ACM CCS 2024

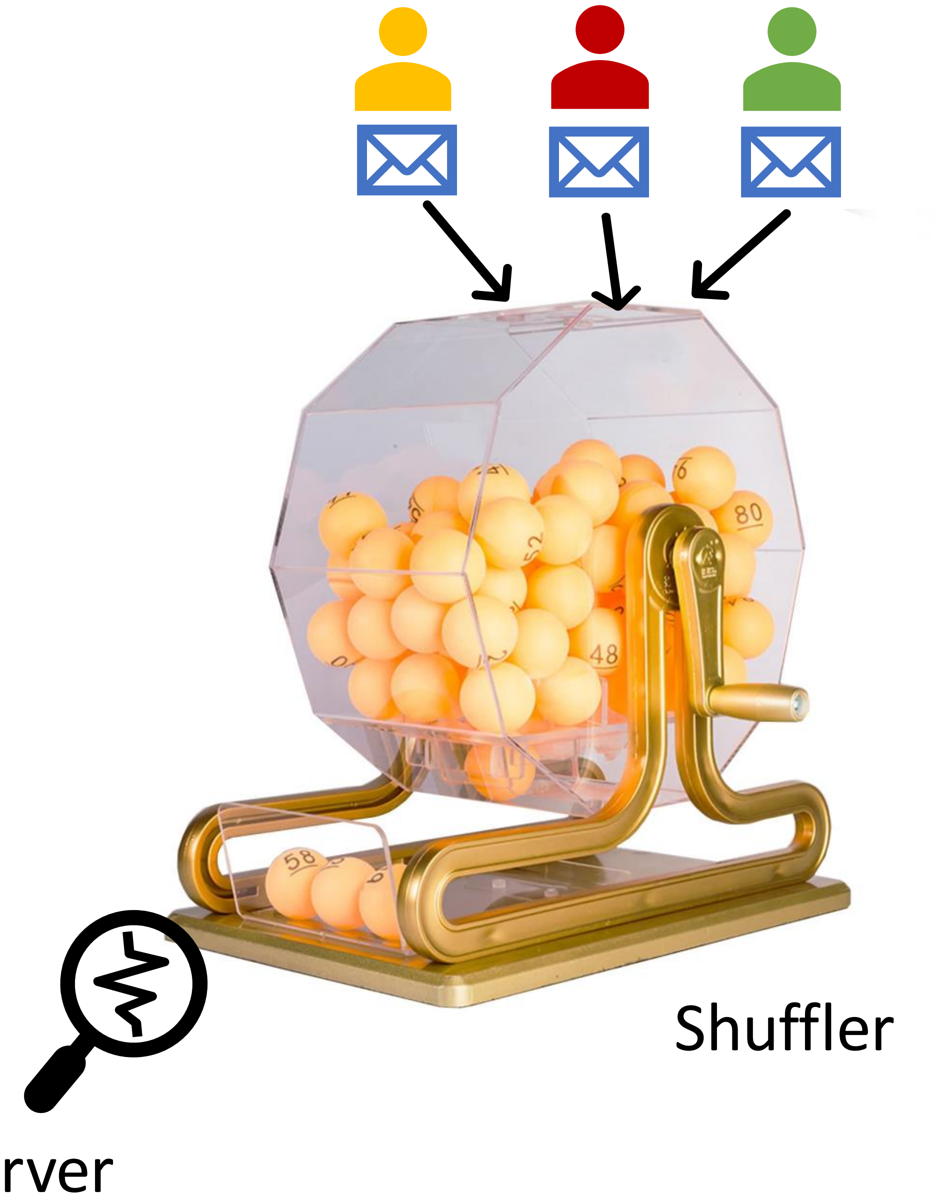
Computationally Secure Aggregation and PIR in the Shuffle Model

Adrià Gascón, Yuval Ishai, Mahimna Kelkar
Baiyu Li, Yiping Ma, Mariana Raykova

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

The shuffle model

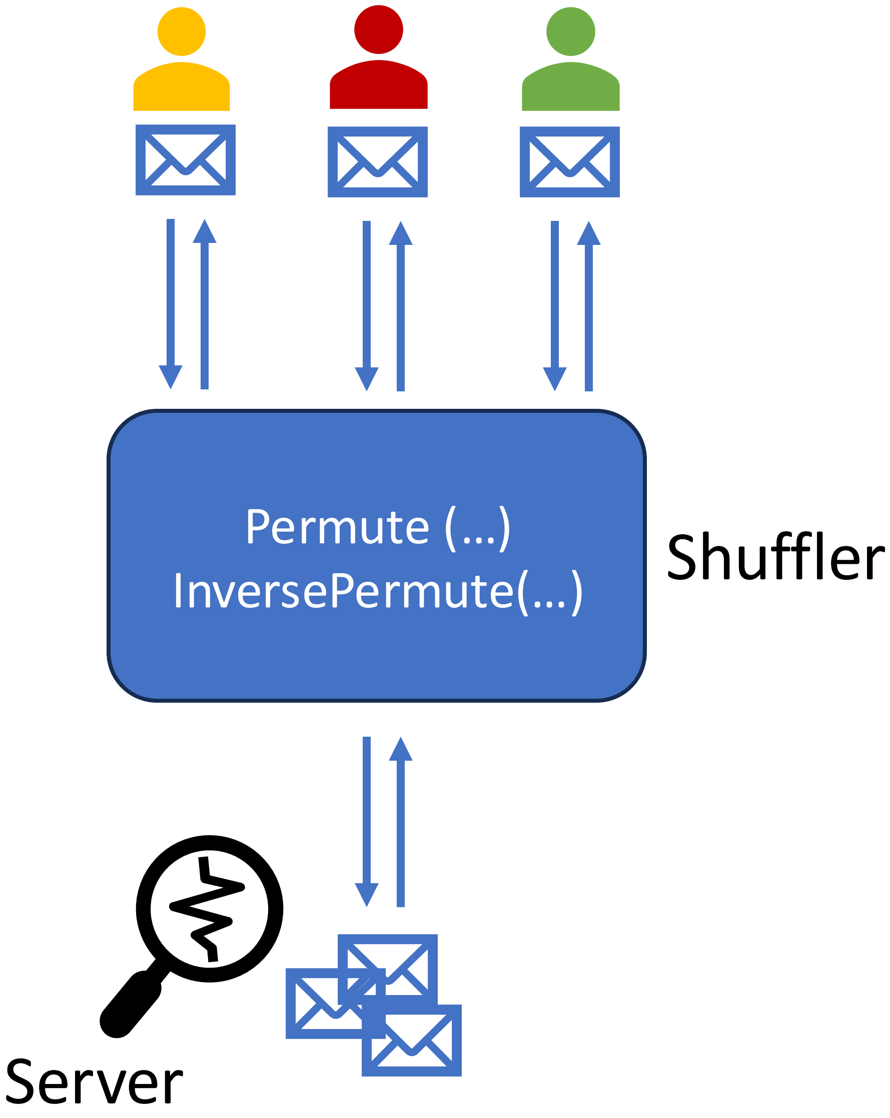
- Purpose: anonymization
- A popular model in differential privacy community
  [Bittau et al. 2017]
  [Cheu et al. 2019]
  [Erlingsson et al. 2019]
  …
- Can be instantiated by, e.g., Tor



The shuffle model

- Purpose: anonymization
- A popular model in differential privacy community
- Can be instantiated by, e.g., Tor
- Later in our PIR setting:
  - We assume it is two-way
  - Can be viewed as a second shuffle server who does not hold the database

