Length Leakage in Oblivious Data Access Mechanisms

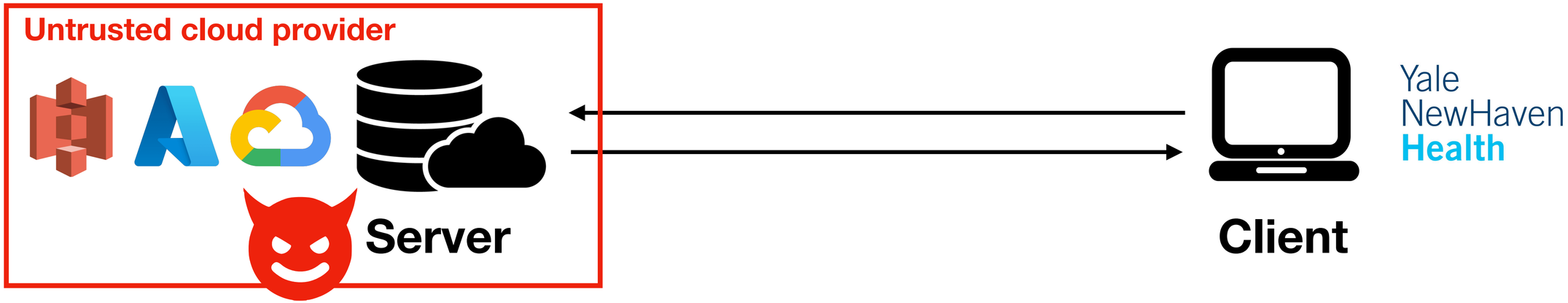
Grace Jia, Rachit Agarwal, Anurag Khandelwal



