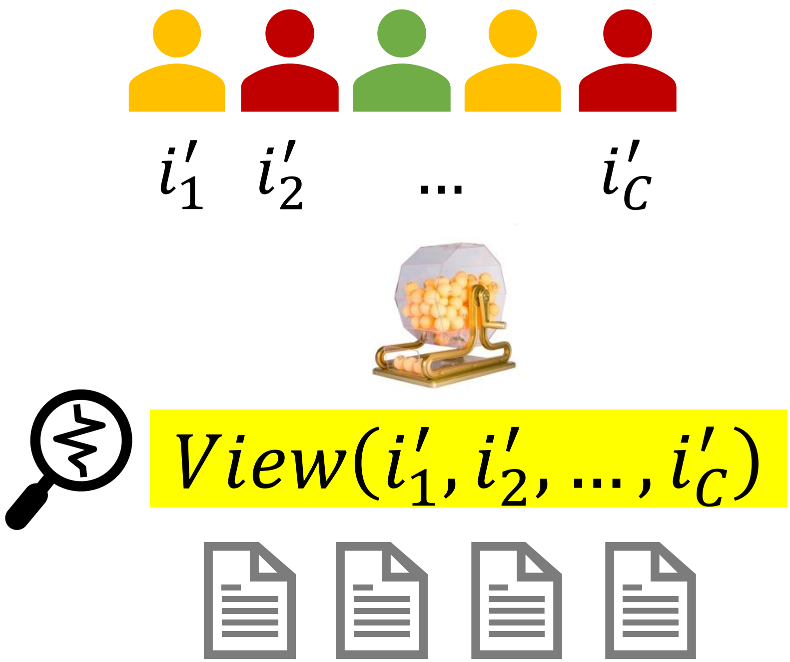
Anonymization does not trivialize the PIR problem



The shuffler hides who sends which message, but does not hide the message itself

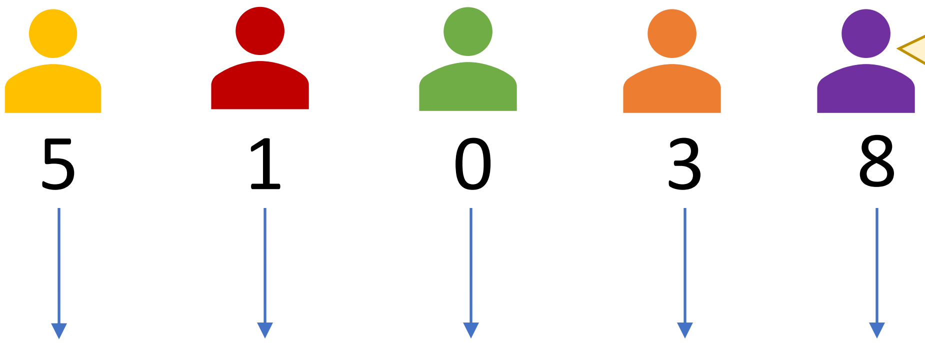
What we want for security





The split-and-mix paradigm [IKOS06]

A secure sum problem

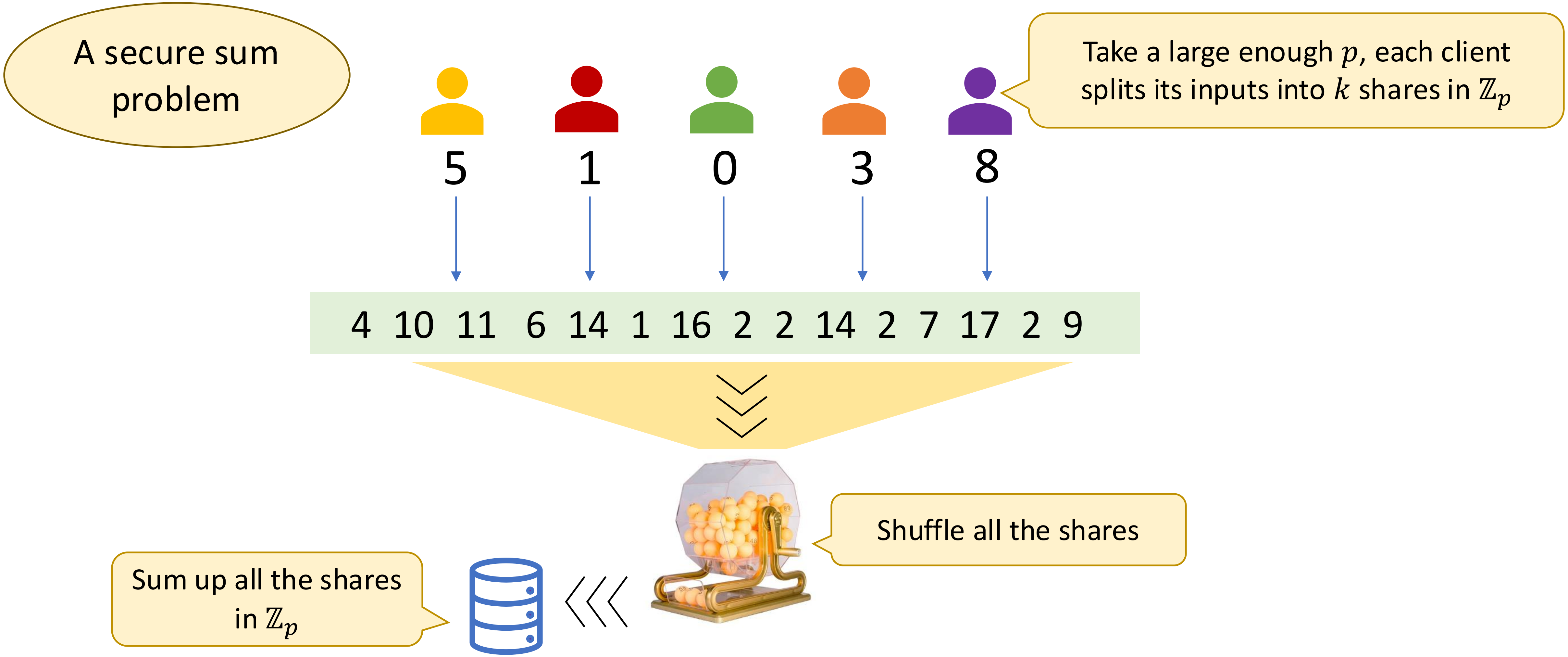


Take a large enough p, each client splits its inputs into k shares in $\mathbb{Z}_p$