A hybrid model between single-server and two-server



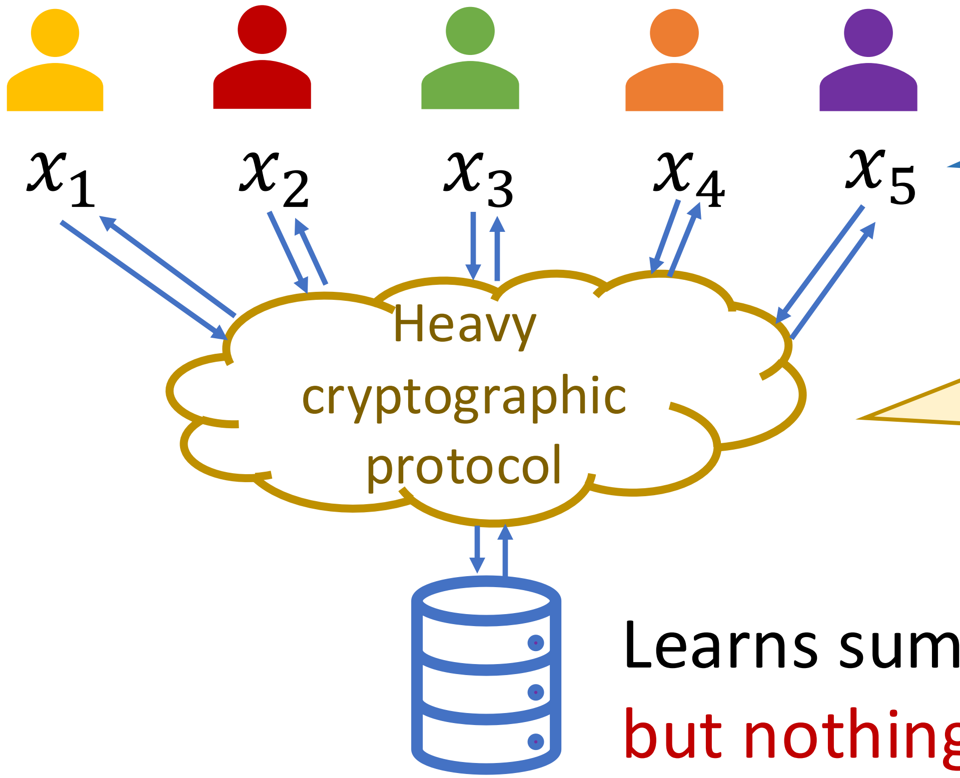
The main theme

Secure aggregation

Private information retrieval (PIR)

Improving efficiency of secure computation tasks
utilizing the shuffler

Single-server secure aggregation



Can be a number, or a vector

- Expensive cryptographic computation
- Interaction between participants

Learns sum of x_i's

but nothing about individual x_i

Introducing anonymity into this problem

What we can get:

- Lightweight local computation
- Non-interactive

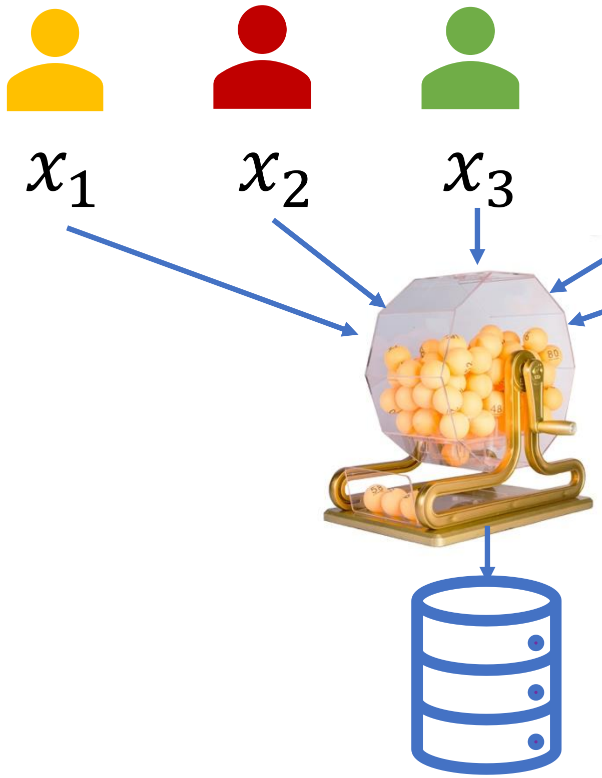
x_1 x_2 x_3 x_4 x_5

A global shuffler

Learns sum of x_i's

but nothing about individual x_i

Anonymity does not trivialize the problem

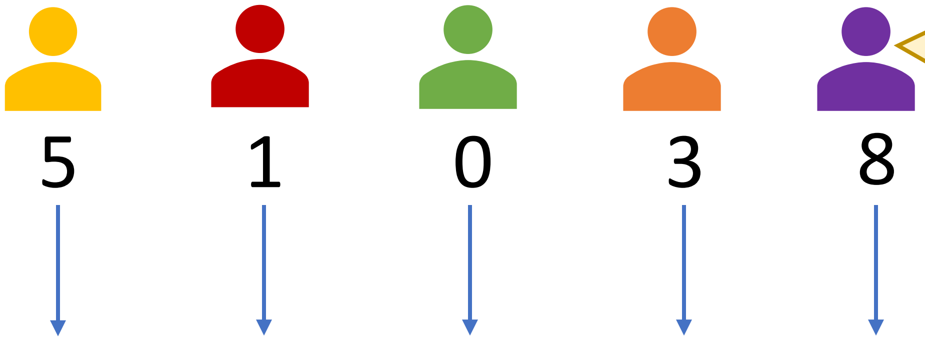


The shuffler hides who sends which message, but does not hide the message itself

Gets permuted x_i's, adds them up

The split-and-mix paradigm [IKOS06]

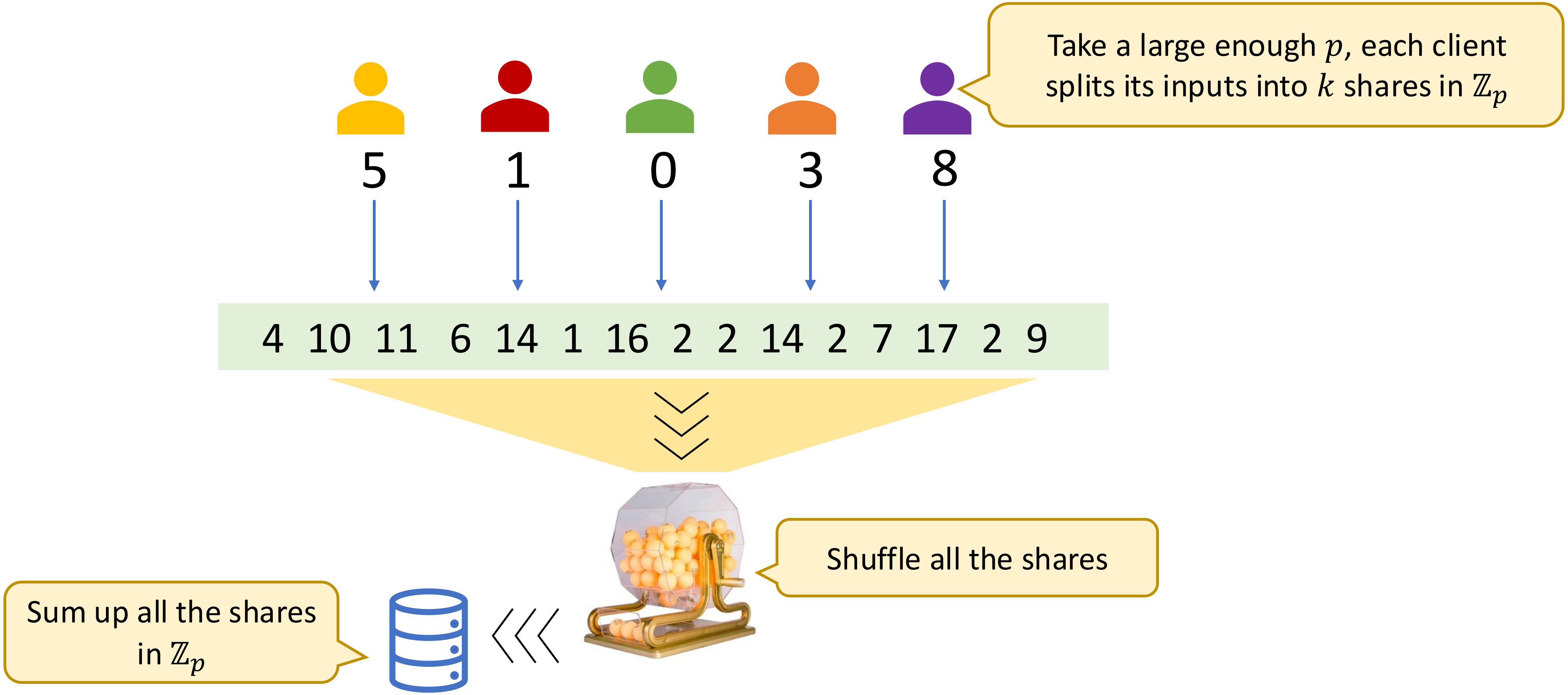
Take a large enough p, each client splits its inputs into k shares in $\mathbb{Z}_p$