Yale

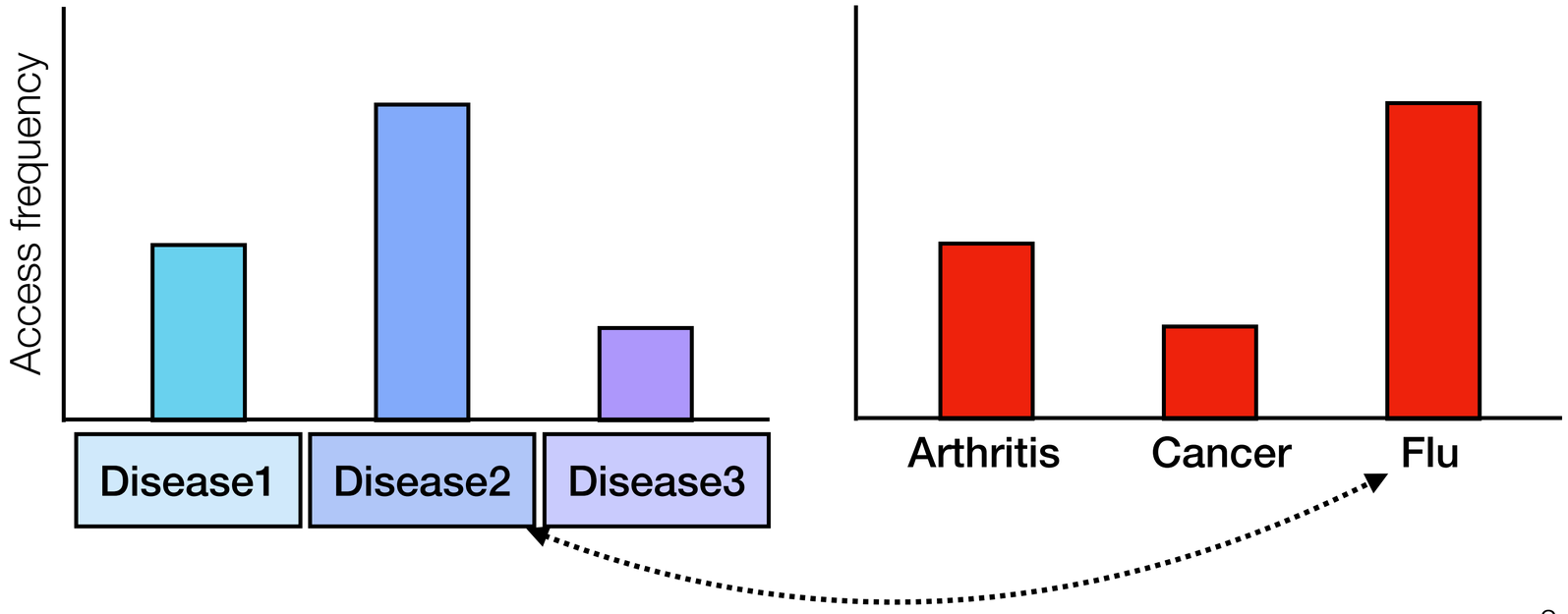
Cornell University

Access pattern leakage in cloud storage

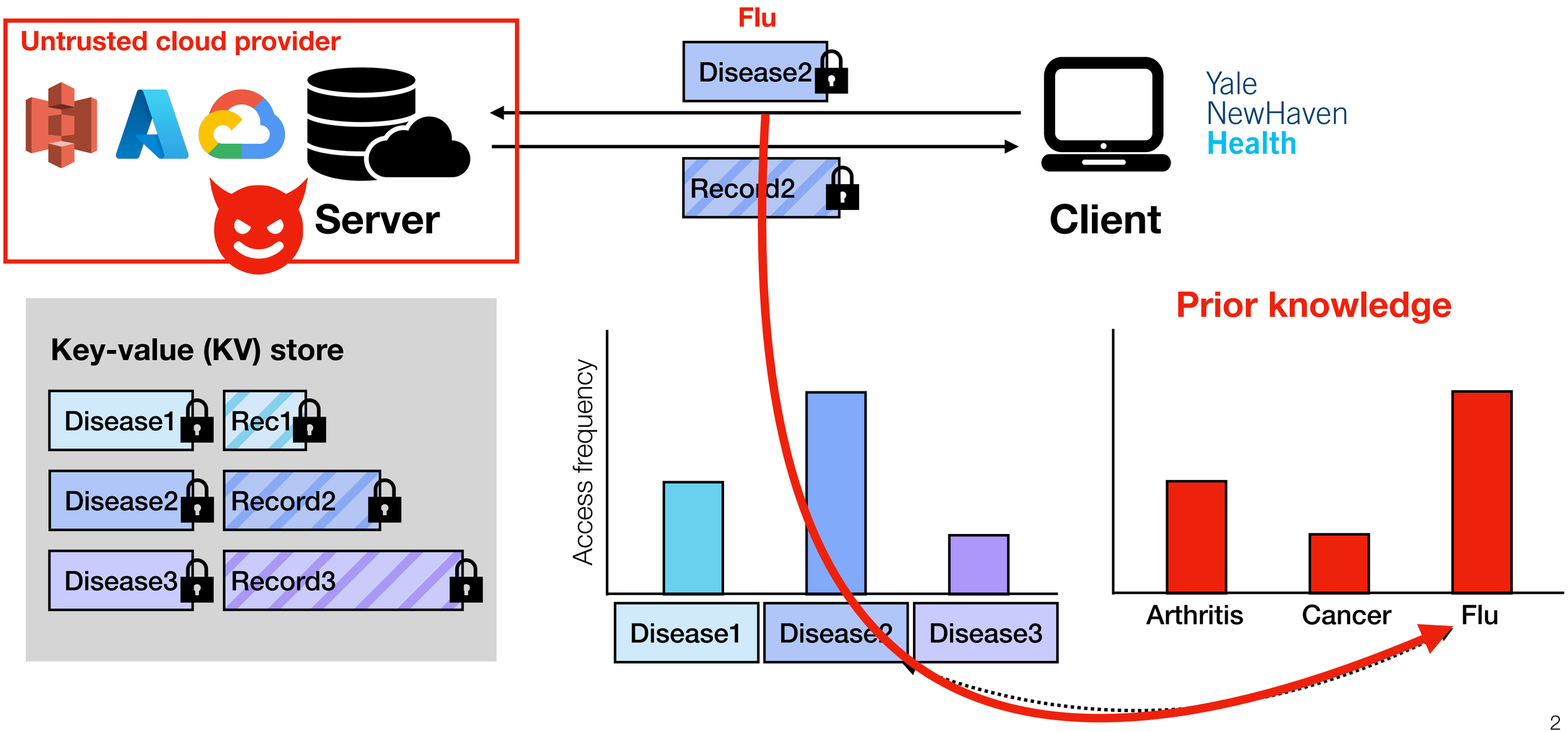


Key-value (KV) store

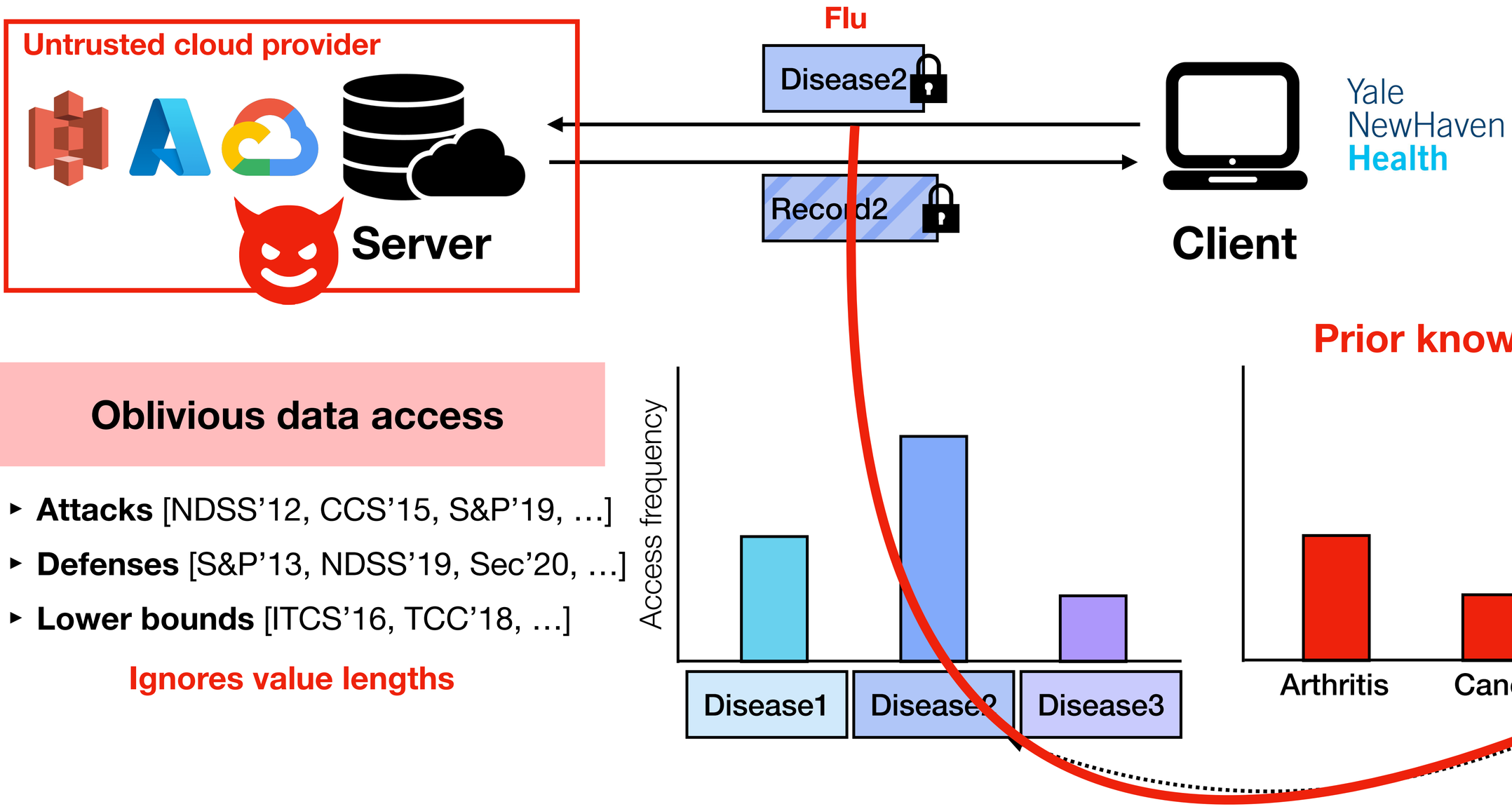
Prior knowledge



Access pattern leakage in cloud storage



Access pattern leakage in cloud storage



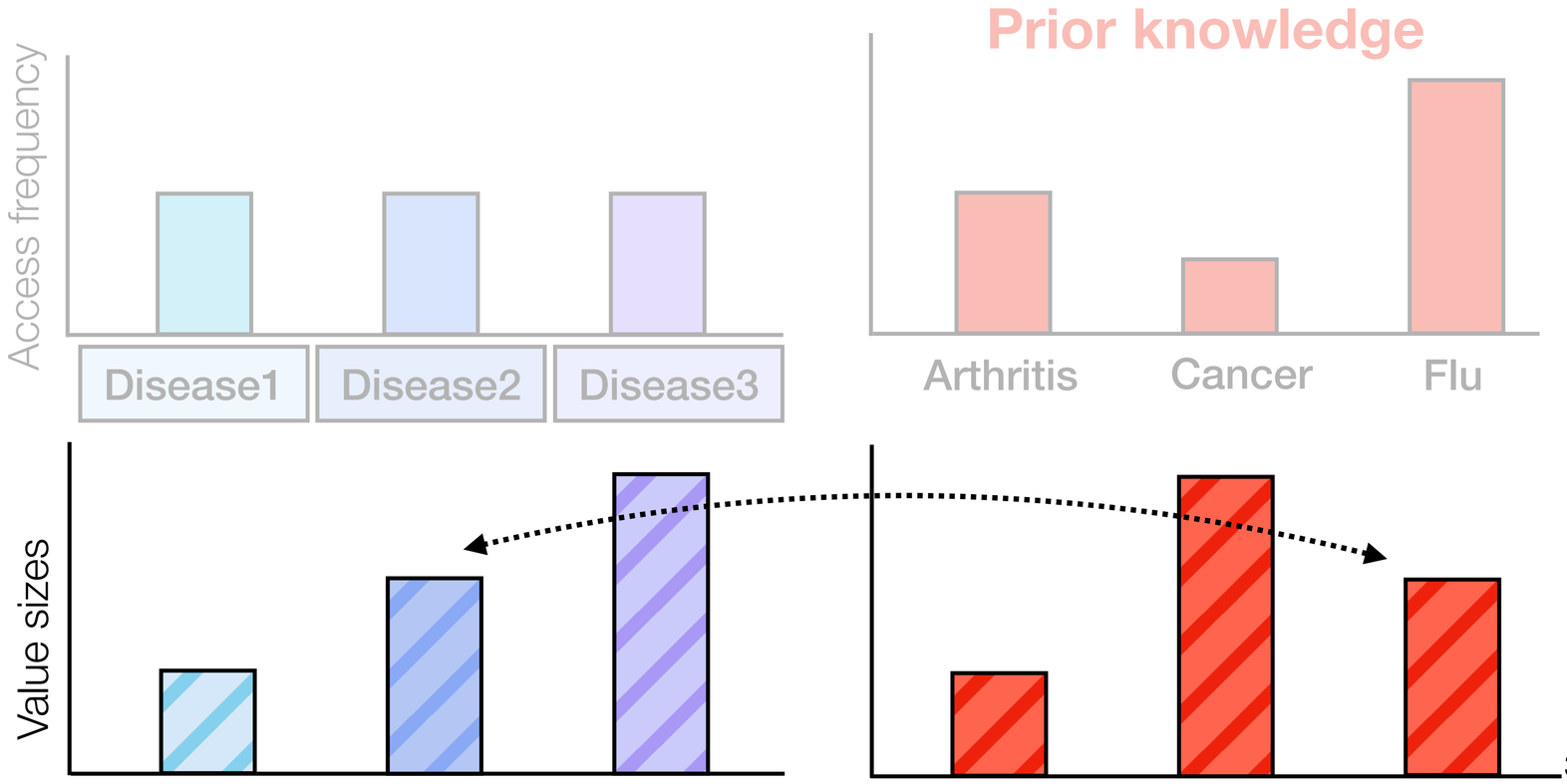
Oblivious data access

- **Attacks** [NDSS'12, CCS'15, S&P'19, …]
- **Defenses** [S&P'13, NDSS'19, Sec'20, …]
- **Lower bounds** [ITCS'16, TCC'18, …]