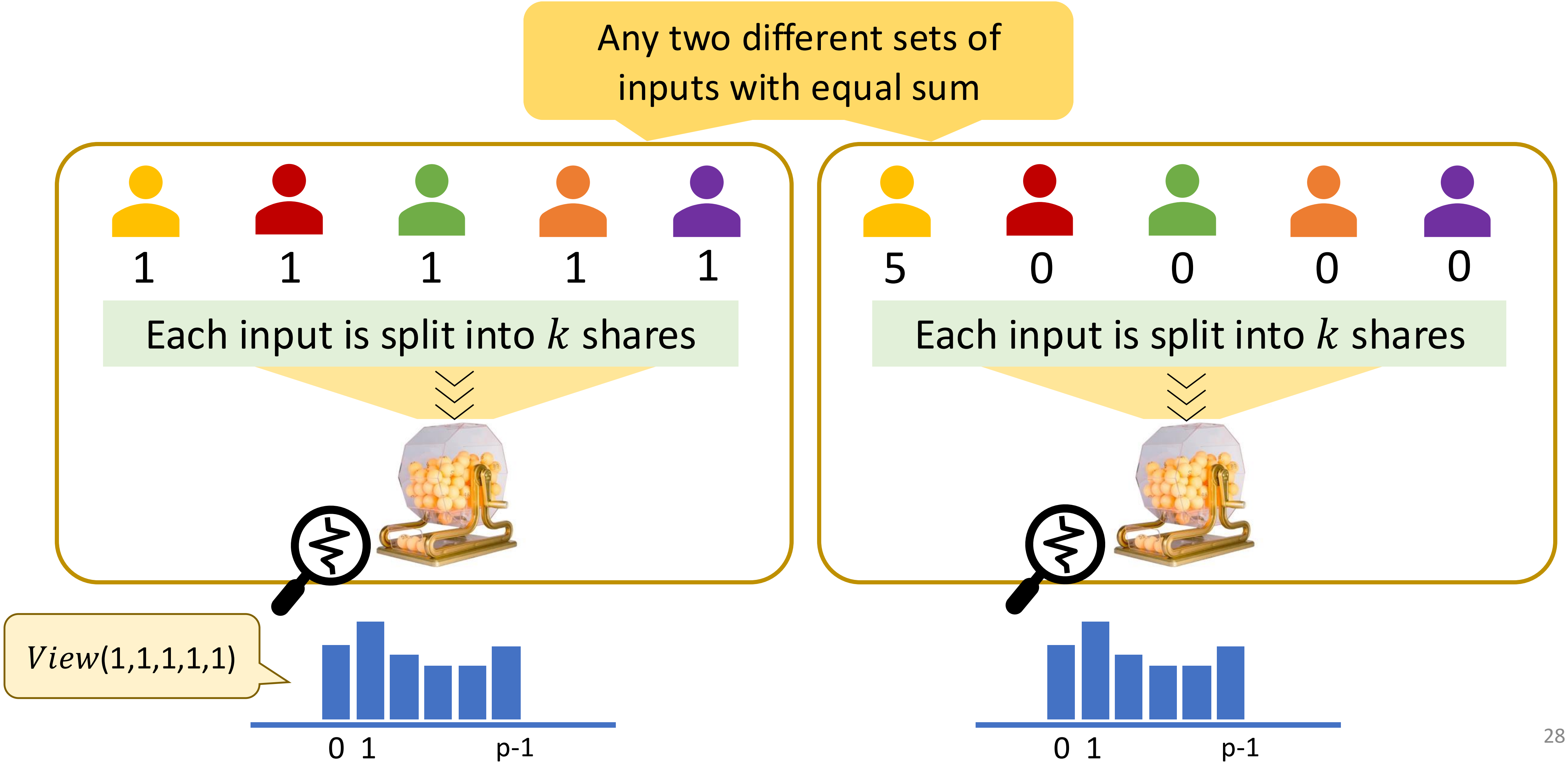
4+10+11 6+14+1 16+2+2 14+2+7 17+2+9

E.g., $p = 20,\ k = 3$

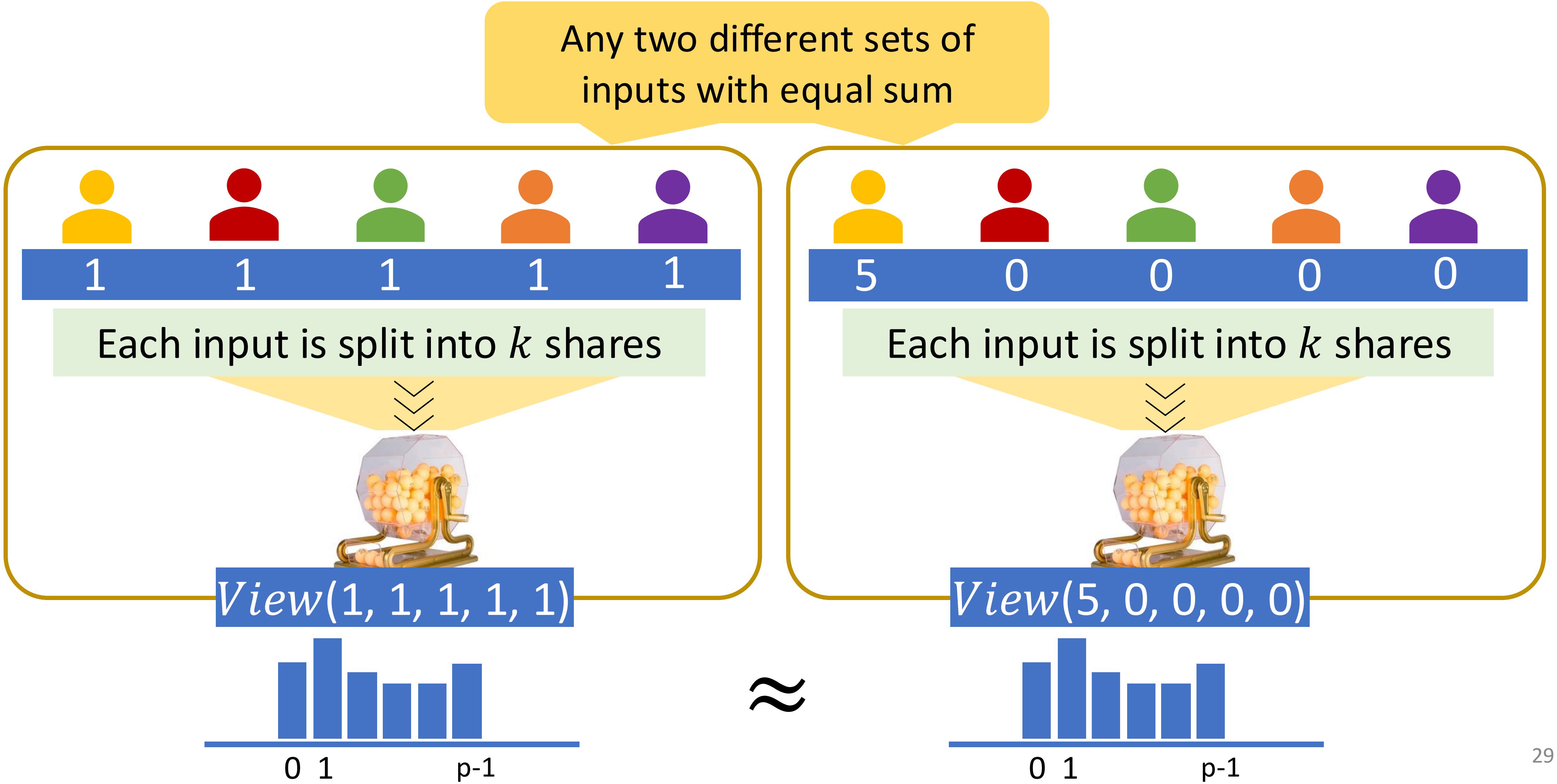
The split-and-mix paradigm [IKOS06]



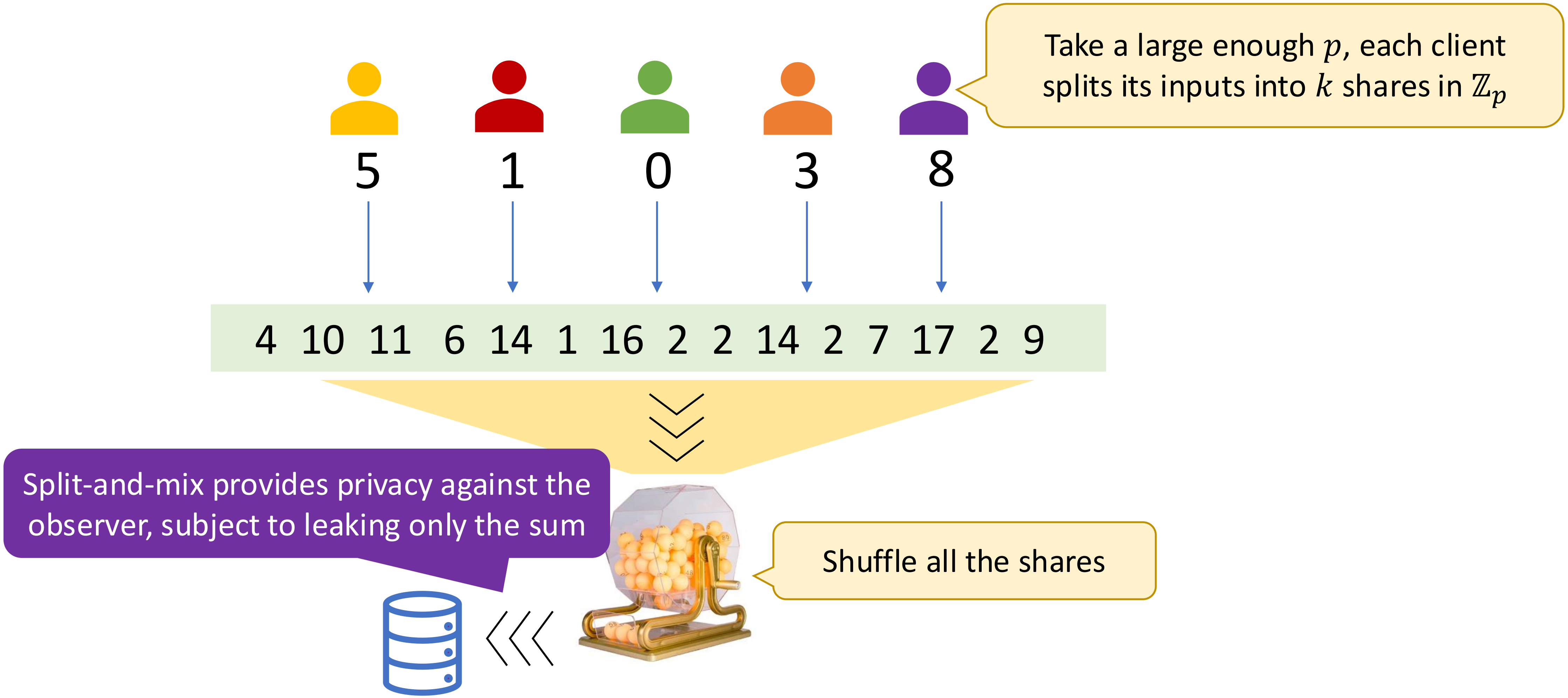
Security formalization of split-and-mix



Security formalization of split-and-mix



Split-and-mix as a tool

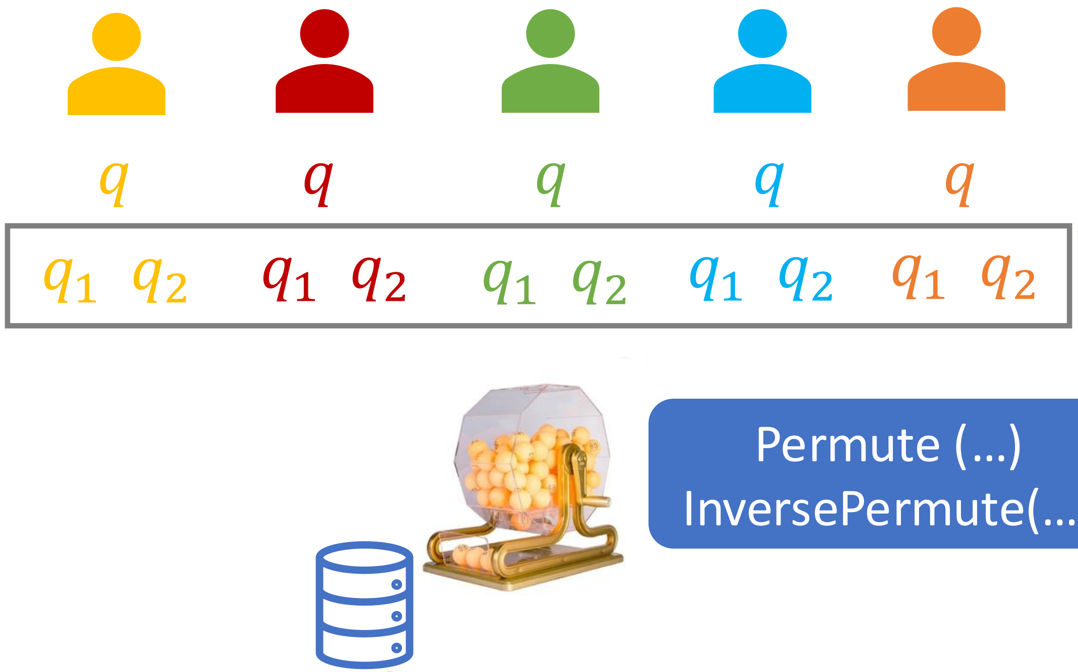


Rest of this talk

- Construction backbone: “Split and mix”
- Result 1: A generic construction of IT-PIR in the shuffle model
- Result 2: PIR from computationally secure split-and-mix
- An interesting orthogonal problem: hiding record size without padding
- Discussion and open questions

IT-PIR from split-and-mix

The key idea is to view the multi-server PIR in the additive sharing paradigm



Split each query into additive shares?

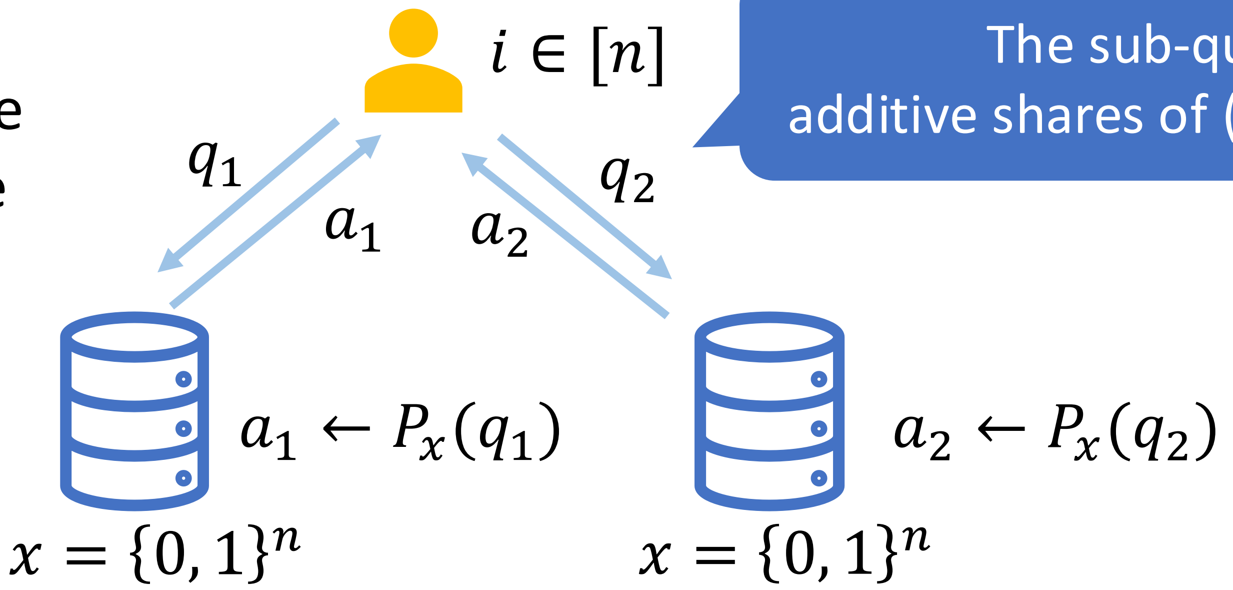
Permute (…)
InversePermute(…)