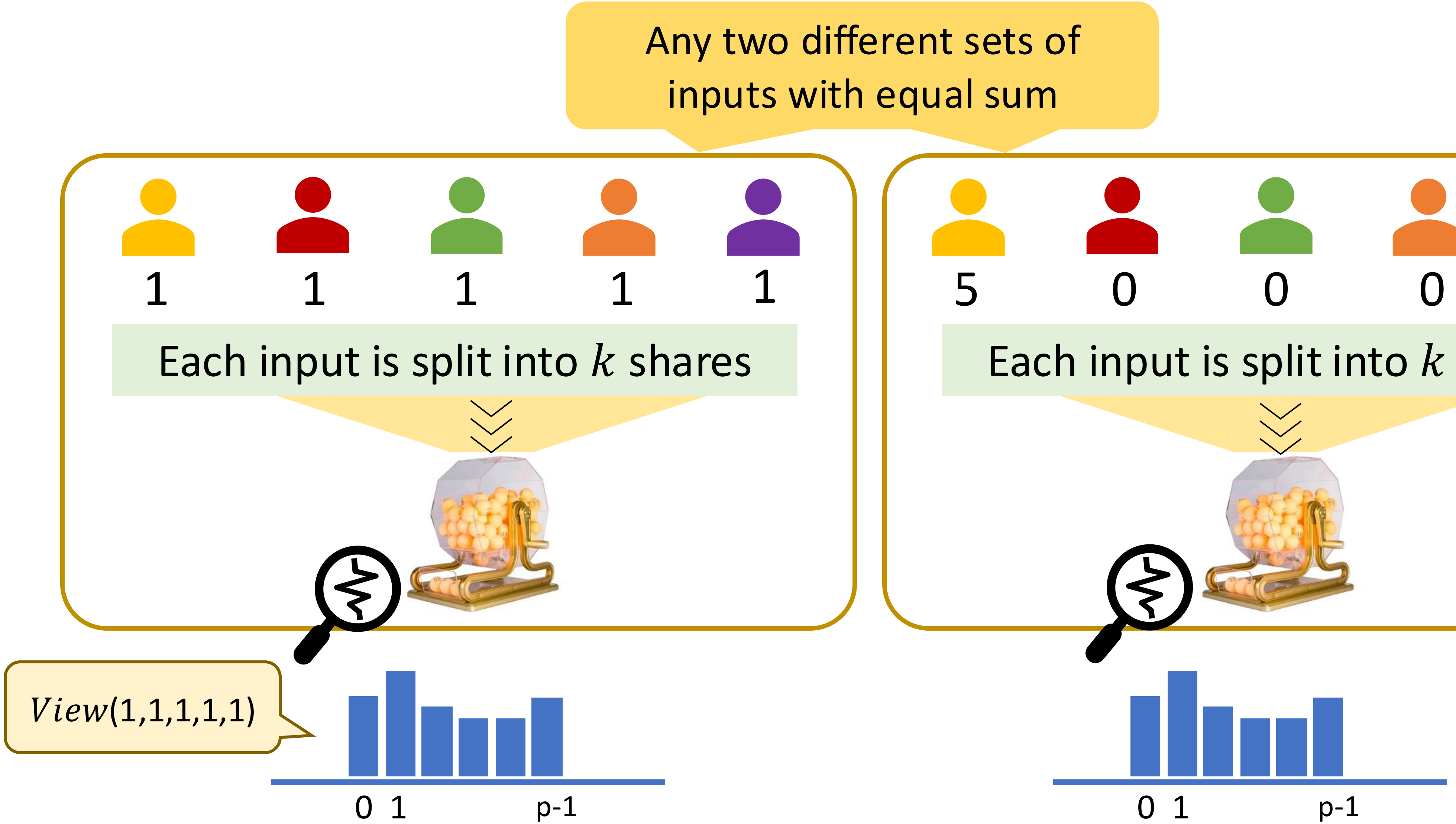
4+10+11 6+14+1 16+2+2 14+2+7 17+2+9

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

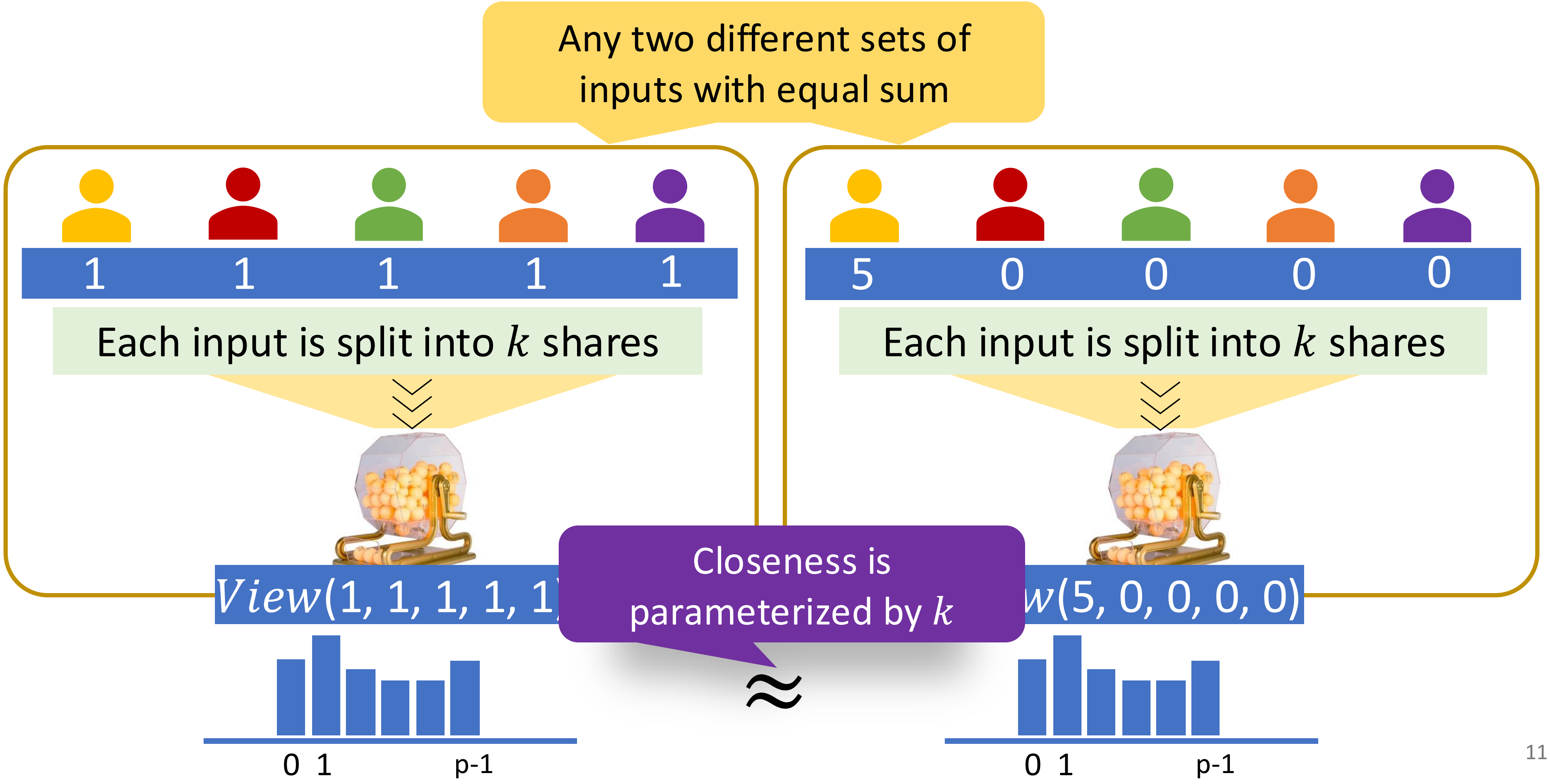
The split-and-mix paradigm [IKOS06]