Ignores value lengths

The problem of length leakage

Oblivious stores are still vulnerable to length leakage attacks
[CCS'16, CCS'18, …]



Our work

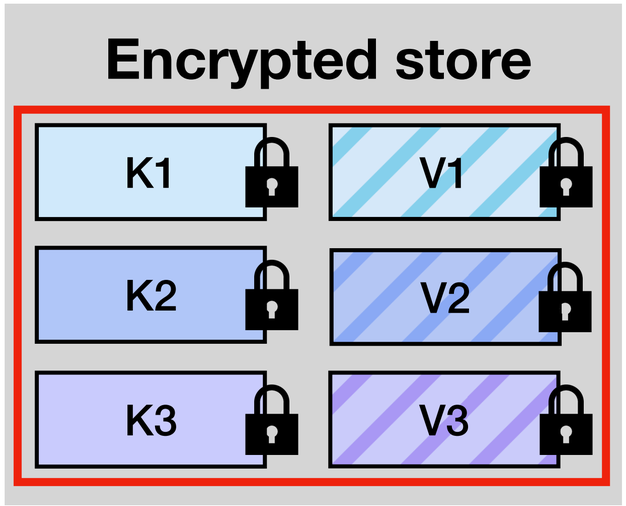
New **security model** combining access pattern and length leakage

New **three-way tradeoff** between **security, bandwidth, and storage**

Performance lower bounds under given security

Secure constructions that achieve lower bounds

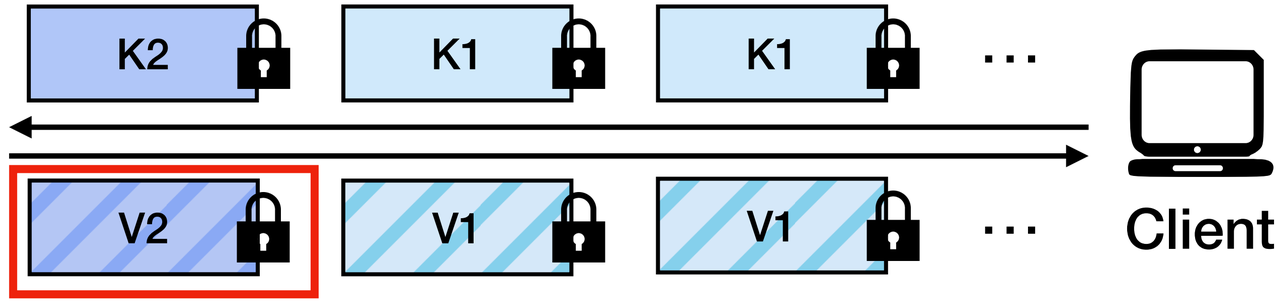
Informal security definition



Passive persistent adversary

Server

Transcript of accesses



Bandwidth footprint = no. of bytes fetched per query

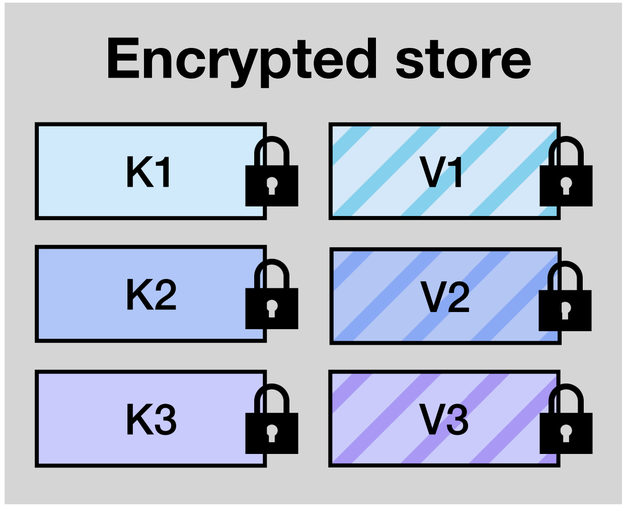
Storage footprint = no. of bytes to store all values

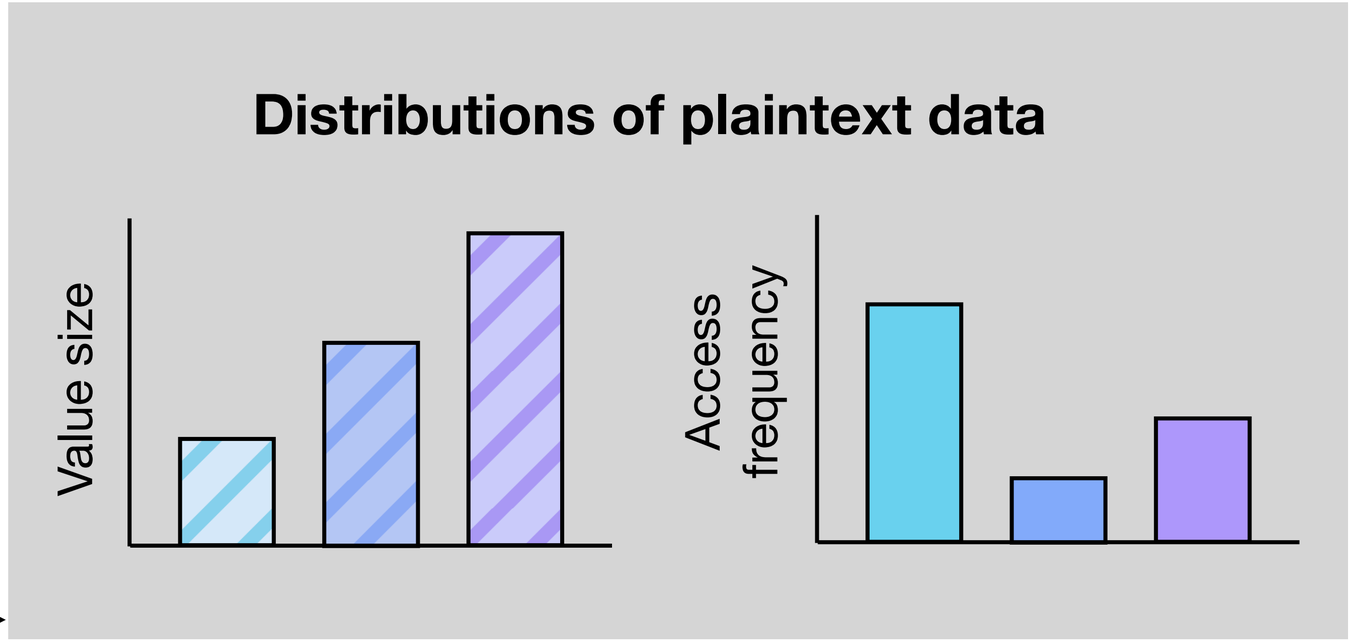
No. of encrypted keys

Baseline security (ROR-CDLA):
Mapping of plaintext to encrypted keys is always hidden