Answer to each share

IT-PIR from split-and-mix

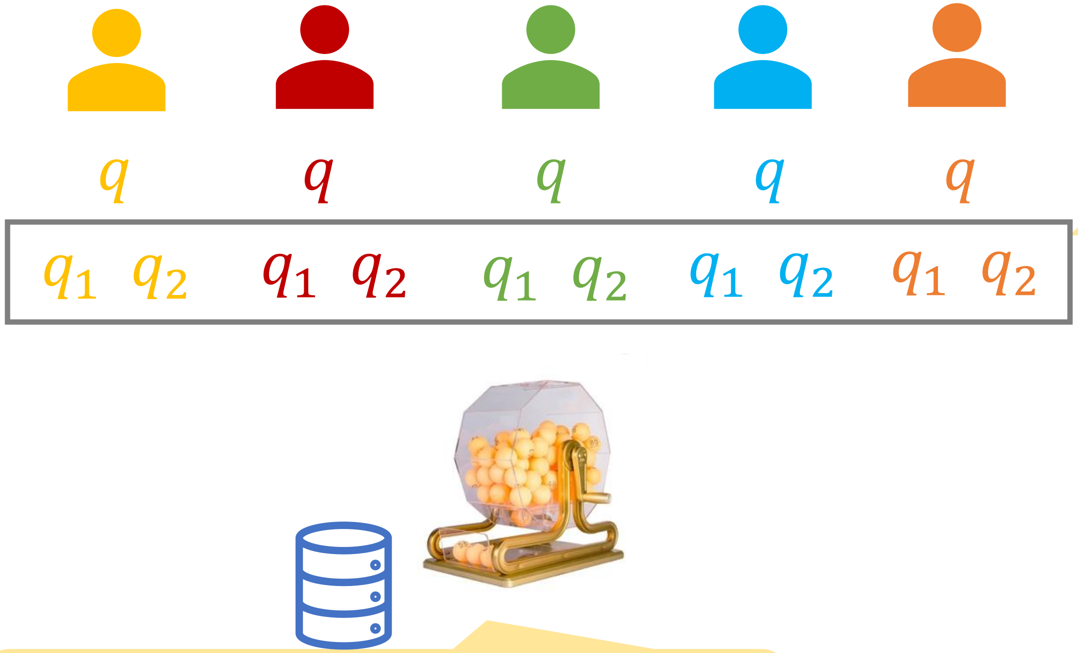
- Sub-queries are additive shares
- Answer algorithm is simply P_x(share)

[BIK04]
$O(\log n)$ query size
$O(\sqrt{n})$ answer size



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

IT-PIR from split-and-mix



Query using the two-server “additive PIR” protocol

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

Are we done?

IT-PIR from split-and-mix

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!

IT-PIR from split-and-mix

- **Can we split to more shares?** Yes, but worse efficiency:

  The k-server "additive PIR" in [BIK04] 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 $O(n^{\frac{1}{2}} \operatorname{polylog}(n))$

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

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

\$

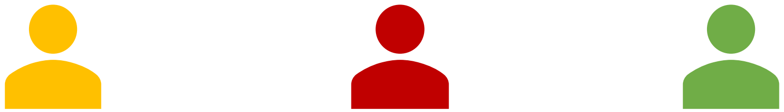
If we can make $i_1, i_2, \dots, i_C$ and $i'_1, i'_2, \dots, i'_C$ closer, e.g., 1 2 3 4 4 v.s. 1 2 3 4 5

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

Our proof technique

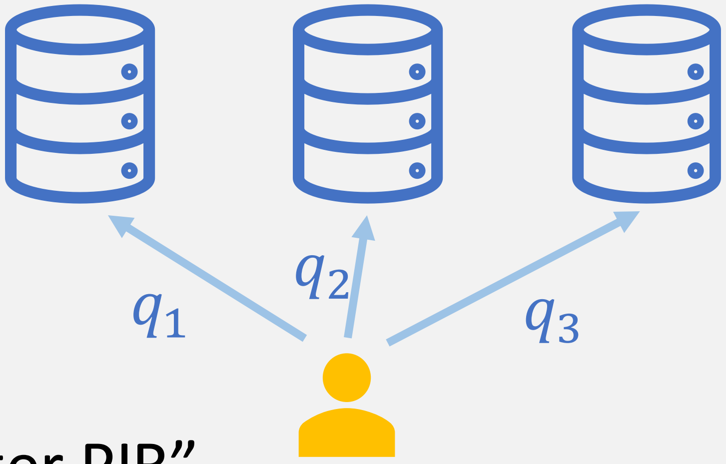
General constructions: an “inner-outer” paradigm

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 e.g., q_1 is uniformly random over Q

General constructions: an “inner-outer” paradigm

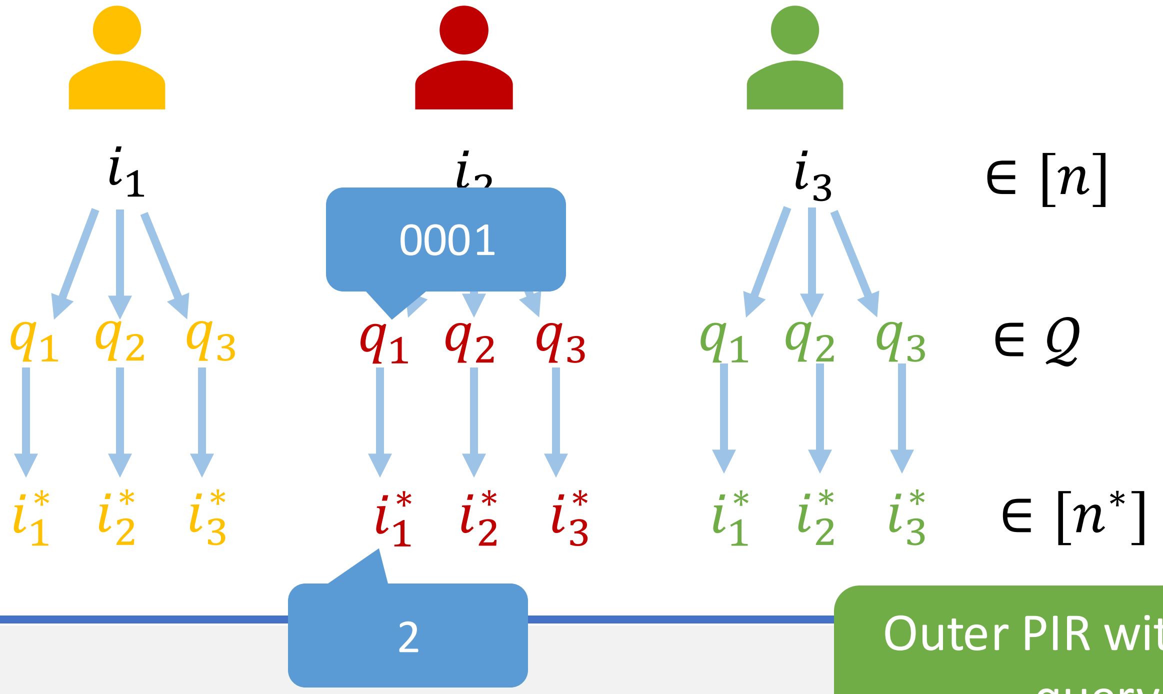
How to randomize the indices?

Run outer PI

Each random in Q

Interpret a

Each random in $[n^*]$



What we get from outer PIR

Outer PIR with $O(\log n)$ query size

A list with size $n^* = |Q|$

Sort all sub-queries in Q

0000	0001	0010	0011	…	1111
1	2	3	4	…	n^*

General constructions: an "inner-outer" paradigm

Recall: not secure if doing "additive PIR" directly here

How to randomize the indices?

i_1 i_2 i_3 $\in [n]$

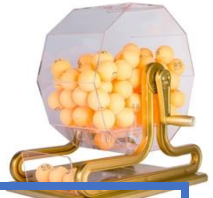
Run outer PIR query algorithm

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^* $\in [n^*]$

Use the two-server "additive" PIR



Inner PIR database size $n^* = |\mathcal{Q}|$

$P_x(0000)$ $P_x(0001)$ $\cdots$ $P_x(1111)$

Answers in outer PIR

General constructions: an “inner-outer” paradigm

A brief summary

On any query indices

Run outer PIR query algorithm

Interpret as indices for inner PIR

Use inner PIR for retrieve answers;
Inner PIR sub-queries are shuffled

The server prepares this in advance

A single server!

i_1

Outer PIR: sub-query size $O(\log n)$

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

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

Inner PIR: The two-server “additive PIR”

$P_x(0000)$	$P_x(0001)$	$\cdots$	$P_x(1111)$

Size n^*

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

- Construction backbone: “Split and mix”
- Result 1: A generic construction of IT-PIR in the shuffle model
- Result 2: PIR from computationally secure split-and-mix
- An interesting orthogonal problem: hiding record size without padding
- Discussion and open questions

Recall the security of split-and-mix

$$View(1, 1, 1, 1, 1) \approx View(5, 0, 0, 0, 0)$$

- Prior works only study statistical security [IKOS06, GMPV20, BBGN20]

#Clients	100	1000	10000
#Shares k (IT. 40 bits)	6317	3856	2775

Each client input: a vector $2^{15} \times \mathbb{F}_2$

New: computational security for split-and-mix

$$\mathit{View}(1, 1, 1, 1, 1) \approx \mathit{View}(5, 0, 0, 0, 0)$$

- Prior works only study statistical security [IKOS06, GMPV20, BBGN20]
- This work studies computational security, aiming to reduce the #shares k (and hence improving concrete efficiency)

#Clients	100	1000	10000
#Shares k (IT. 40 bits)	6317	3856	2775
#Shares k (Comp. 128 bits)	405	88	37

Each client input: a vector $2^{15} \times \mathbb{F}_2$

Our results from computational split-and-mix

Computational security for split-and-mix based on SD, MDSD

Single-server secure aggregation in the shuffle model

Single-server PIR in the shuffle model

Up to 25X savings for communication compared to the best statistical split-and-mix baseline

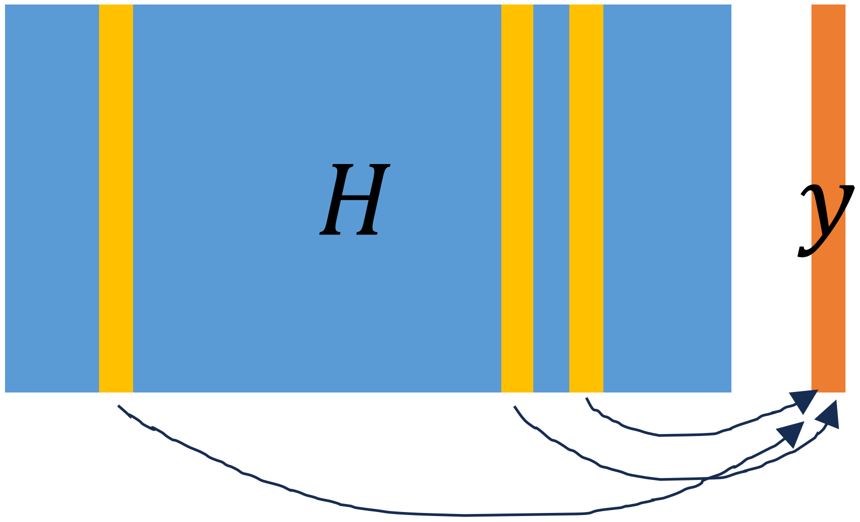
(Even giving advantage to the baseline by compressing the shares)

Up to 22X improvement of throughput (in the batch setting) over SimplePIR [HHCMV23] with comparable communication cost

Split-and-mix based on Syndrome Decoding (SD)

- The SD assumption (dual-LPN [BFKL94, AIK07])
  H: a random matrix
  y: a target vector (e.g., a client's input)