Security of split-and-mix



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

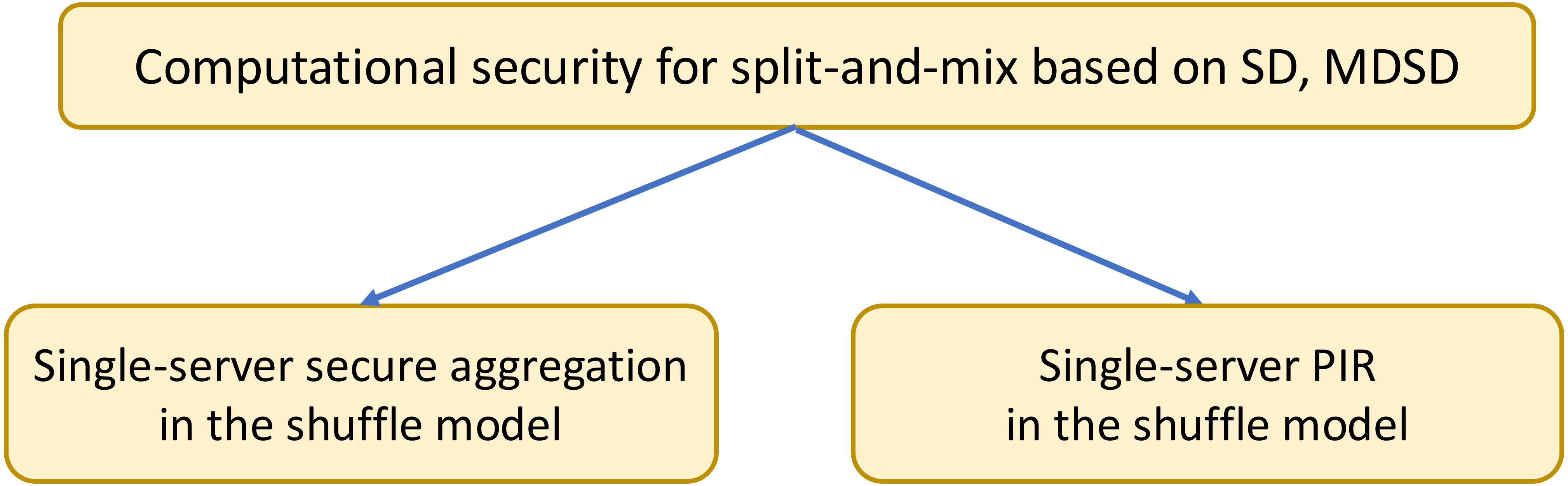
$$\mathit{View}(1, 1, 1, 1, 1) \approx \mathit{View}(5, 0, 0, 0, 0)$$

- 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



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

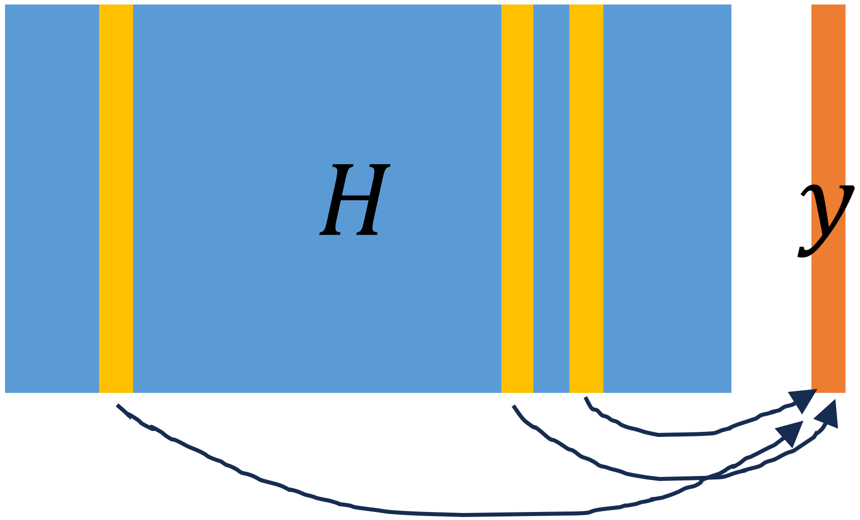
(Even giving advantage to the baseline by compressing the shares)

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 $H \cdot e = y$

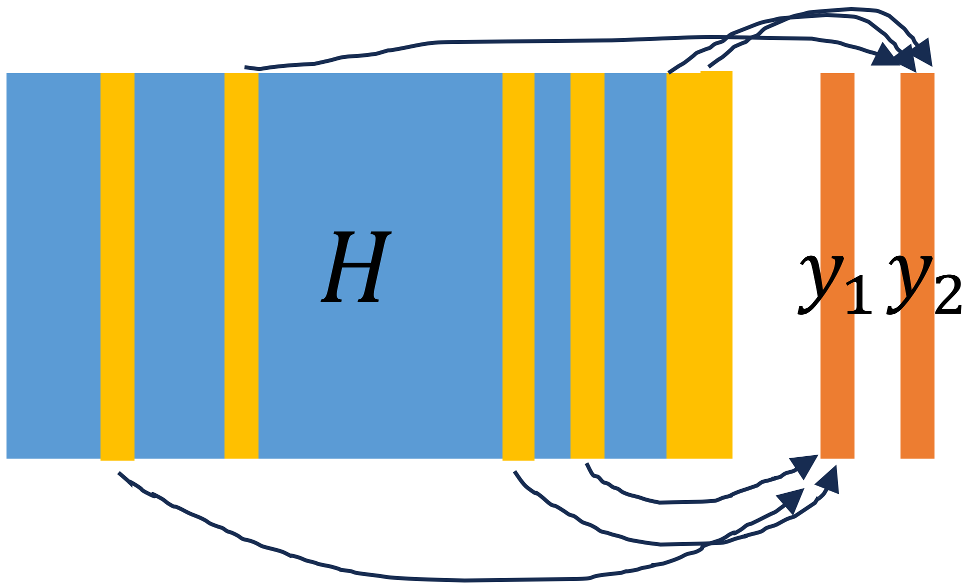


Split-and-mix based on Syndrome Decoding (SD)