⇒ same value sizes, uniform access distribution

Informal security definition





dependent on

Bandwidth footprint

Storage footprint

No. of encrypted keys

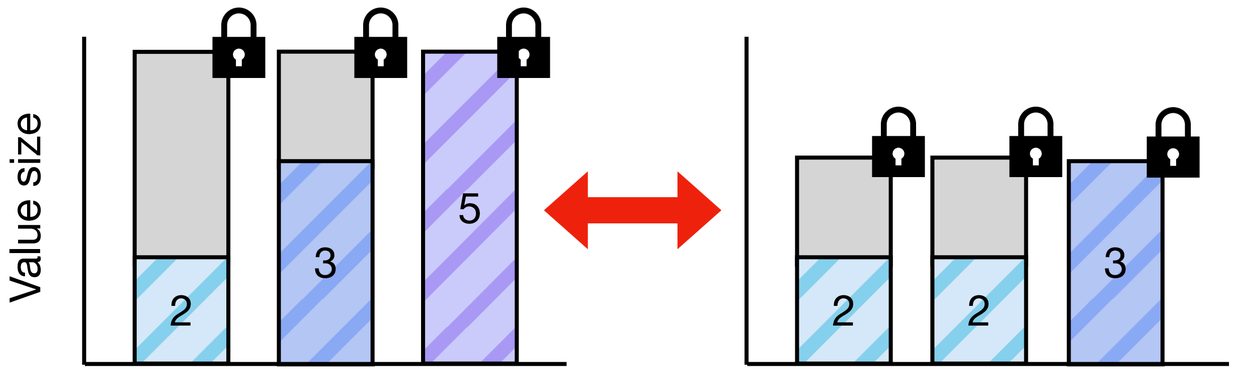
Considered leakage profiles

Largest value size	Value sizes	Access distribution	Value sizes & access distribution

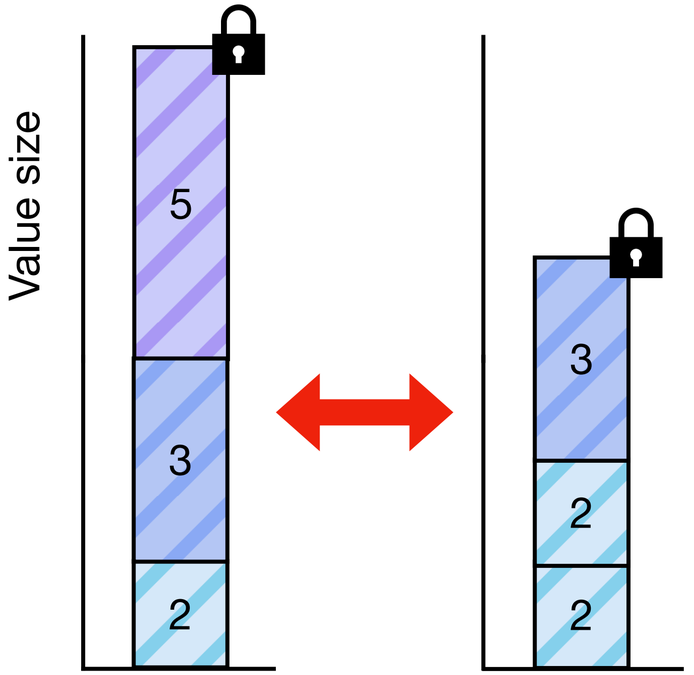
Baseline security (ROR-CDLA): Mapping of plaintext to encrypted keys is always hidden

Implications of leakage profiles

Design #1: Padding
(e.g., in oblivious access mechanisms)



Design #2: Bin-packing
(e.g., size-locked index [Sec'21])

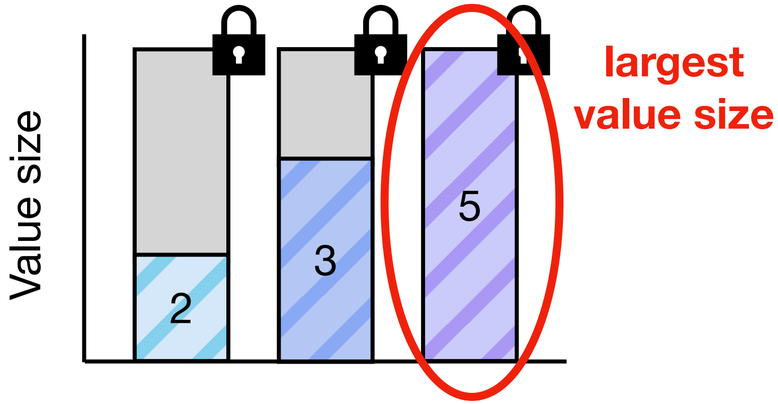


Leaks largest value size

Leaks sum of value sizes

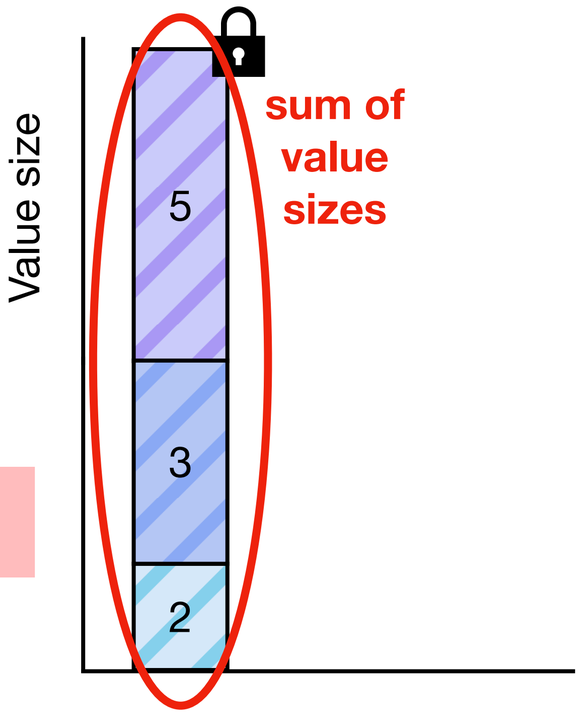
A new three-way tradeoff

Design #1: Padding
(e.g., in oblivious access mechanisms)



Less leakage

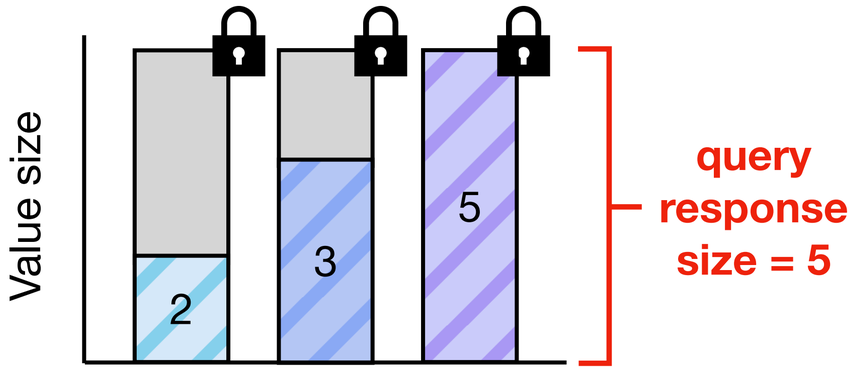
Design #2: Bin-packing
(e.g., size-locked index [Sec'21])



More leakage

A new three-way tradeoff

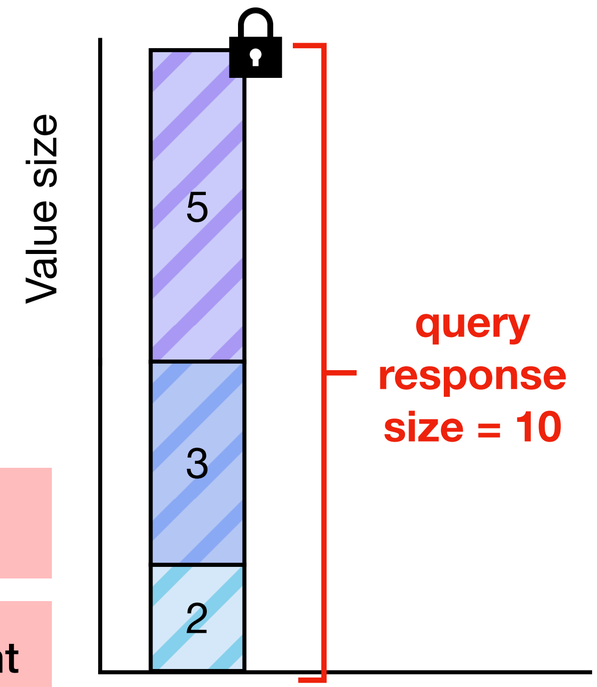
Design #1: Padding
(e.g., in oblivious access mechanisms)