Computationally hard to find low-weight vector e such that $H \cdot e = y$



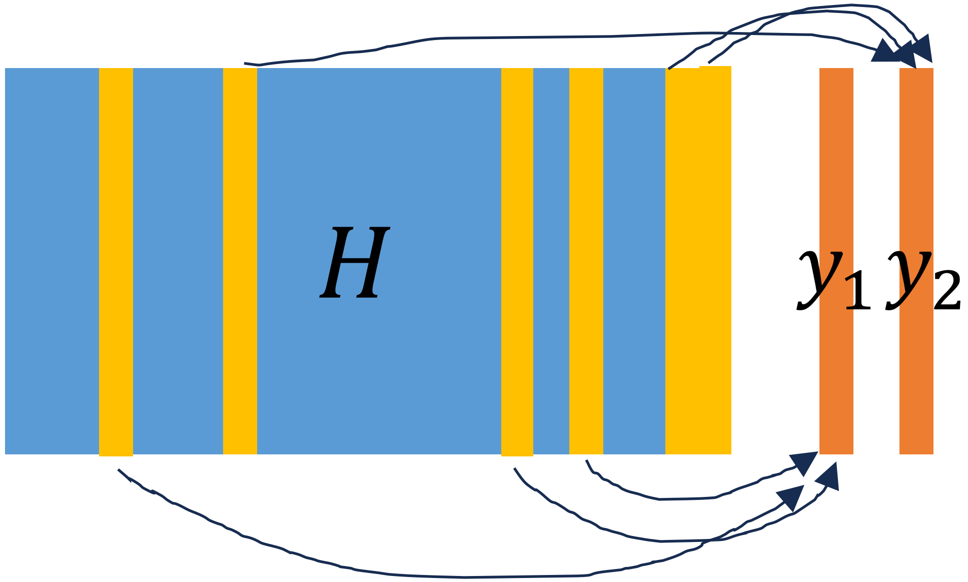
Split-and-mix based on Syndrome Decoding (SD)

- "Multi-Disjoint" Syndrome Decoding
  H: a random matrix
  $Y = [y_1, y_2, \ldots]$: multiple target vectors (e.g., multiple client inputs)

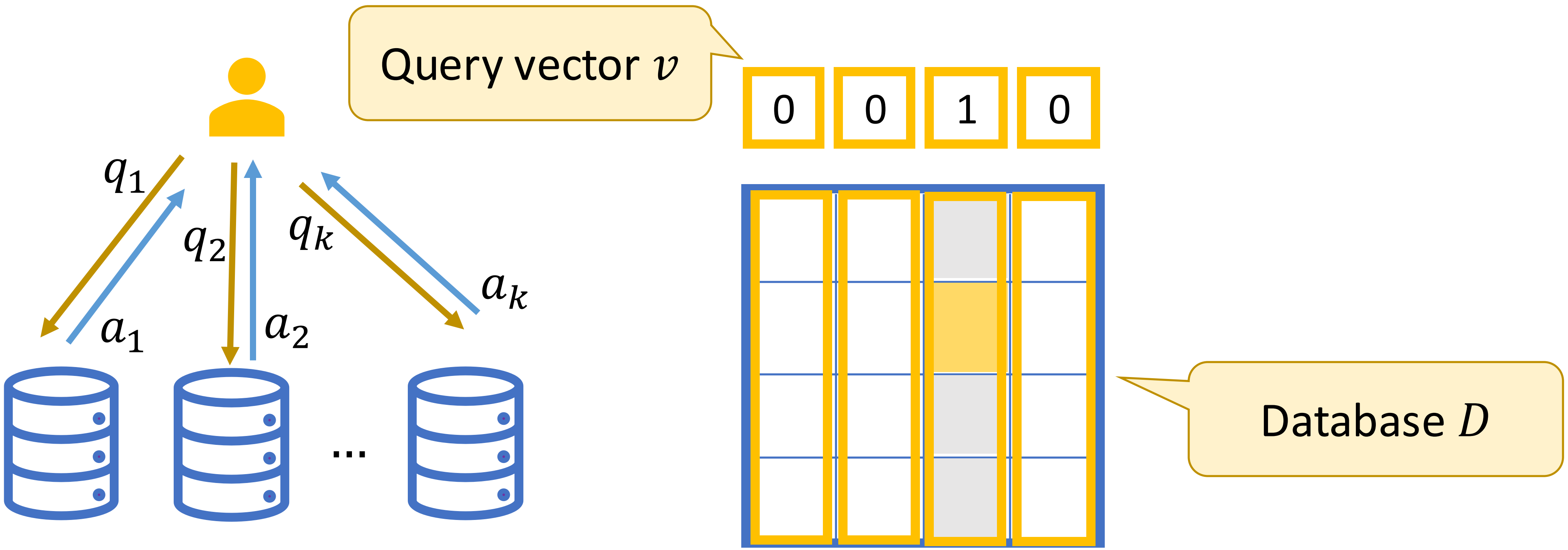
The positions of 1 in E's columns are disjoint

Computationally hard to find "low-weight" E such that $H \cdot E = Y$

We generalize SD to **M**ulti-**D**isjoint **S**yndrome **D**ecoding to handle multiple clients



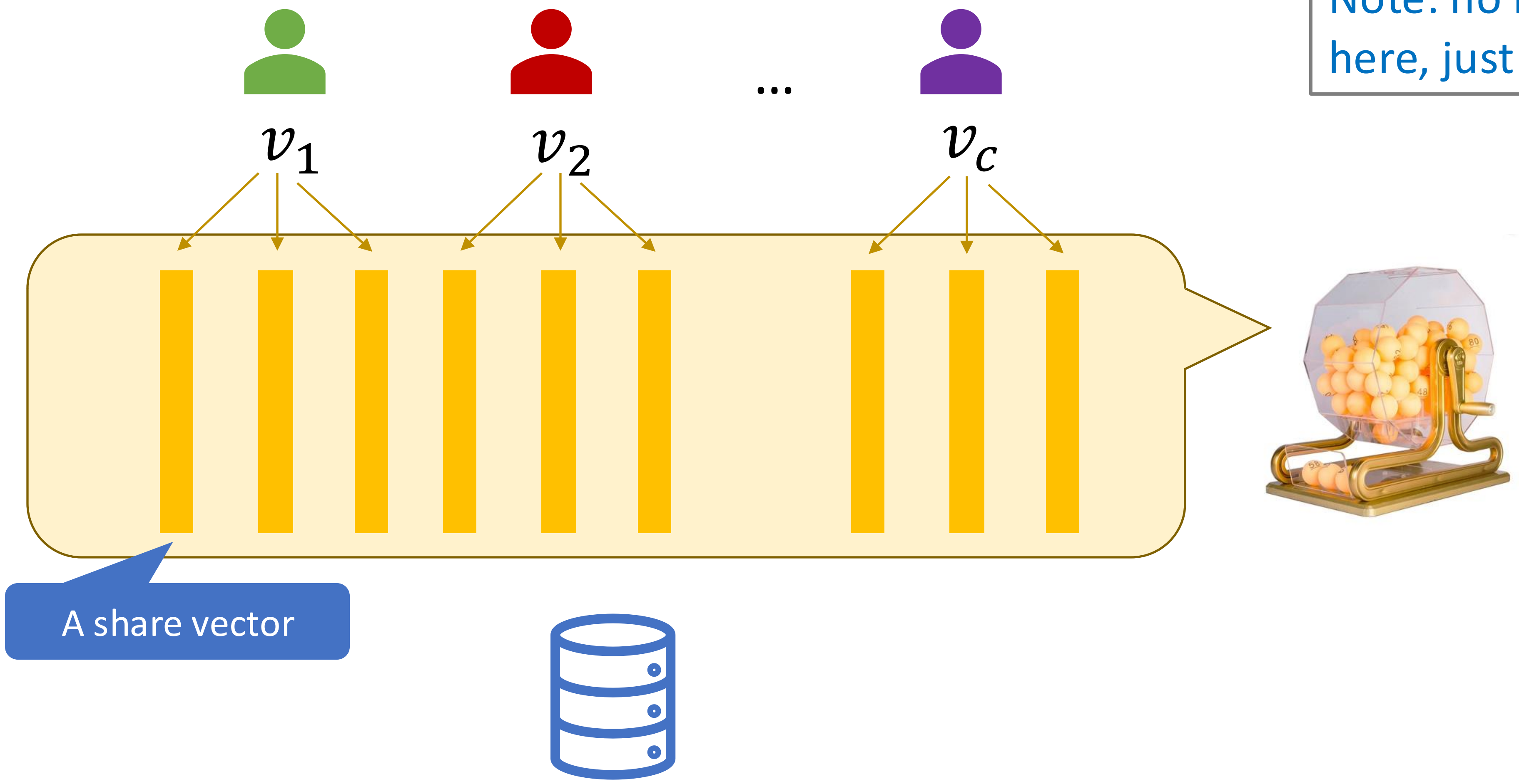
Starting point: a classic multi-server PIR



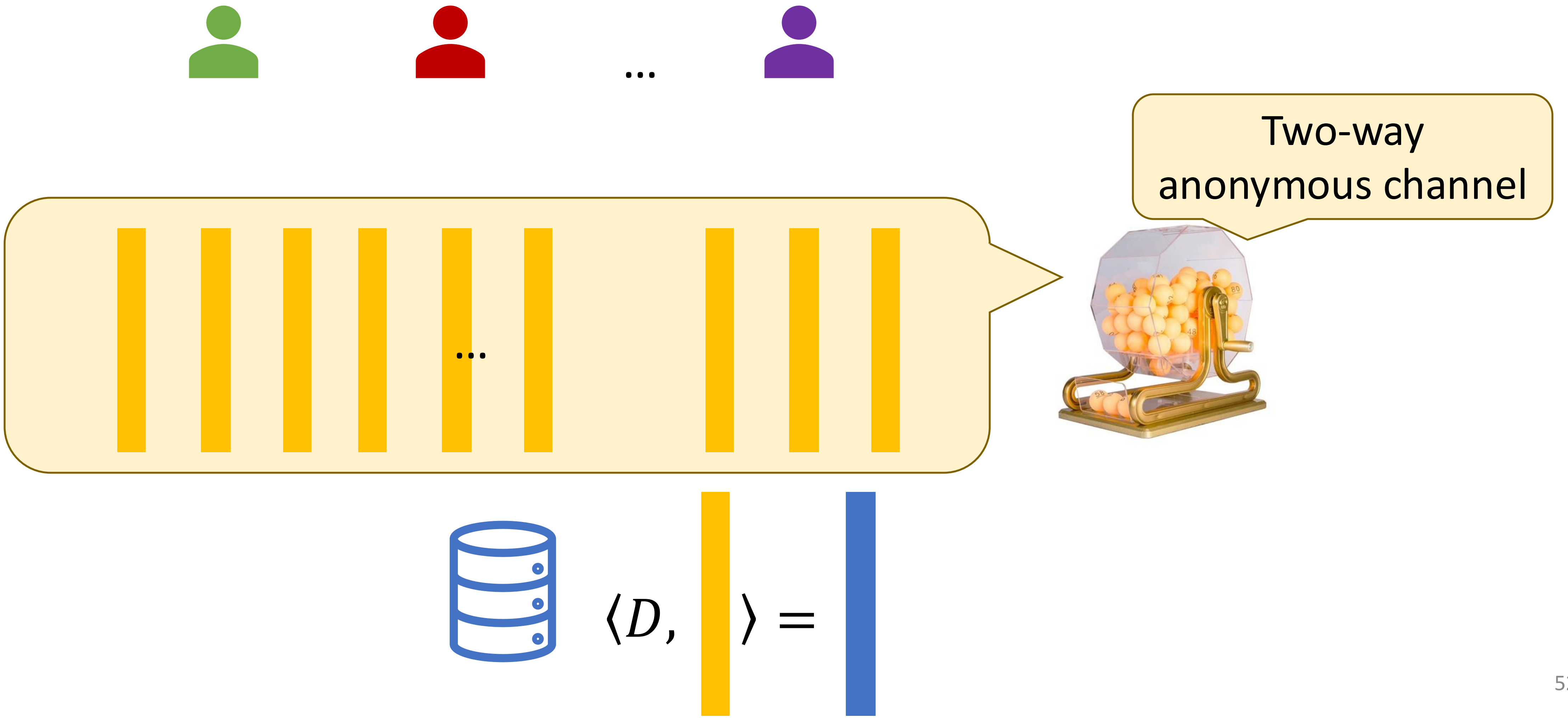
$$\begin{aligned} v \cdot D &= (q_1 + \cdots + q_k) \cdot D \\ &= q_1 \cdot D + \cdots + q_k \cdot D \\ &\coloneqq a_1 + \cdots + a_k \end{aligned}$$