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

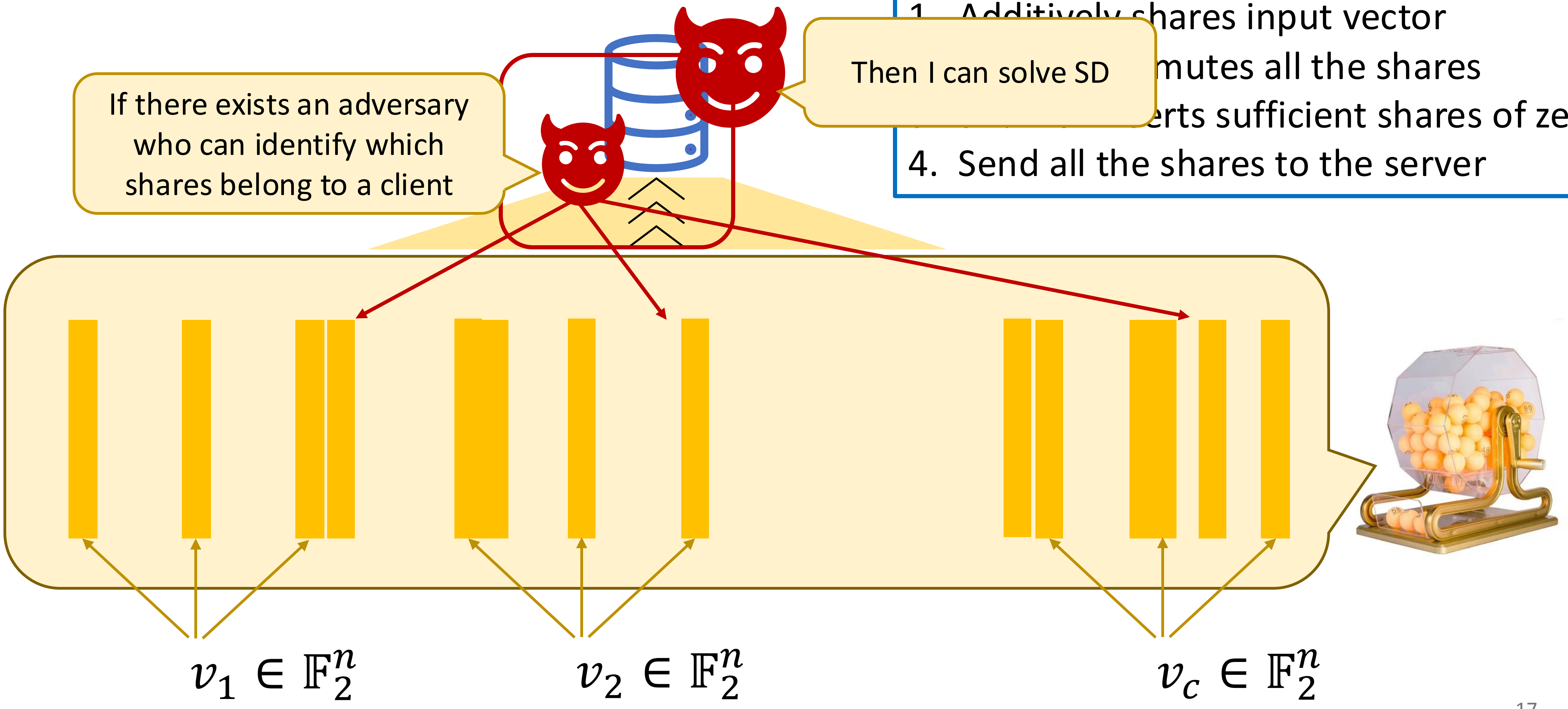
Computationally hard to find "low-weight" E such that $H \cdot E = Y$

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



The resulting aggregation protocol in a nutshell

1. Additively shares input vector
... mutes all the shares
... erts sufficient shares of zero
4. Send all the shares to the server



Our results

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

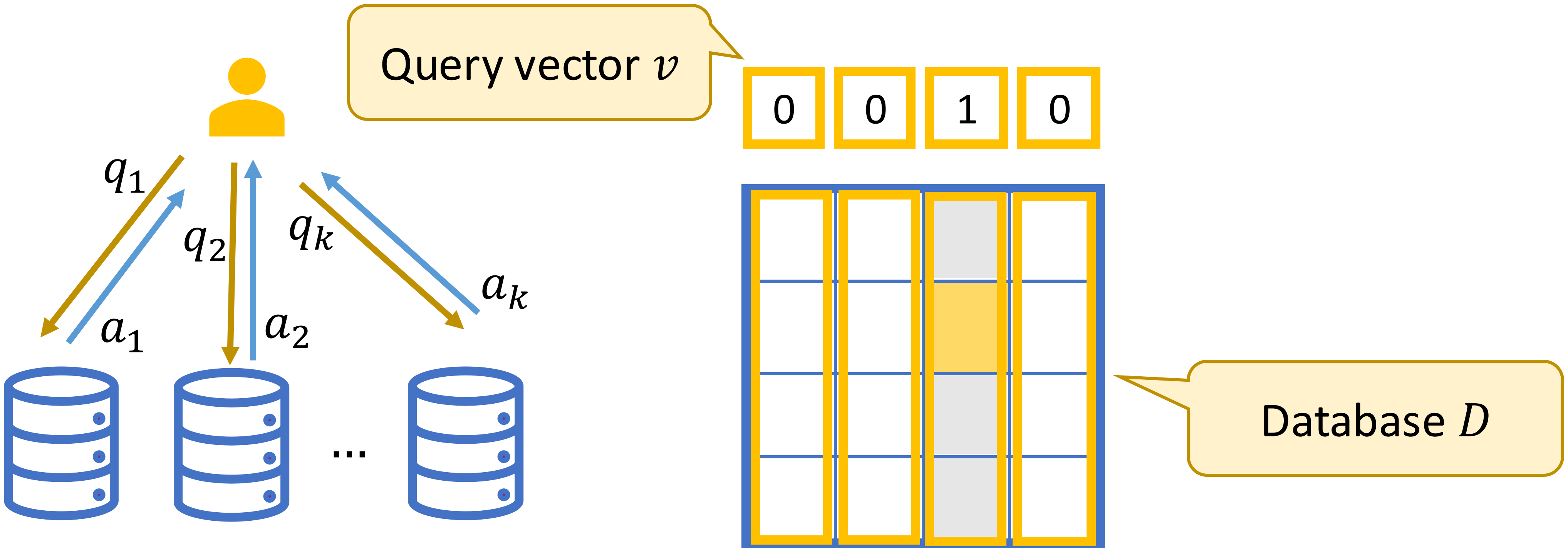
Single-server secure aggregation in the shuffle model

Up to 25X savings for communication compared to the best baseline in the statistical setting