Less leakage

Lower bandwidth footprint

Design #2: Bin-packing
(e.g., size-locked index [Sec'21])

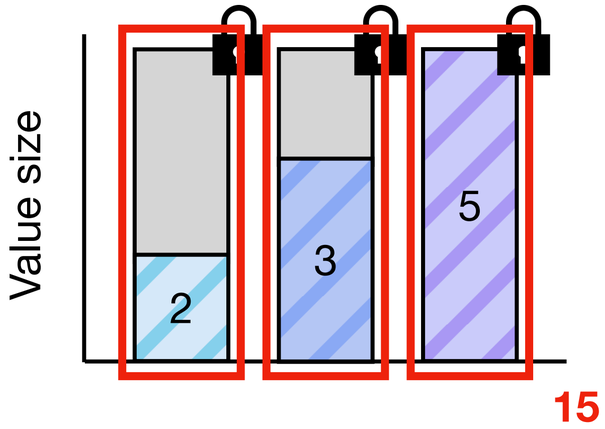


More leakage

Higher bandwidth footprint

A new three-way tradeoff

Design #1: Padding
(e.g., in oblivious access mechanisms)

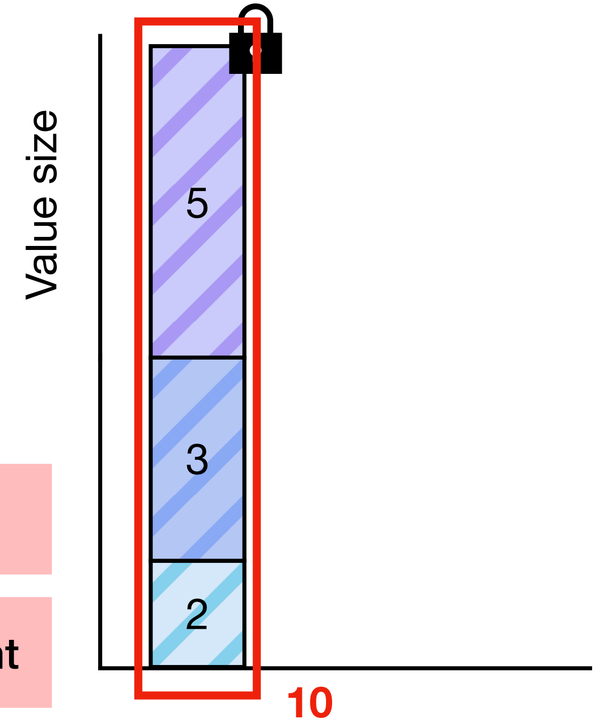


Less leakage

Lower bandwidth footprint

Higher storage footprint

Design #2: Bin-packing
(e.g., size-locked index [Sec'21])



More leakage

Higher bandwidth footprint

Lower storage footprint

Generalizing the tradeoff

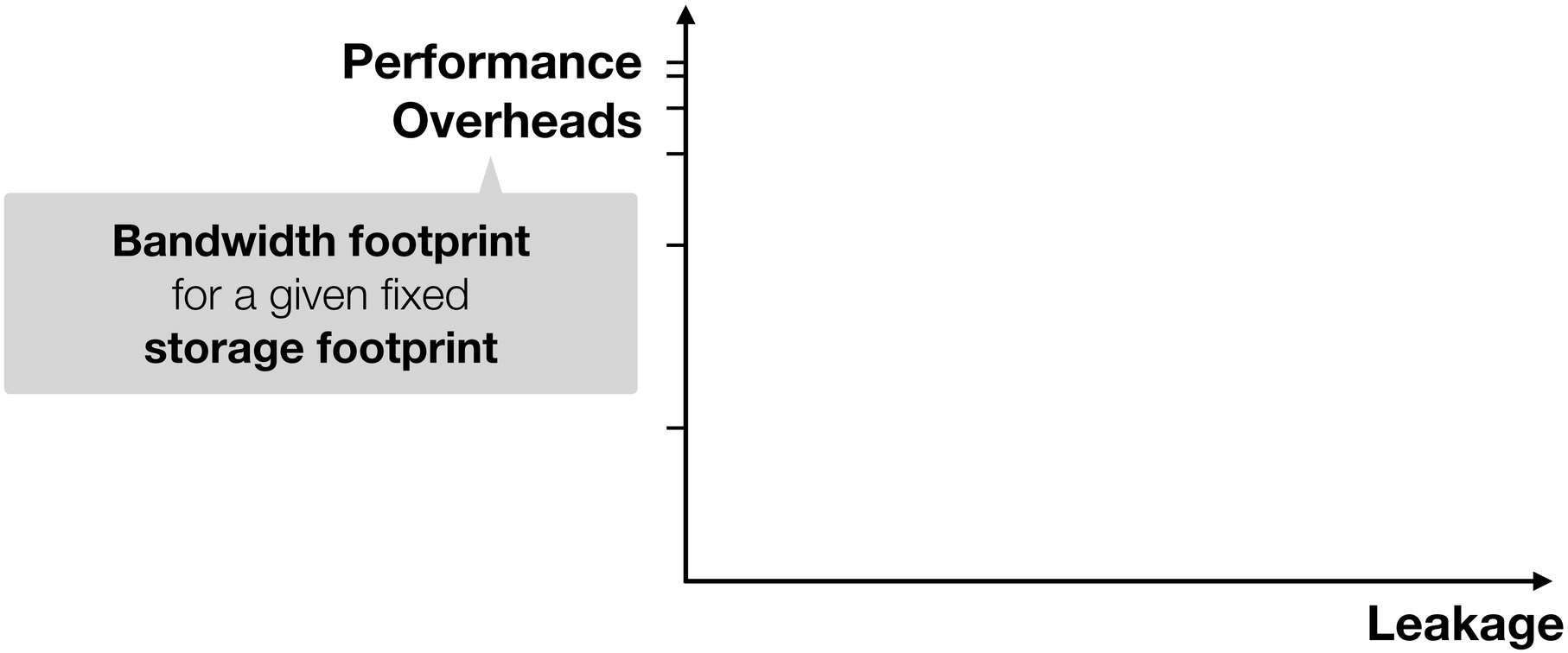
Considered leakage profiles:

Largest value size

Value sizes

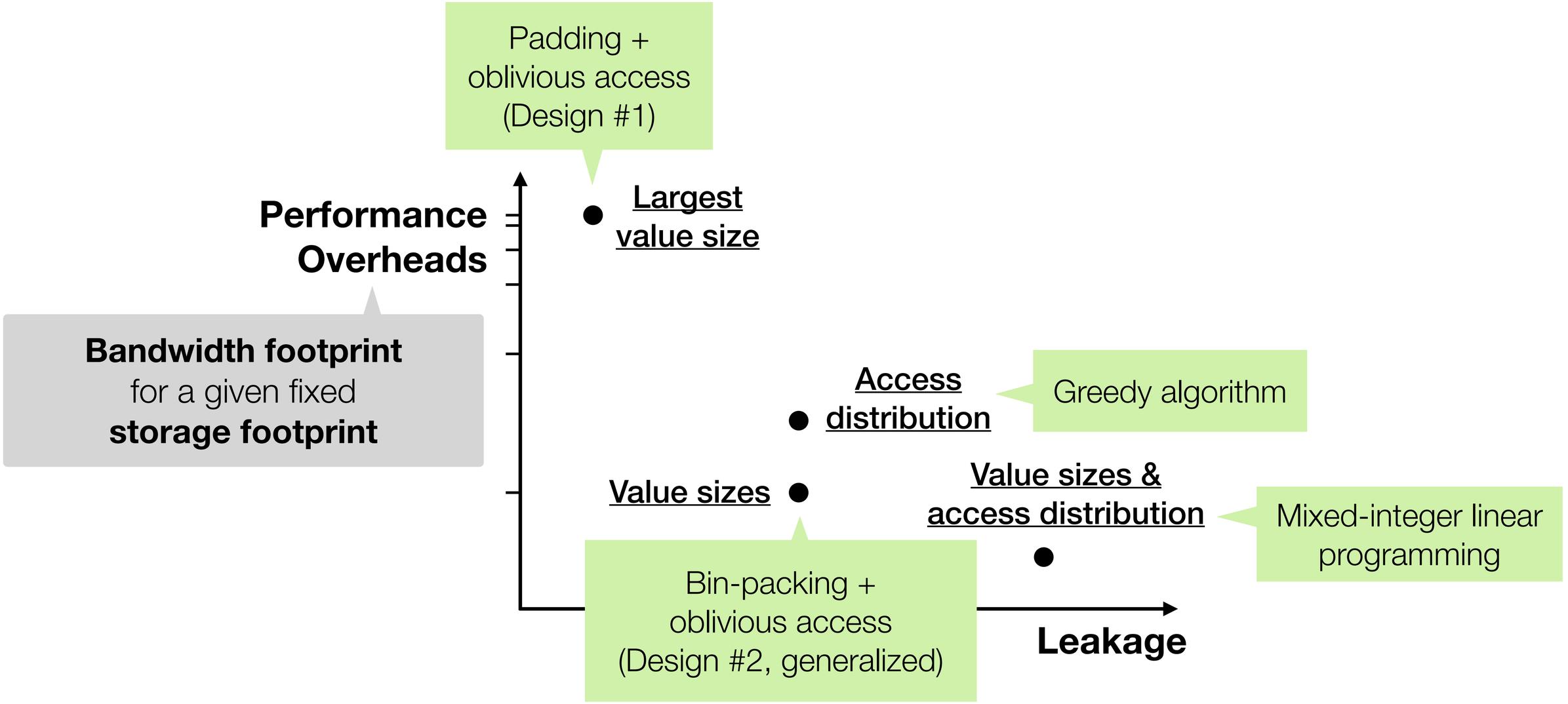
Access distribution

Value sizes & access distribution

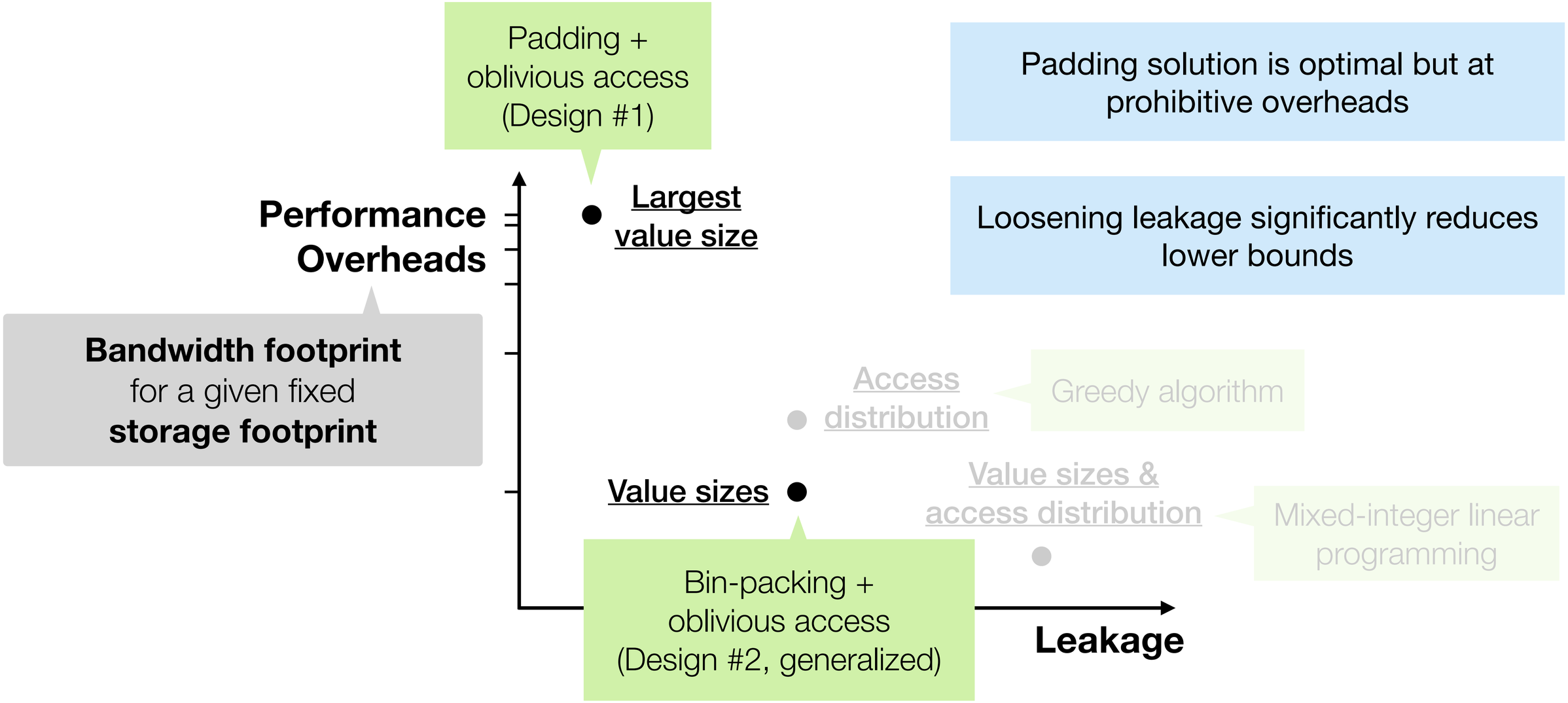


Bandwidth footprint
for a given fixed
storage footprint

Generalizing the tradeoff



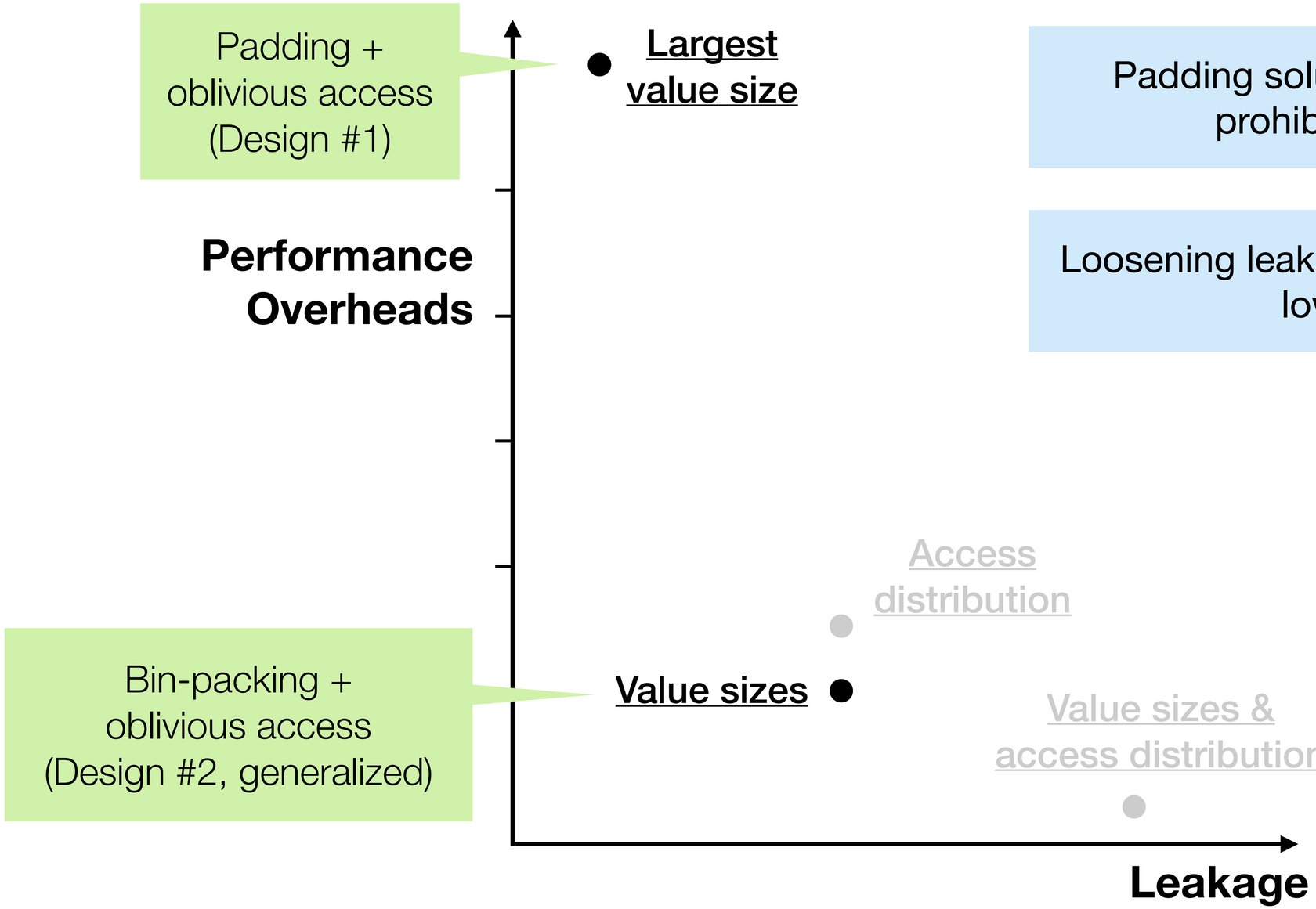
Generalizing the tradeoff



Padding solution is optimal but at prohibitive overheads

Loosening leakage significantly reduces lower bounds