Single-server computationally PIR from split-and-mix

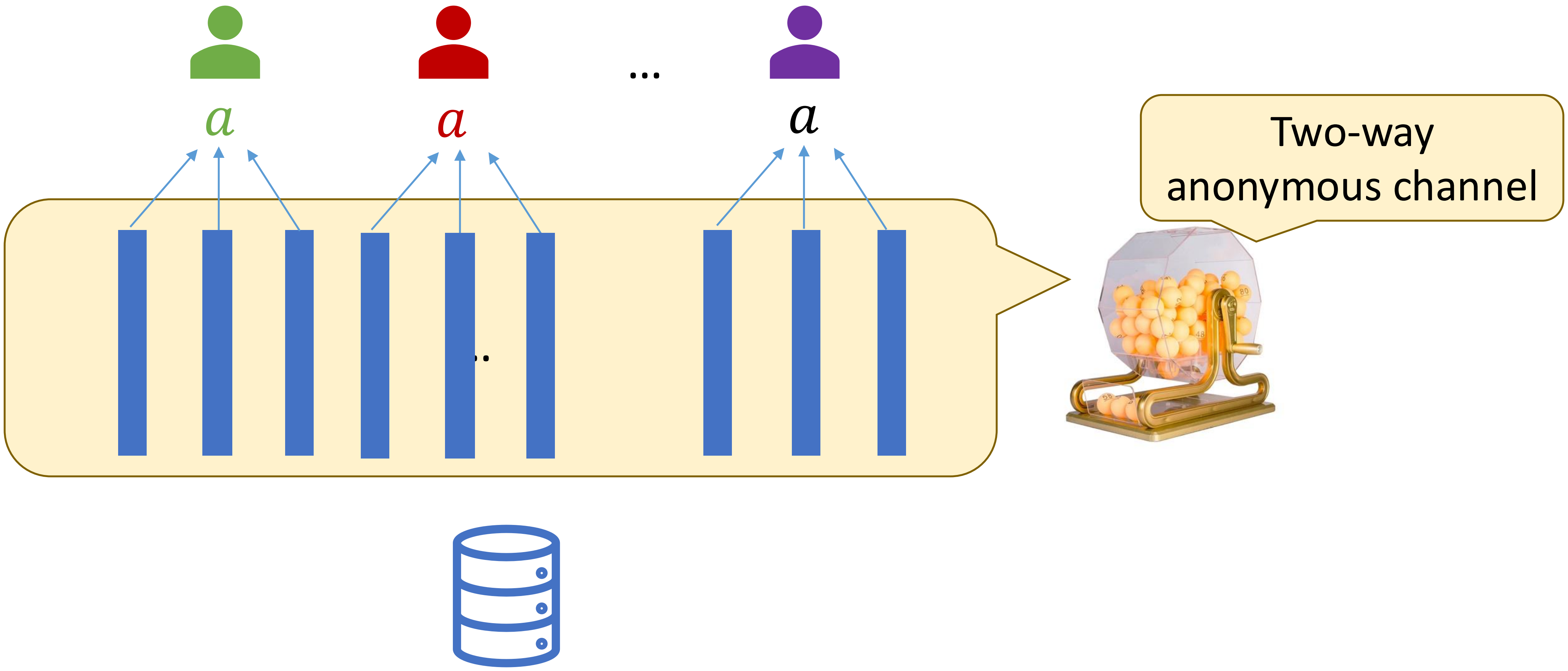
Note: no inner-outer paradigm here, just one PIR scheme



Single-server computationally PIR from split-and-mix

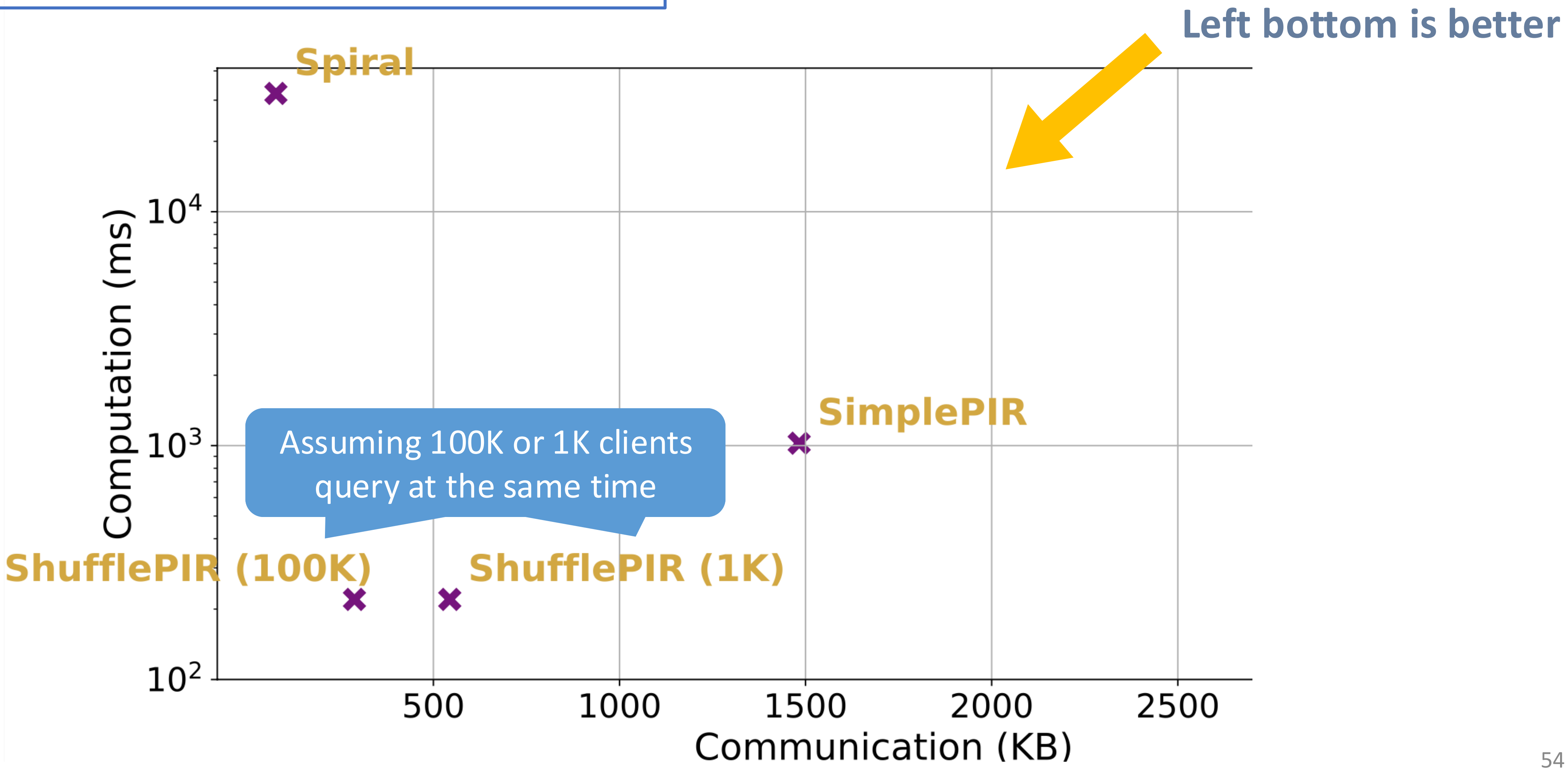


Single-server computationally PIR from split-and-mix



Performance

8GB database, large records (2^{18} entries of 32KB)



Rest of this talk

- Construction backbone: “Split and mix”
- Result 1: A generic construction of IT-PIR in the shuffle model
- Result 2: PIR from computationally secure split-and-mix
- An interesting orthogonal problem: hiding record size without padding
- Discussion and open questions

Discussion

- Two database servers vs. one database server + shuffler

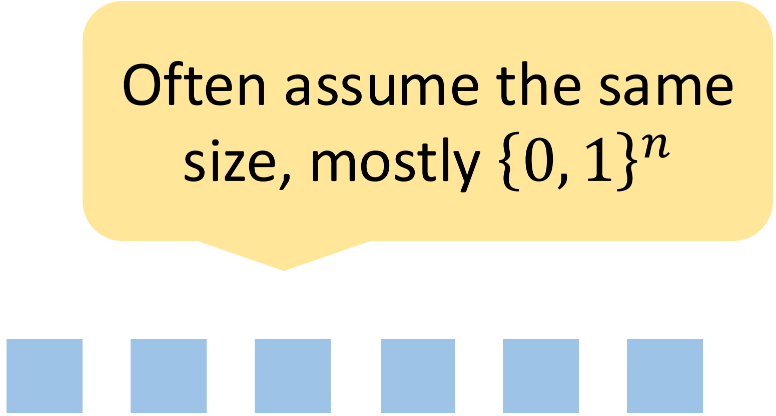
Discussion

- We are in the situation of exploiting tradeoffs: making assumptions, altering models (different types of preprocessing, relaxed security, etc.)
- 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