Single-server PIR in the shuffle model

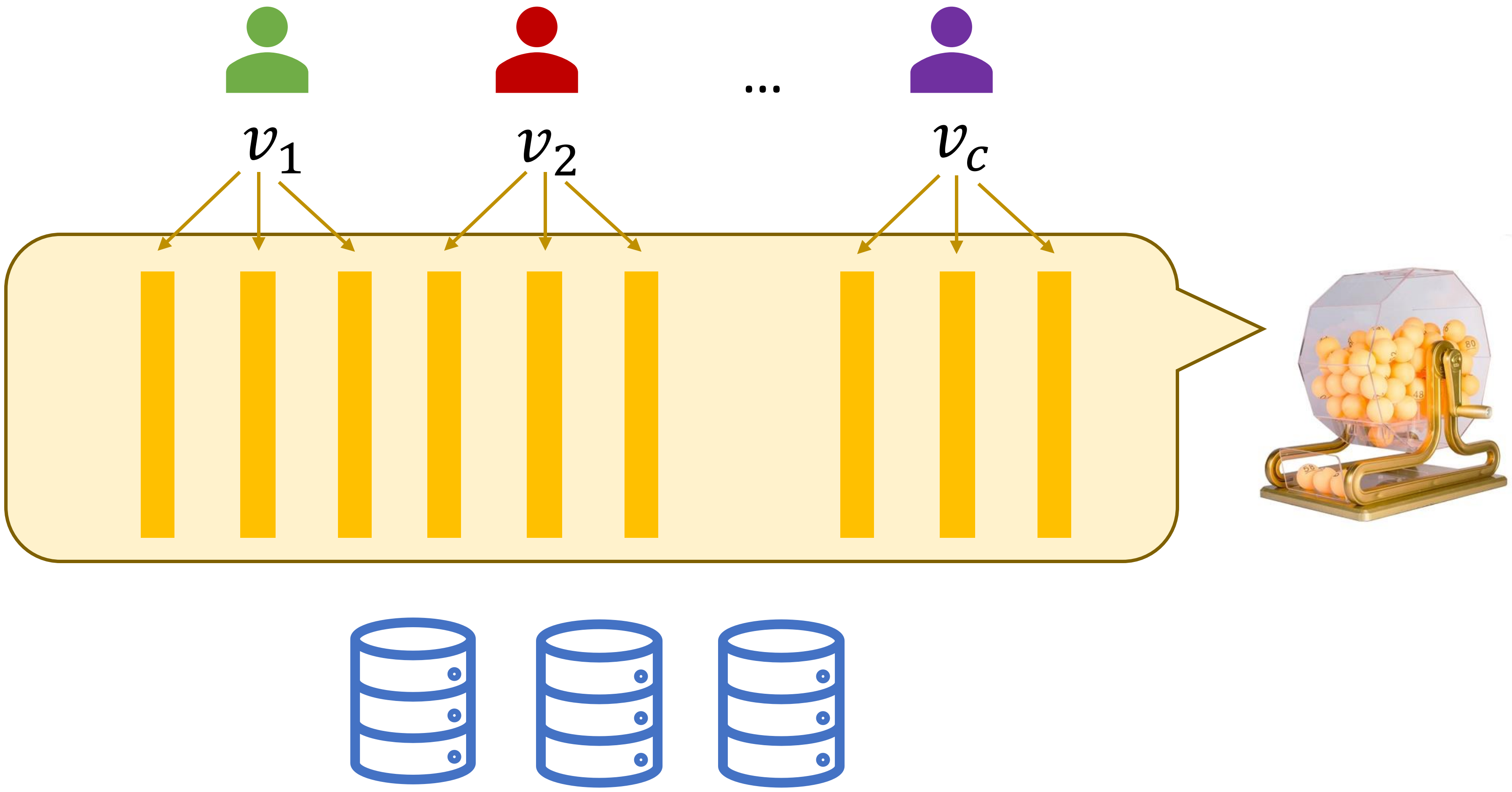
Up to 22X improvement of throughput (in the batch setting) over SimplePIR [HHCMV23] and comparable communication cost

Starting point: a classic multi-server PIR

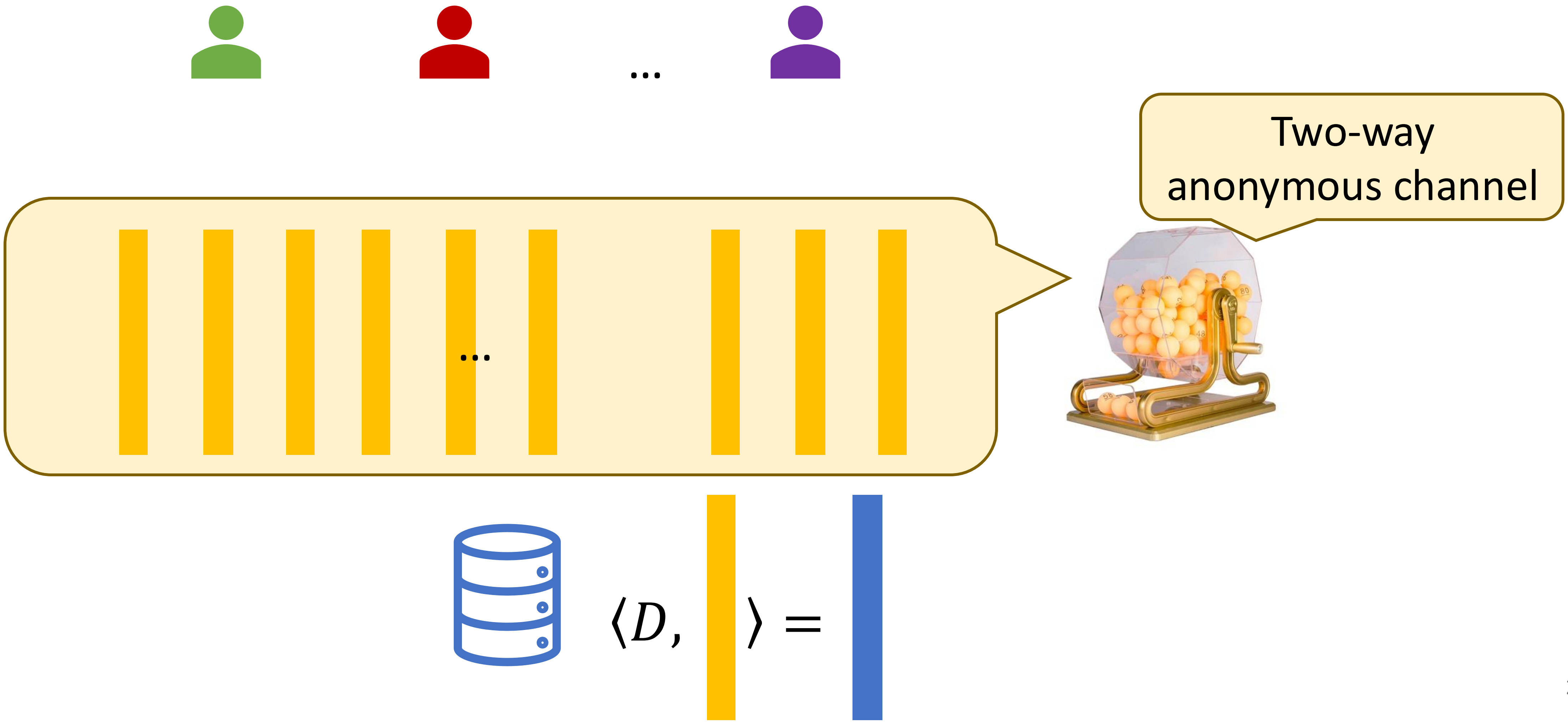


$$
\begin{aligned}
v \cdot D &= (q_1 + \cdots + q_k) \cdot D \\
&= q_1 \cdot D + \cdots + q_k \cdot D \\
&\coloneqq a_1 + \cdots + a_k
\end{aligned}
$$