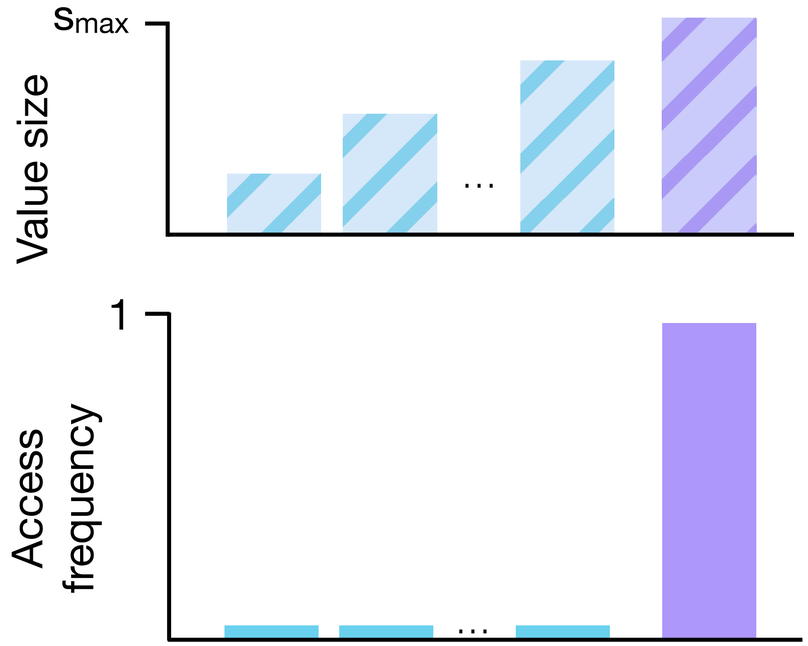
Generalizing the tradeoff



Padding solution is optimal but at prohibitive overheads

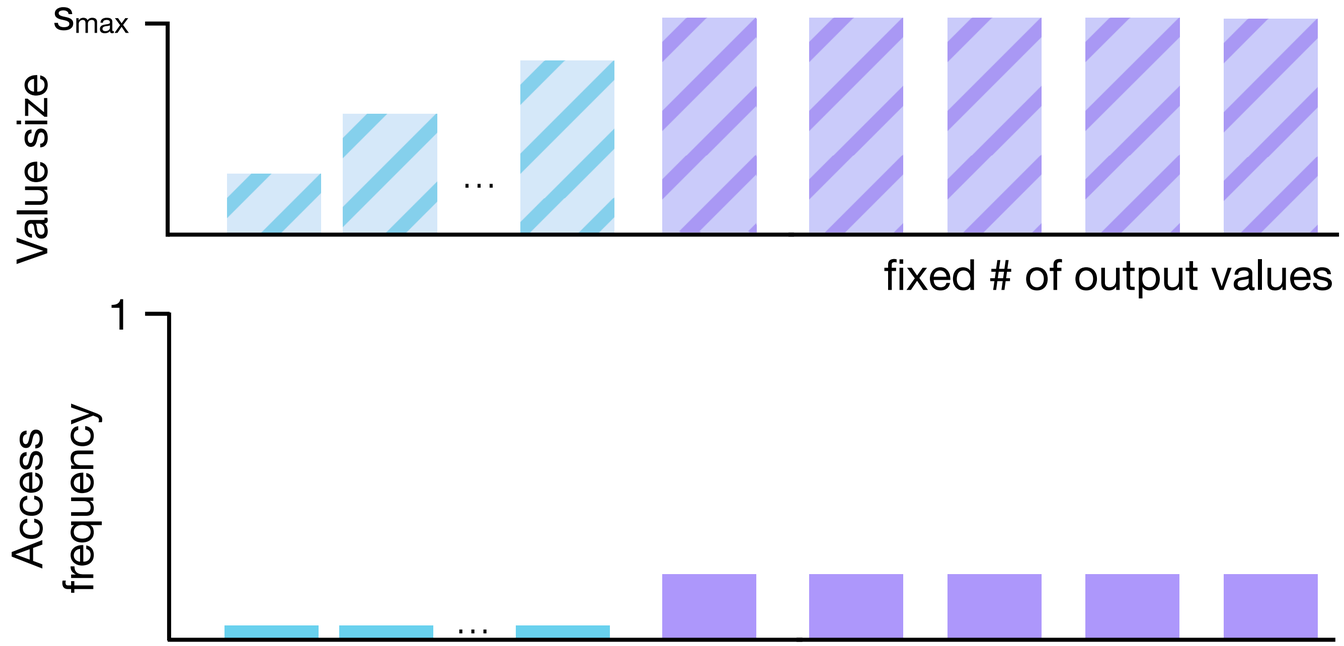
Loosening leakage significantly reduces lower bounds

Padding optimal for leaking largest value size



Use Pancake [Sec'20] to hide access distribution

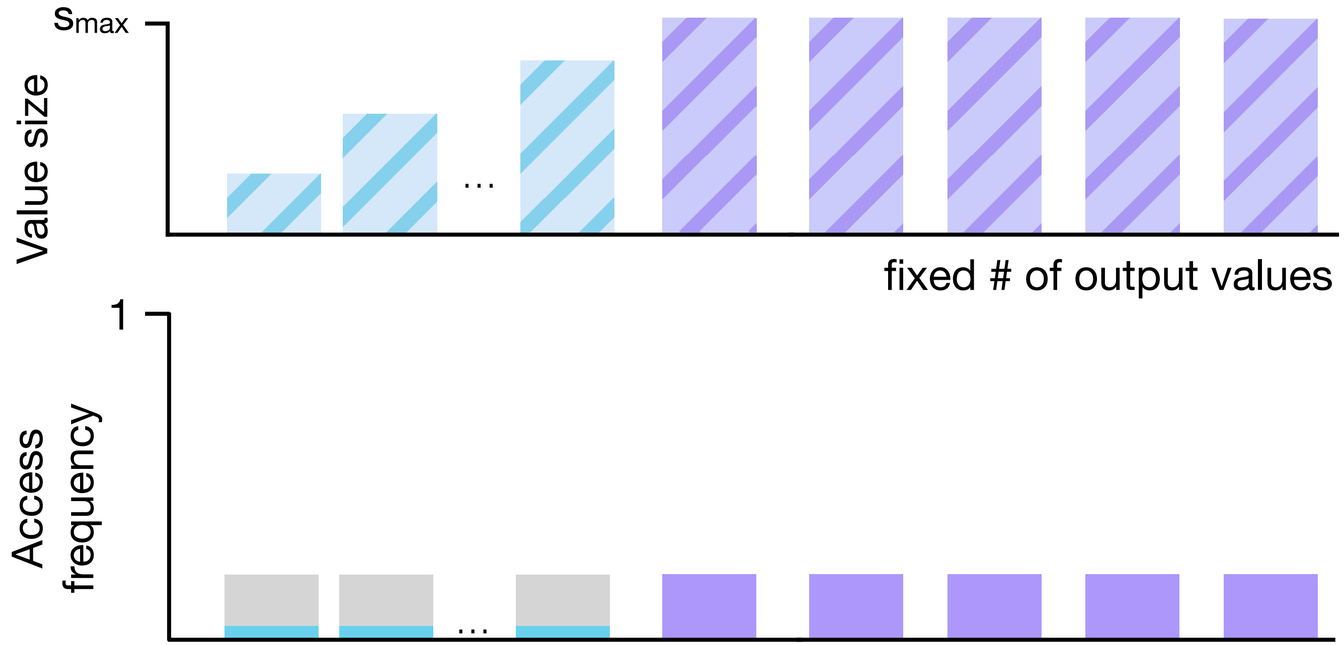
Padding optimal for leaking largest value size