Backup slides

PIR with variable-sized records

- To deploy PIR in real-world applications…

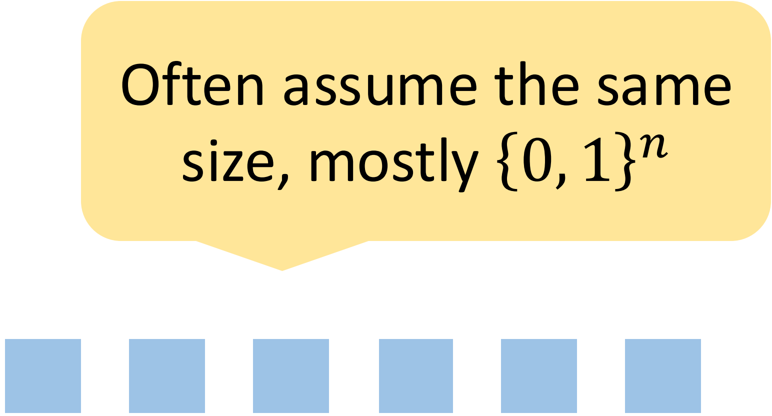


Database entries of PIR in theory

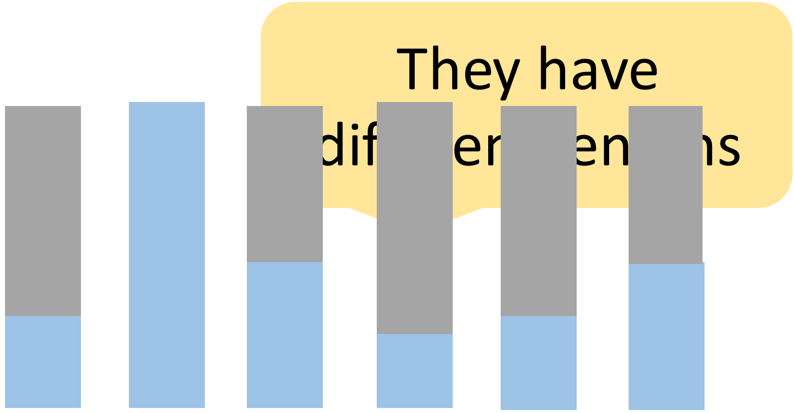
Database records in practice

PIR with variable-sized records

- To deploy PIR in real-world applications…



Database entries of PIR in theory

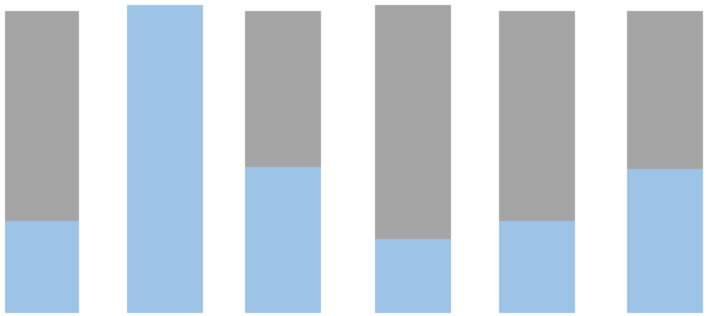


Database records in practice

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

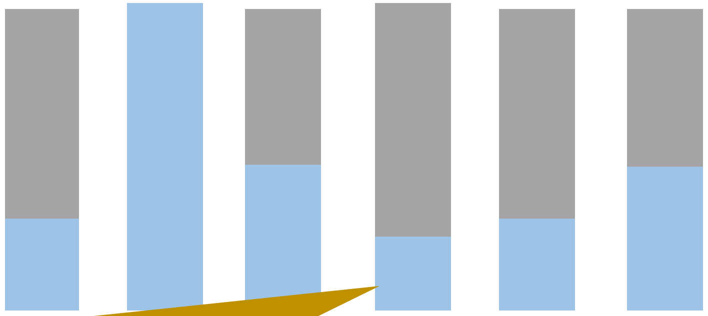
Features

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

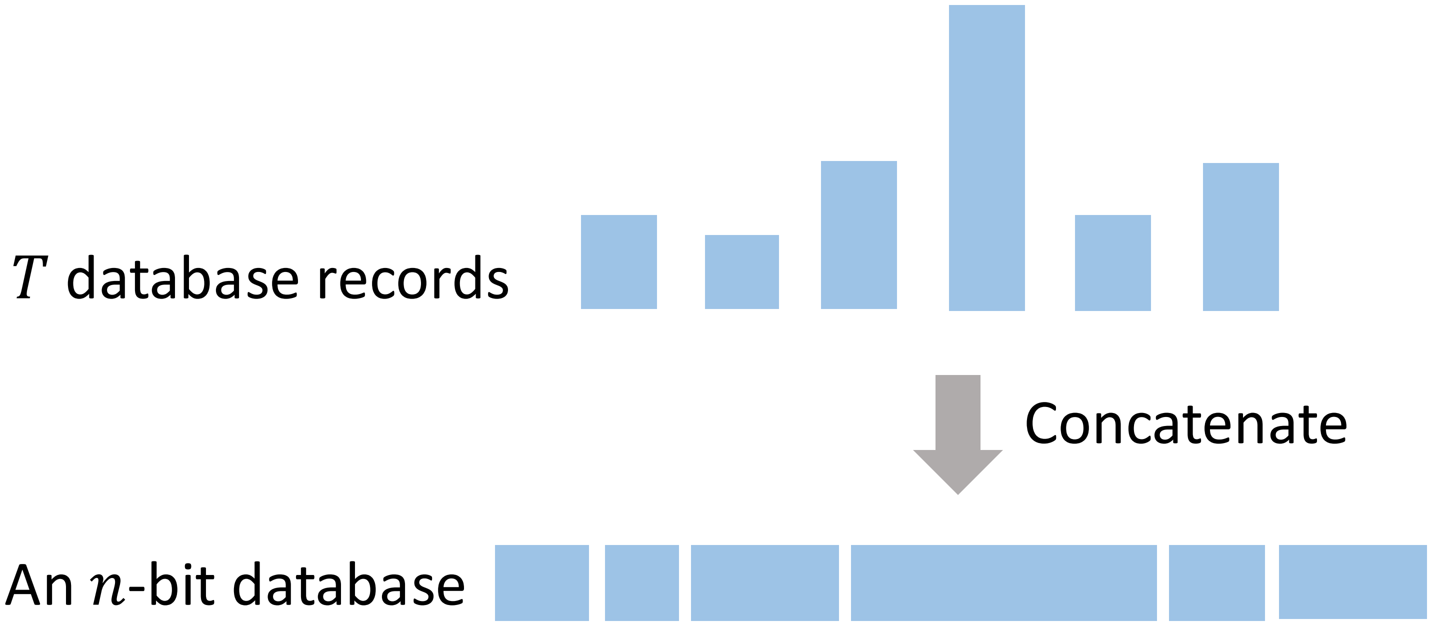
The discrep … an be huge
Majority of the records are small
Most users access the small records much more often than the large records

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

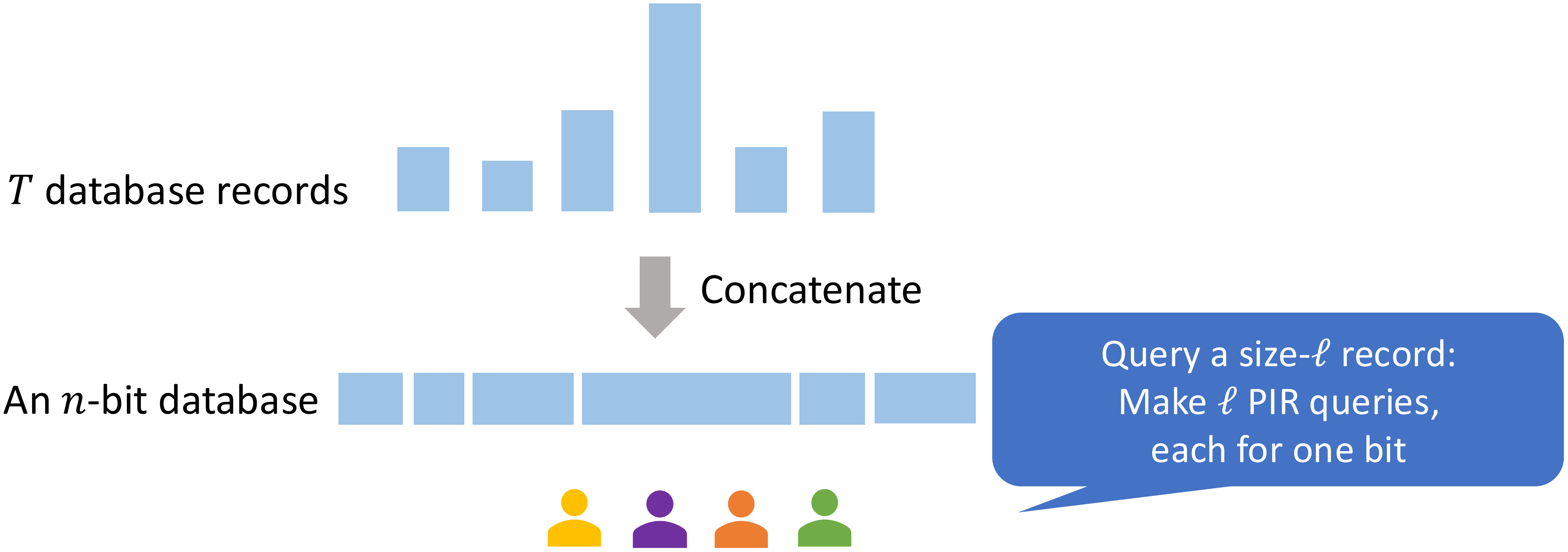
PIR with variable-sized records

- A toy protocol



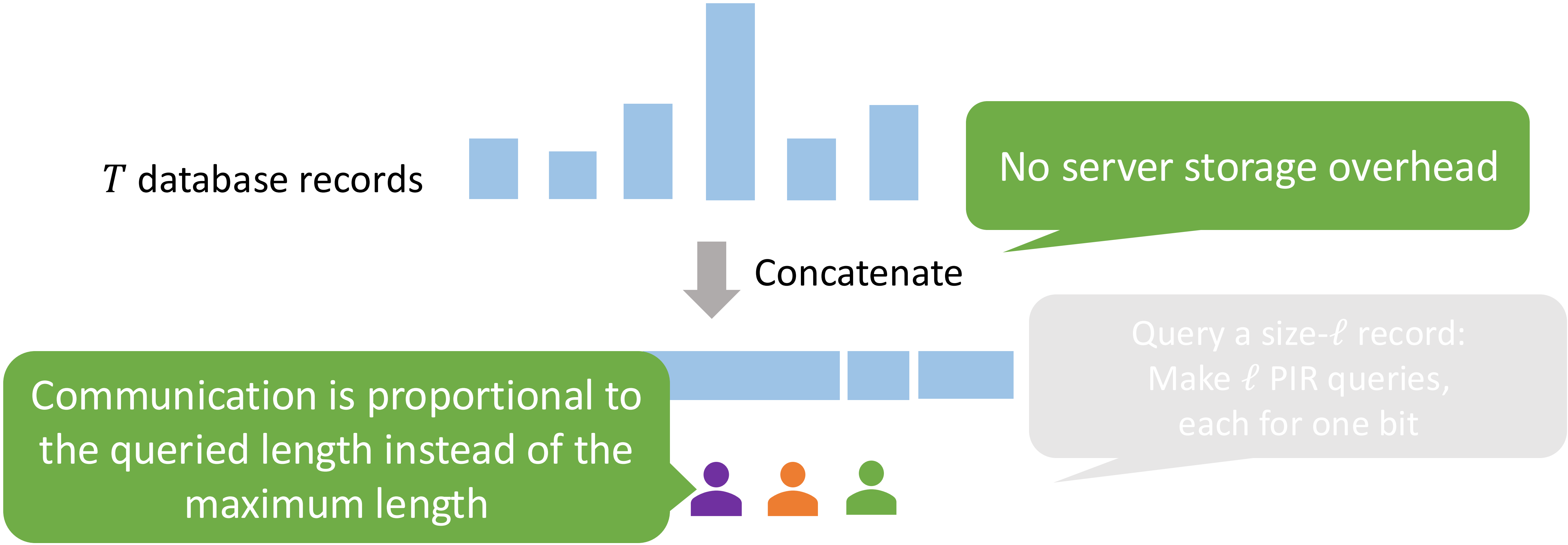
PIR with variable-sized records

- 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?

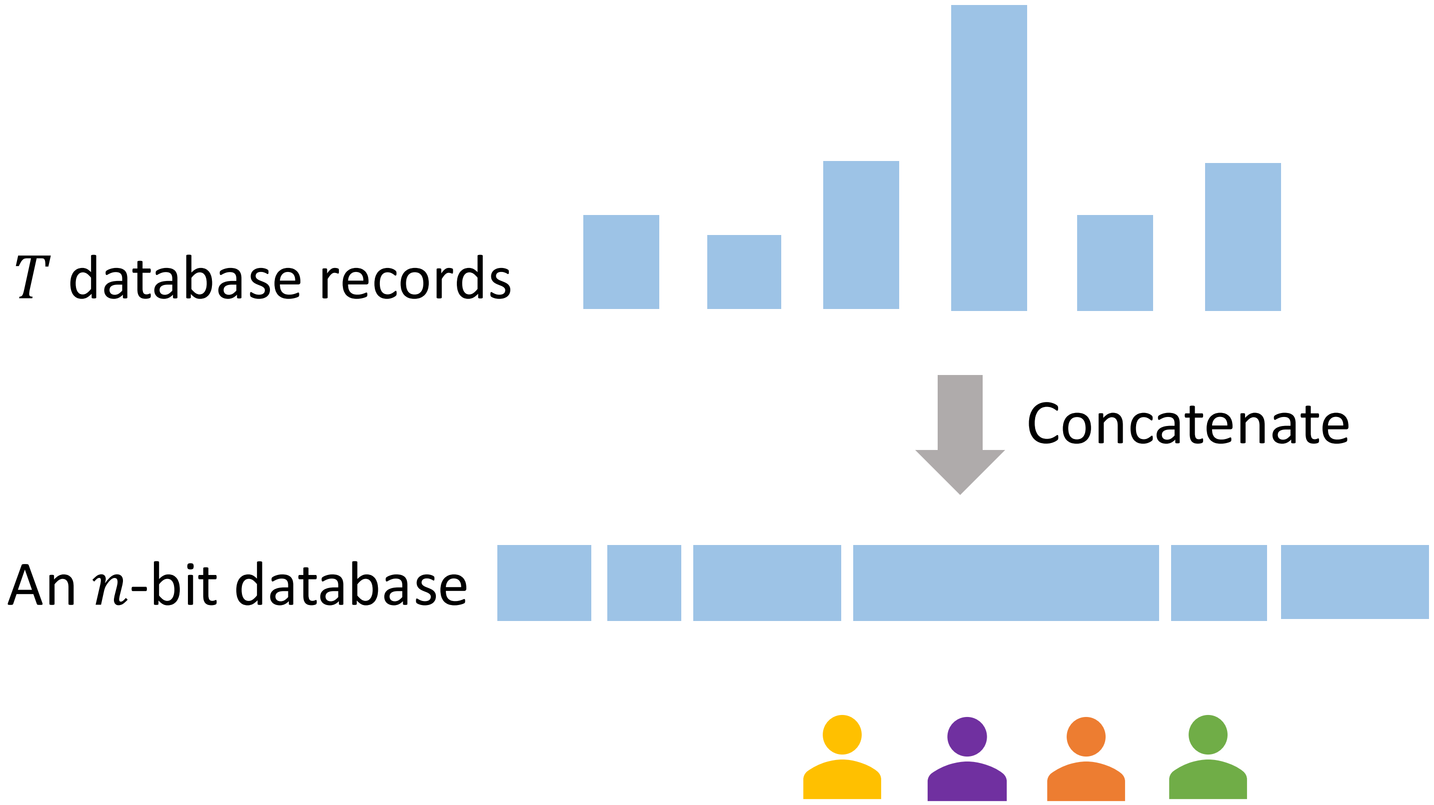
Yes, from ℓ PIR queries to $\text{polylog}\ell$ PIR queries