Single-server PIR from split-and-mix

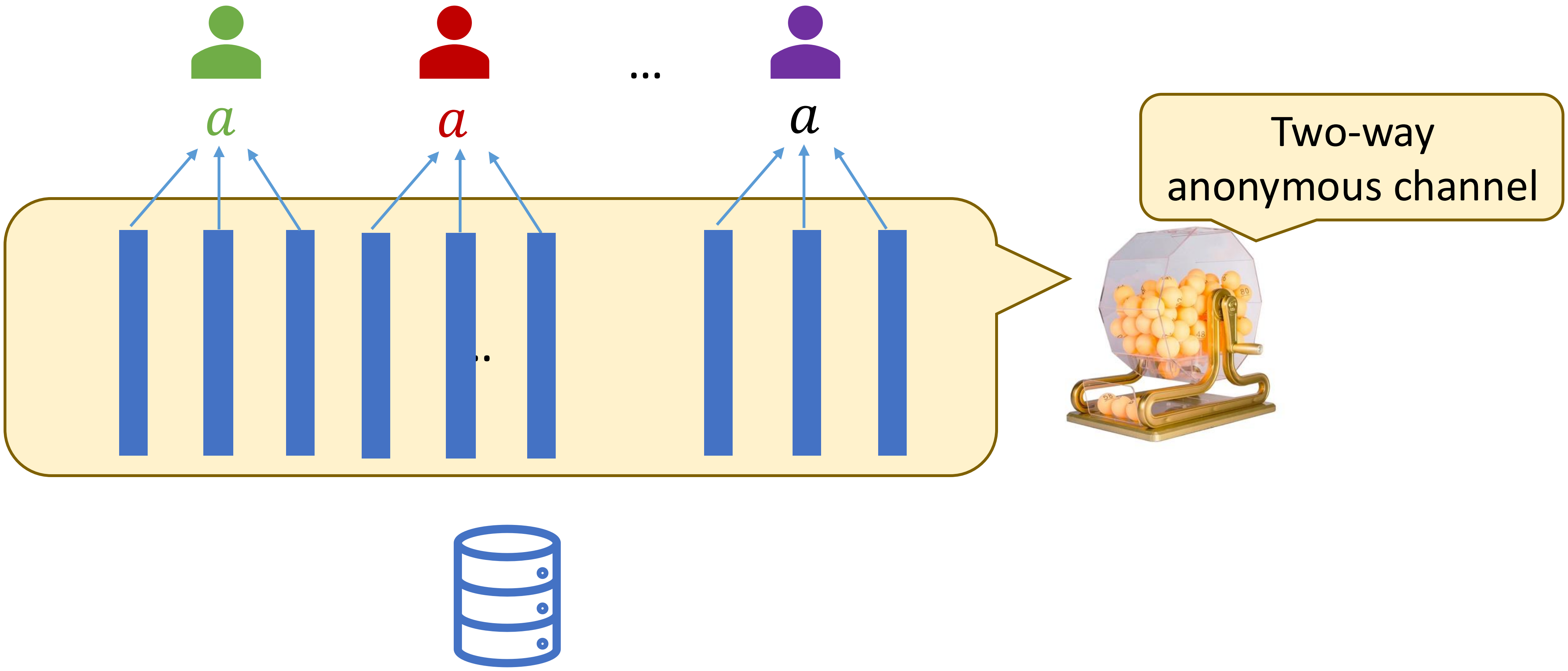


Single-server PIR from split-and-mix



Single-server PIR from split-and-mix

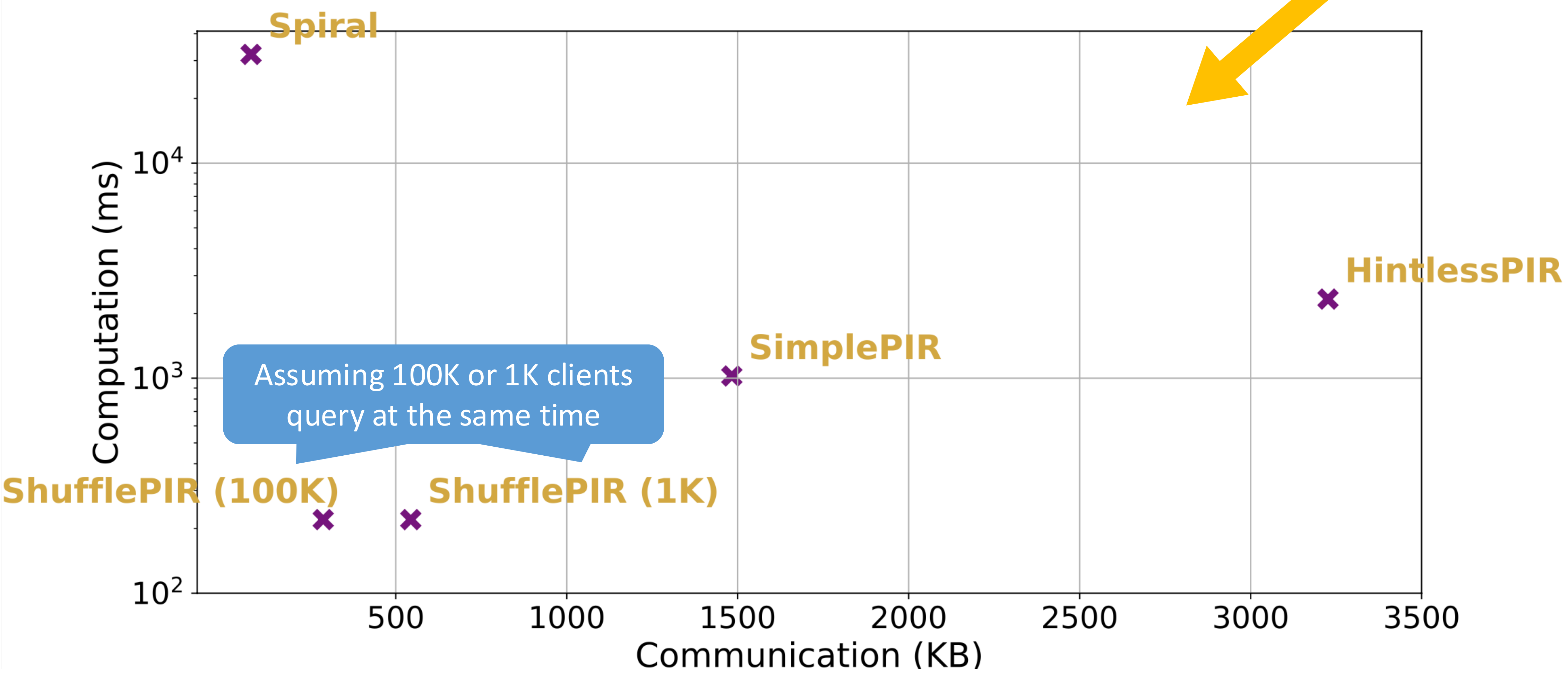
[IKOS06] initialized the study of PIR from split-and-mix, but their construction is rather theoretical



Performance

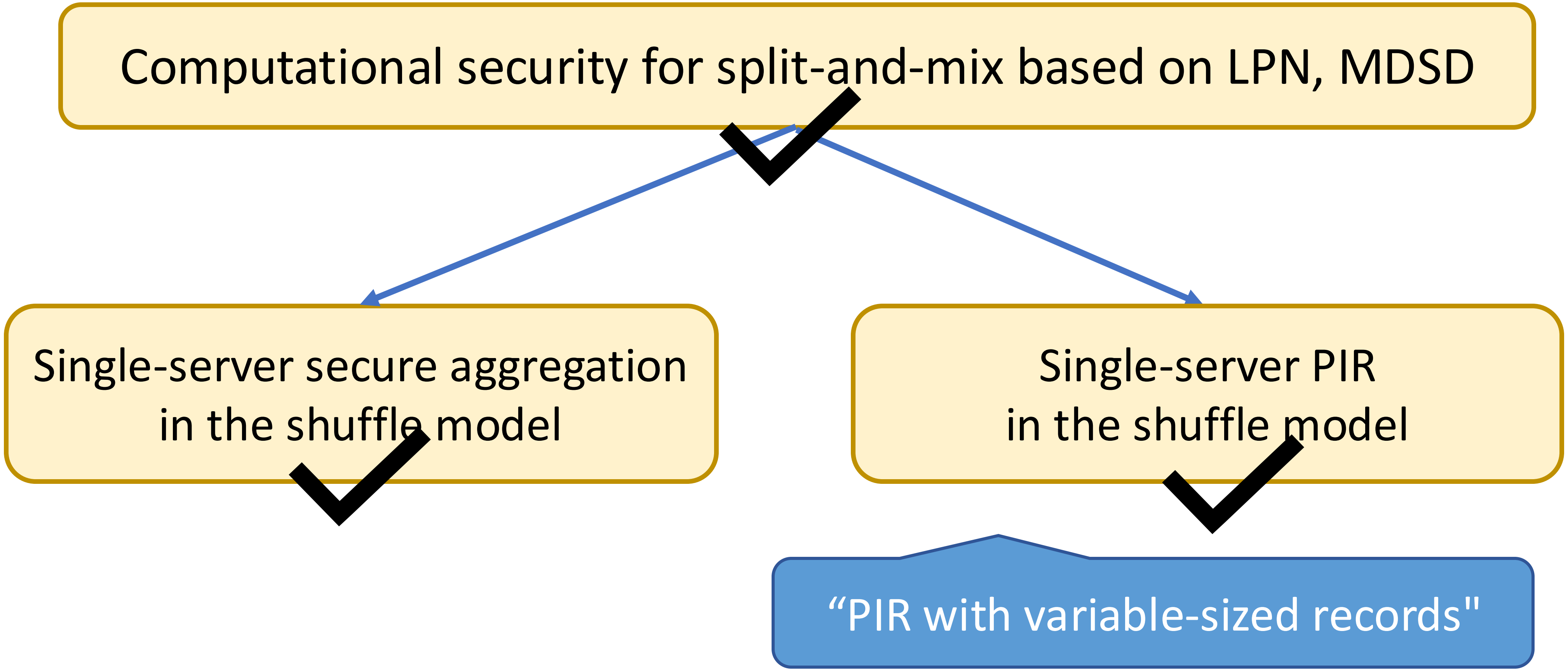
8GB database, large records (2^{18} entries of 32KB)