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

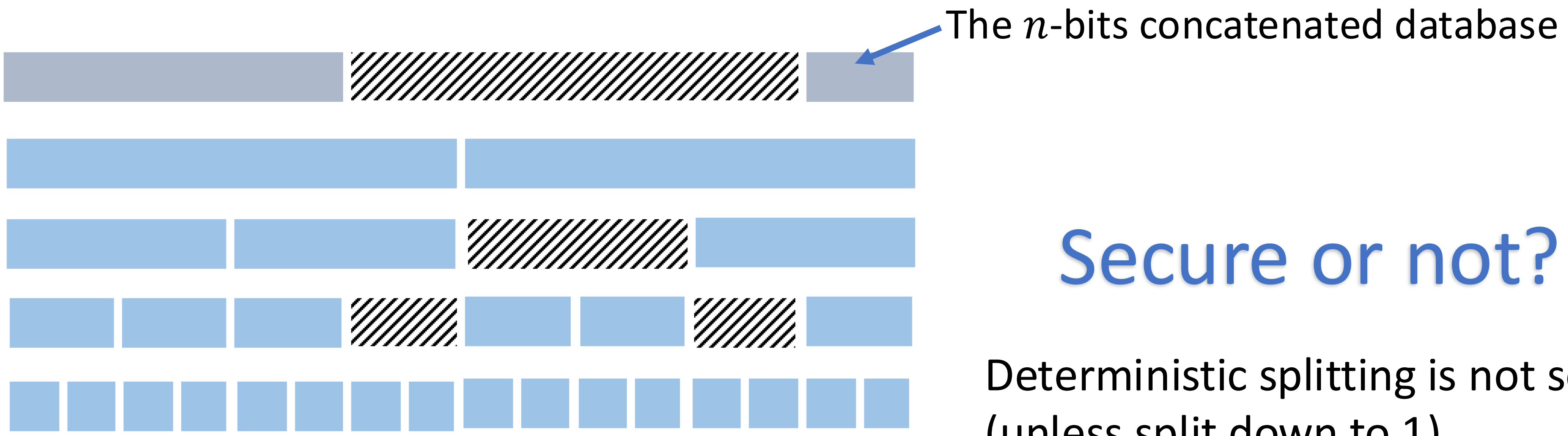


Why not retrieve more bits in each PIR query?

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



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

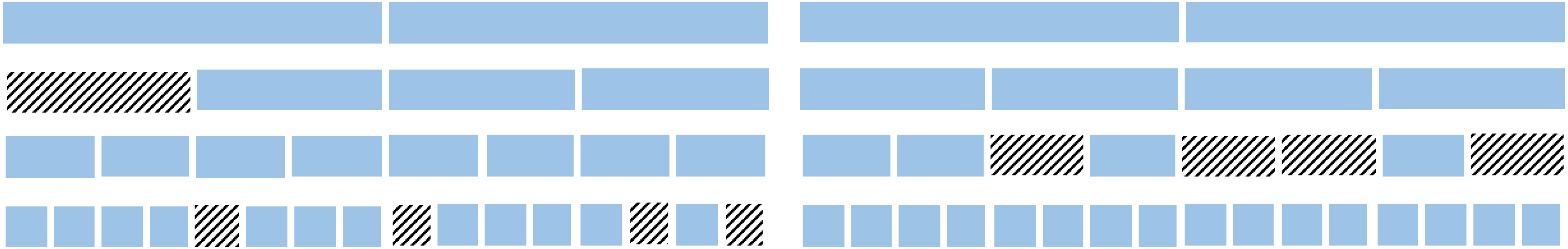
Secure or not?

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

PIR with variable-sized records

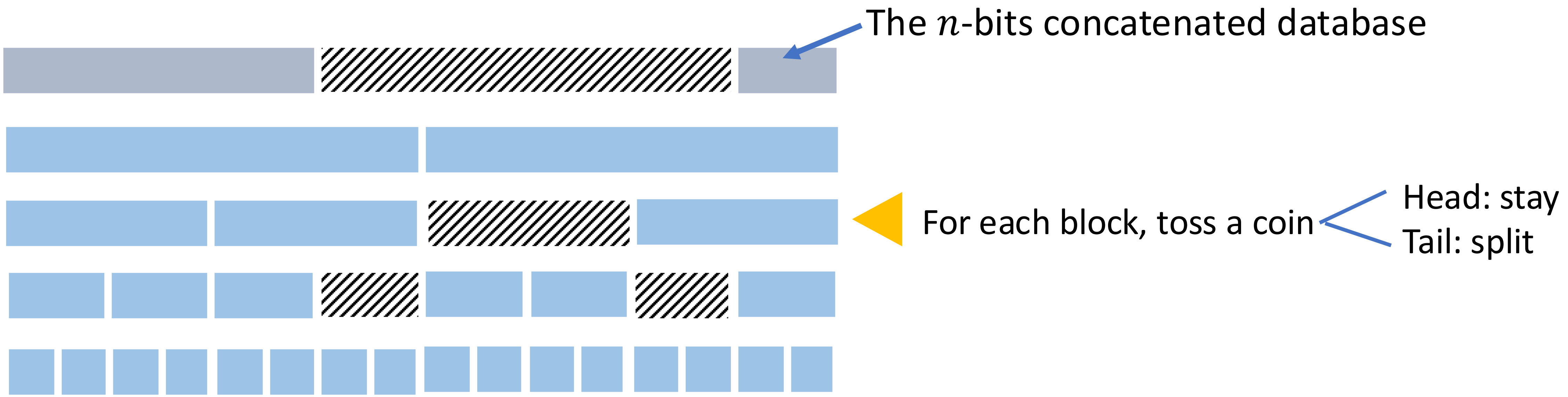
- Splitting records to the powers of two

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



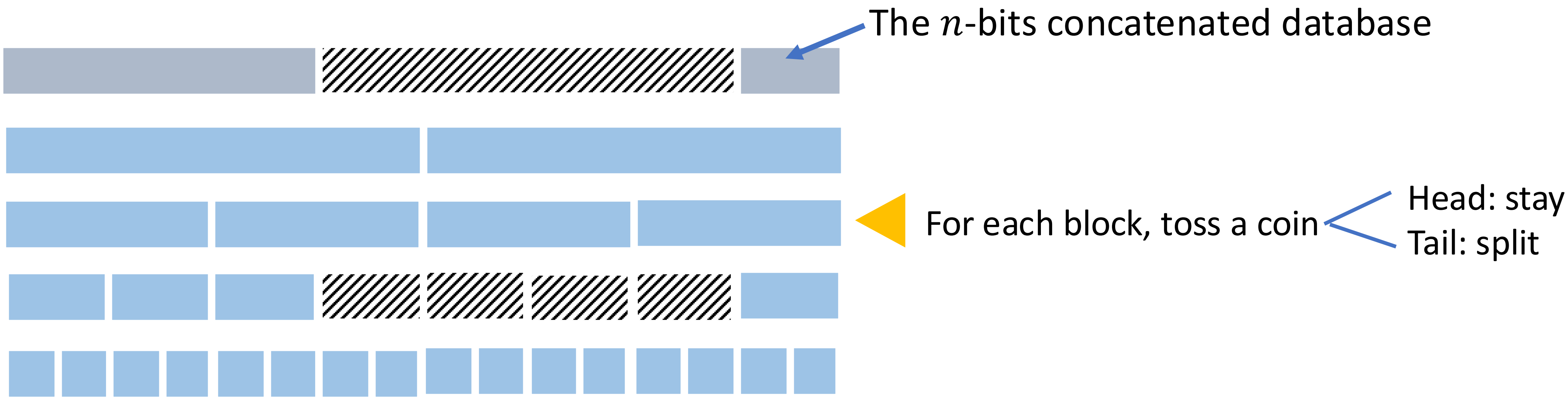
PIR with variable-sized records

- Our approach: recursive splitting



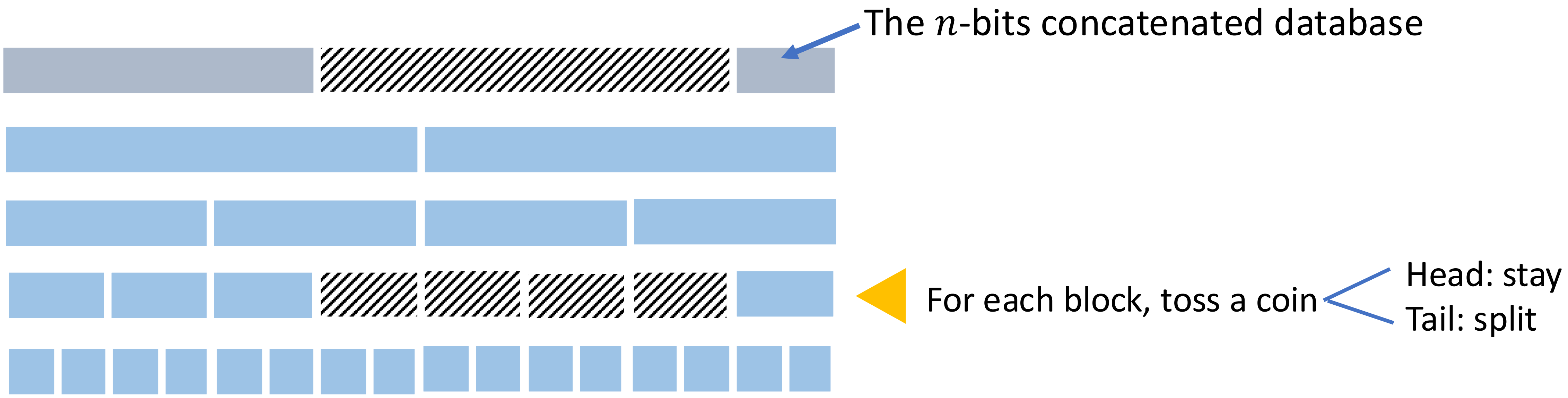
PIR with variable-sized records

- Our approach: recursive splitting



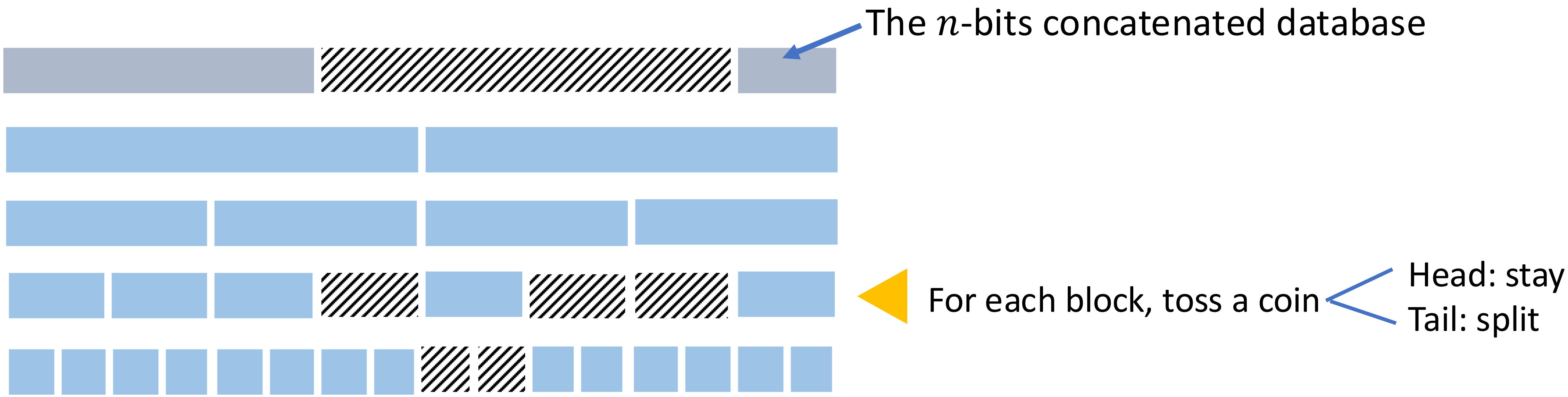
PIR with variable-sized records

- Our approach: recursive splitting



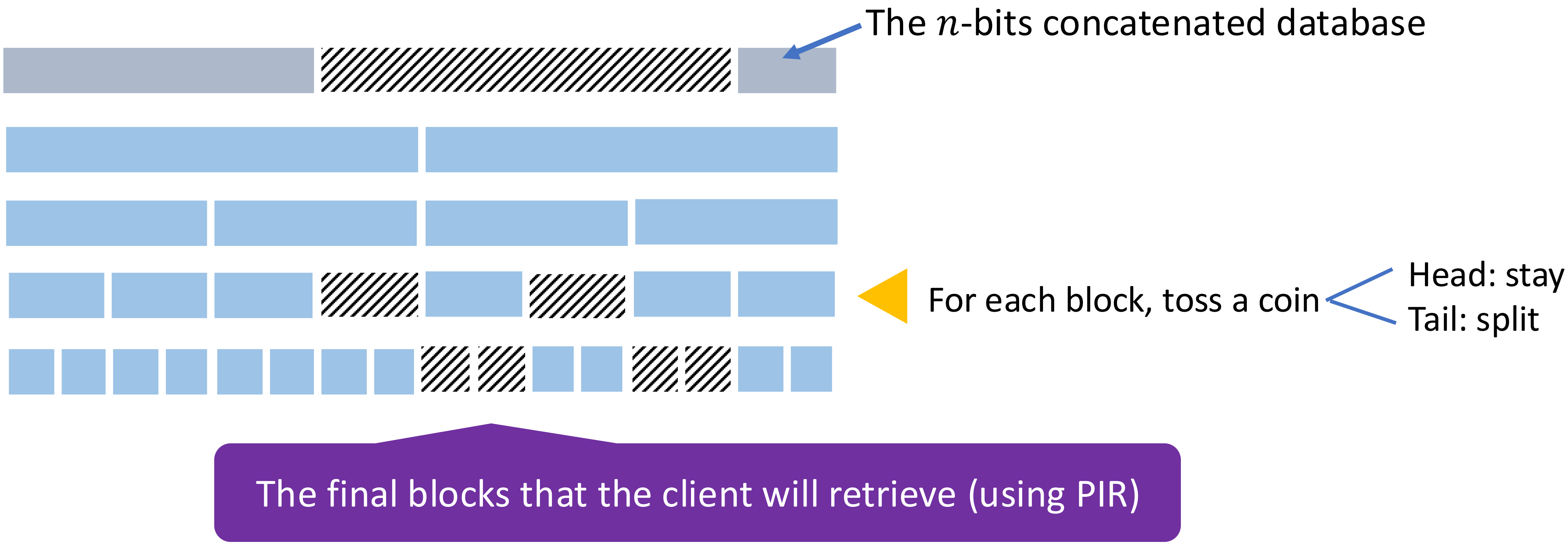
PIR with variable-sized records

- Our approach: recursive splitting



PIR with variable-sized records

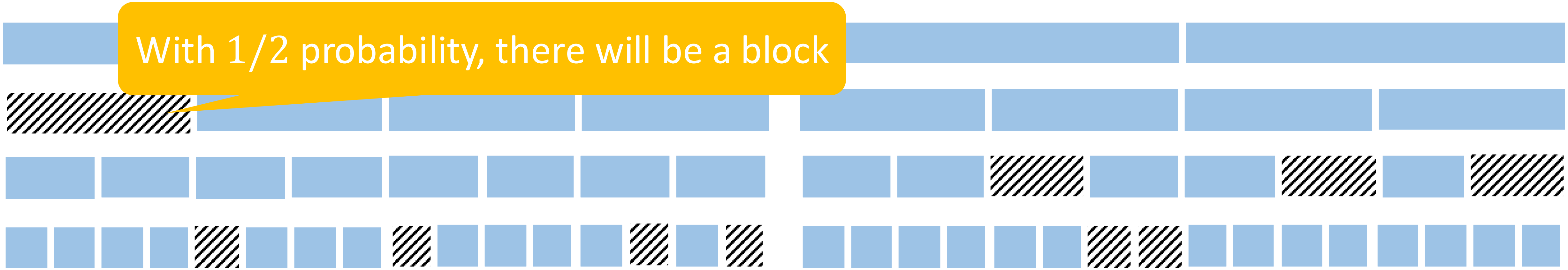
- Our approach: recursive splitting



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

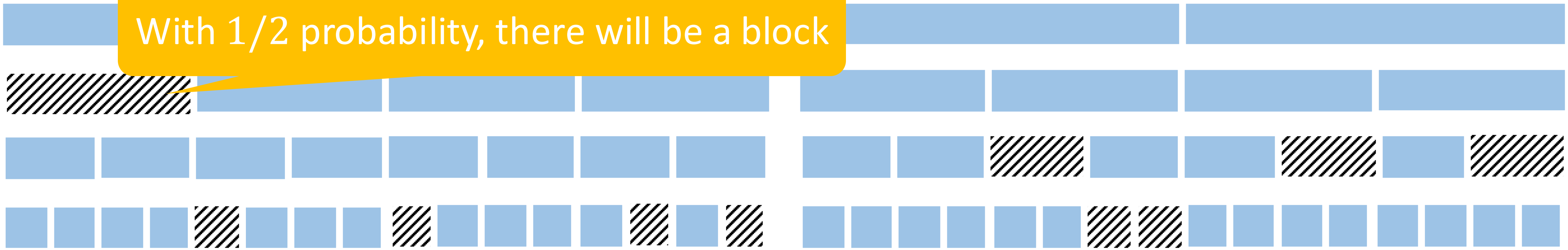


PIR with variable-sized records

- 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

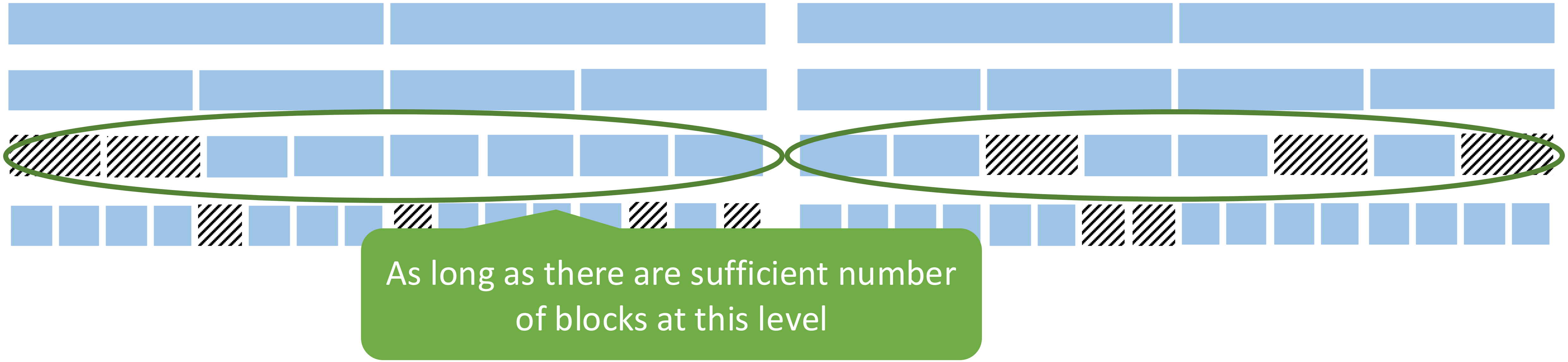
With 1/2 probability, there will be a block



PIR with variable-sized records

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

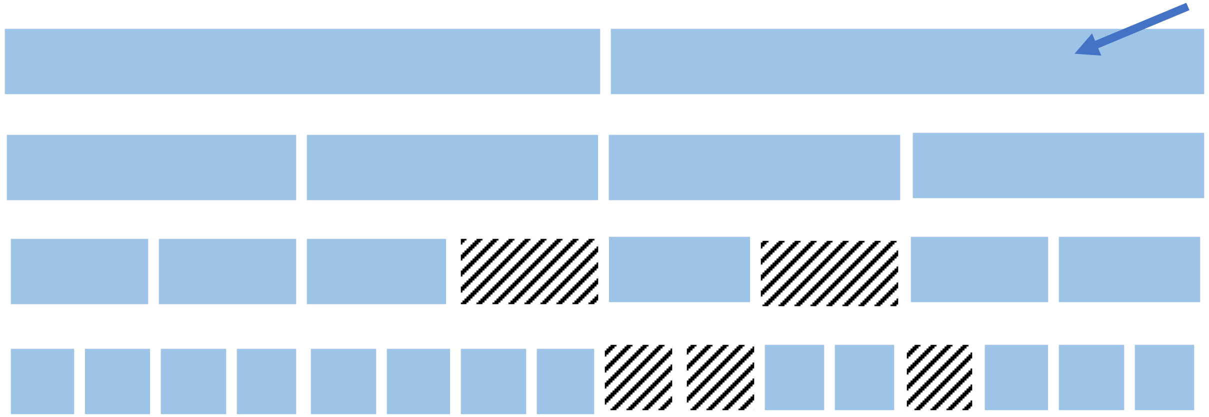
Consider M-3 1 1 1 v.s. M/4 M/4 M/4 M/4

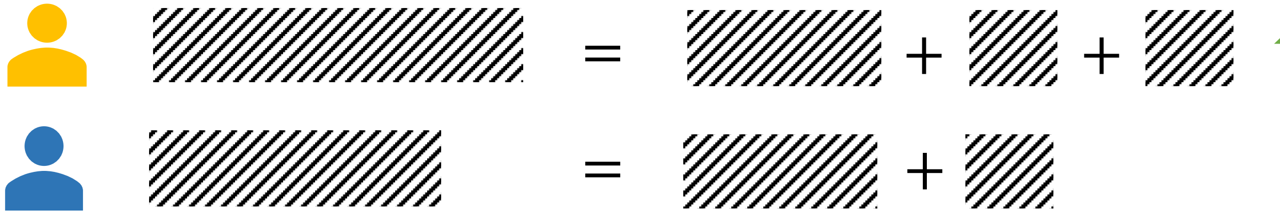


PIR with variable-sized records

- 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