Bottom left is better



Summary

Thanks!

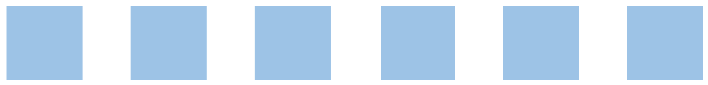


Backup slides

PIR with variable-sized records

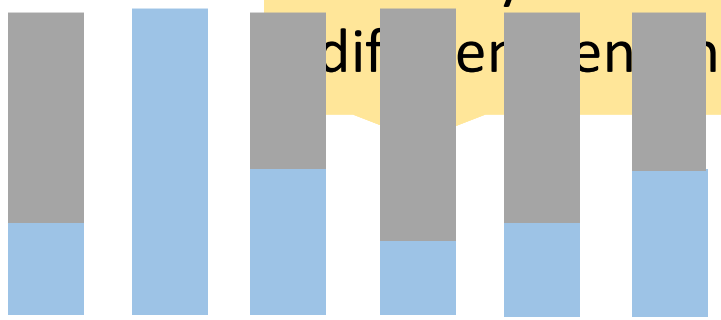
- Deploying PIR in real-world applications

Often assume the same length



Database entries of PIR in theory

They have



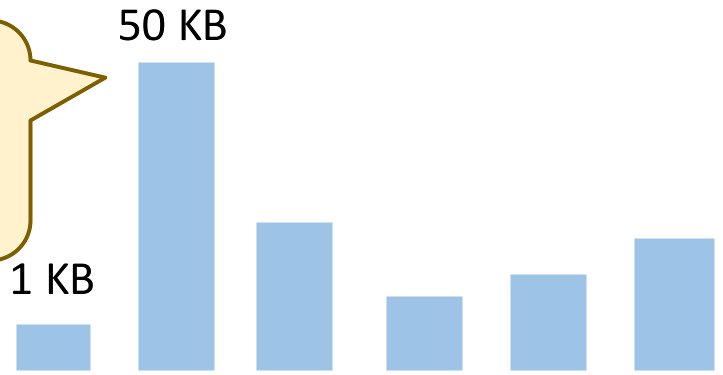
Database records in practice

To retrieve privately, it is necessary to hide record size

PIR with variable-sized records

- Padding solves the problem, but it is inefficient for some applications

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



Database records in practice

To retrieve privately, it is necessary to hide record size

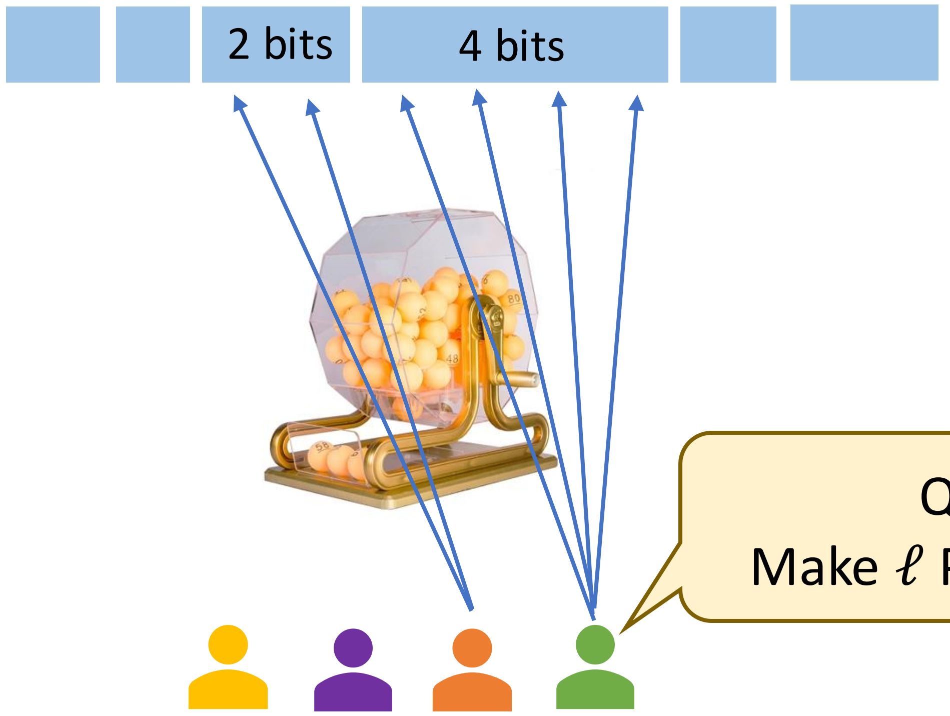
PIR with variable-sized records

- In the standard model, there is no way out
- In the shuffle model:
  - Client communication proportional to the length of the retrieved record
  - Leak only the total size of all queried records (in most cases quite benign)

PIR with variable-sized records

- A toy protocol using PIR as a black box

An n-bit database

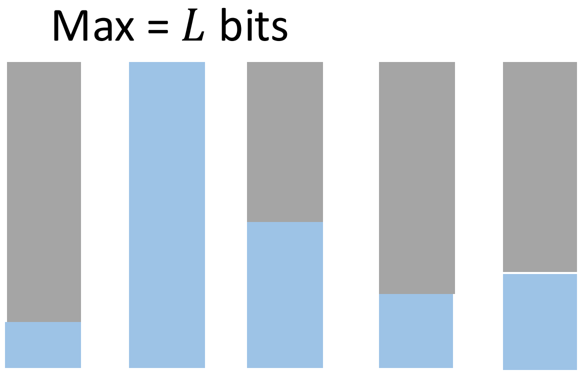


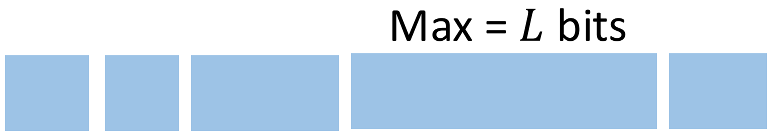
Records concatenated

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

PIR with variable-sized records

To retrieve an ℓ-bit record (out of T records of total n bits)





1 PIR query on DB of size $LT \gg n$	Polylogℓ PIR queries on DB of size n	ℓ PIR queries on DB of size n
No leakage on queried record length	Leak total length of queried records + a bit more	Leak total length of queried records
Standard model	Shuffle model, our construction	Shuffle model, toy version

Discussion

- Assuming non-colluding servers vs. assuming a two-way anonymous channel

Discussion

- Exploiting tradeoffs when designing protocols: making assumptions, relaxing security, etc.
- Guaranteeing different assumptions does not requrie the same amount of efforts: system efforts, law efforts, etc.
- The likelihood of assumptions being compromised in real-world scenarios may vary