Use Pancake [Sec'20] to hide access distribution

1. Maximize replicas of popular keys and divide accesses across them

Padding optimal for leaking largest value size

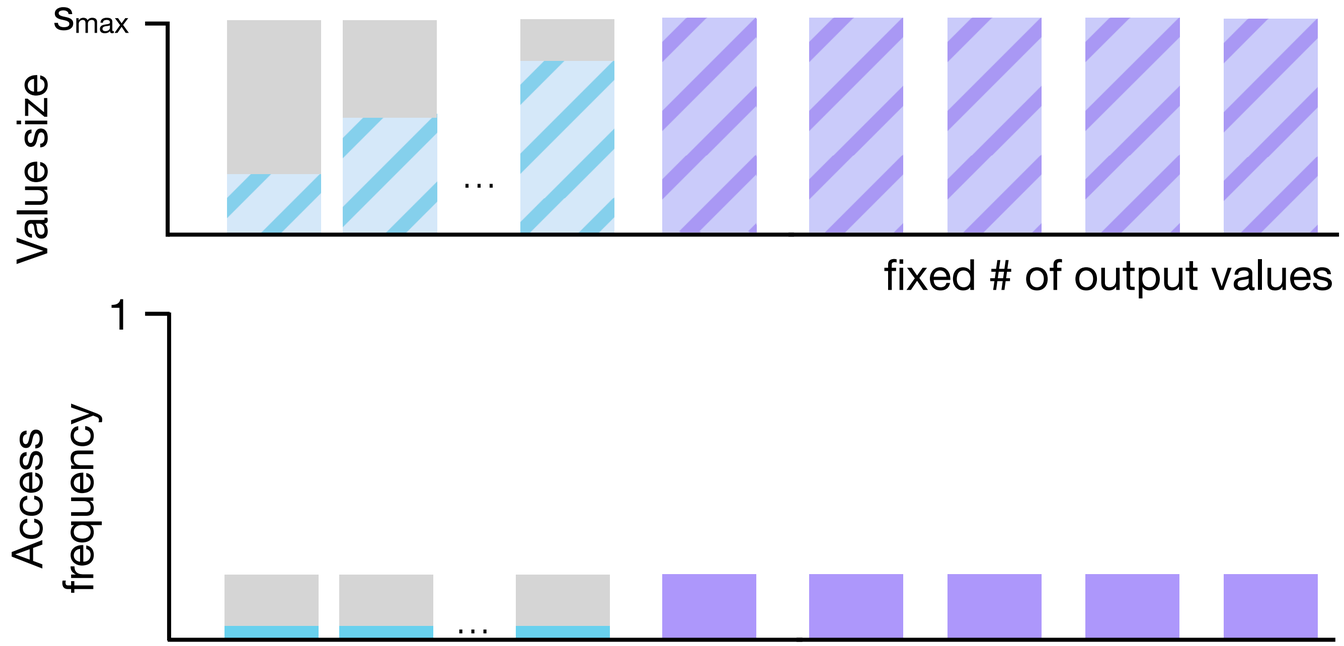


Use Pancake [Sec'20] to hide access distribution

1. Maximize replicas of popular keys and divide accesses across them

2. Inject minimal traffic to unpopular keys

Padding optimal for leaking largest value size



Use Pancake [Sec'20] to hide access distribution

1. Maximize replicas of popular keys and divide accesses across them
2. Inject minimal traffic to unpopular keys

For ROR-CDLA security, pad values (leaks only largest value size)

Resulting scheme "Padded Pancake" actually achieves bandwidth lower bound!

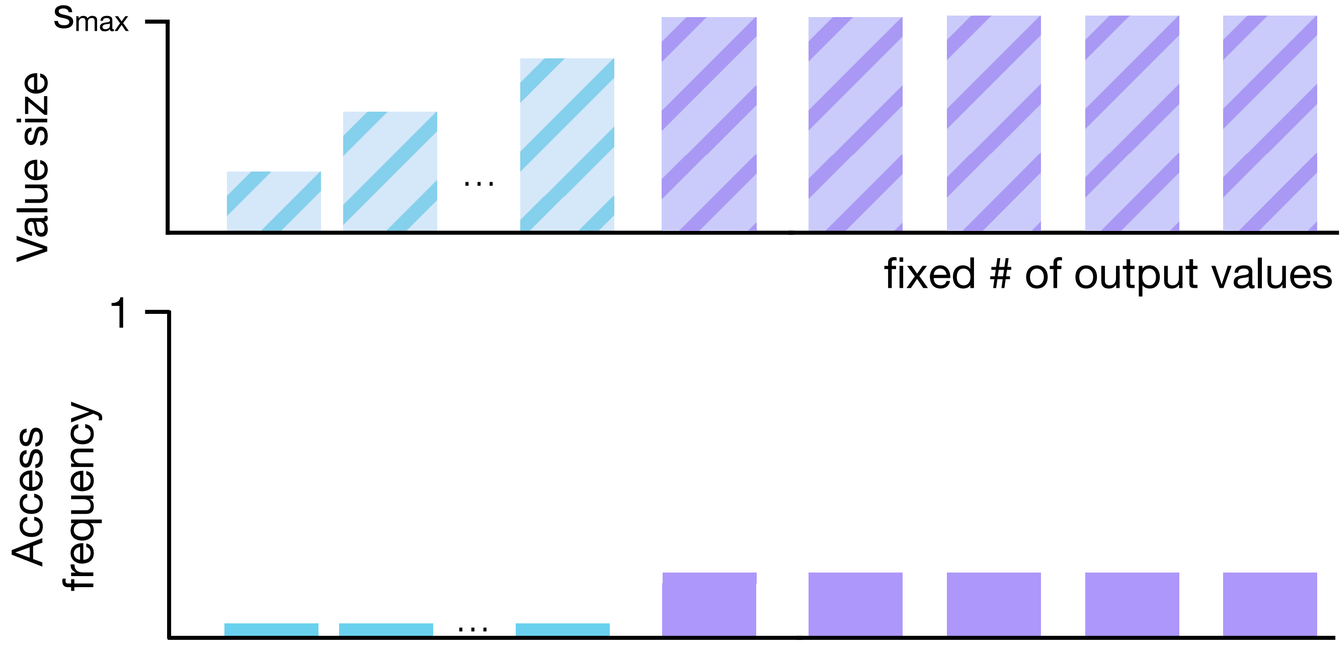
Padding optimal for leaking largest value size

Lower bound: Given storage footprint $c \cdot s_{max}$, bandwidth footprint of any ROR-CDLA scheme must be $\geq d \cdot s_{max}$.

constant for large n (number of plaintext keys)

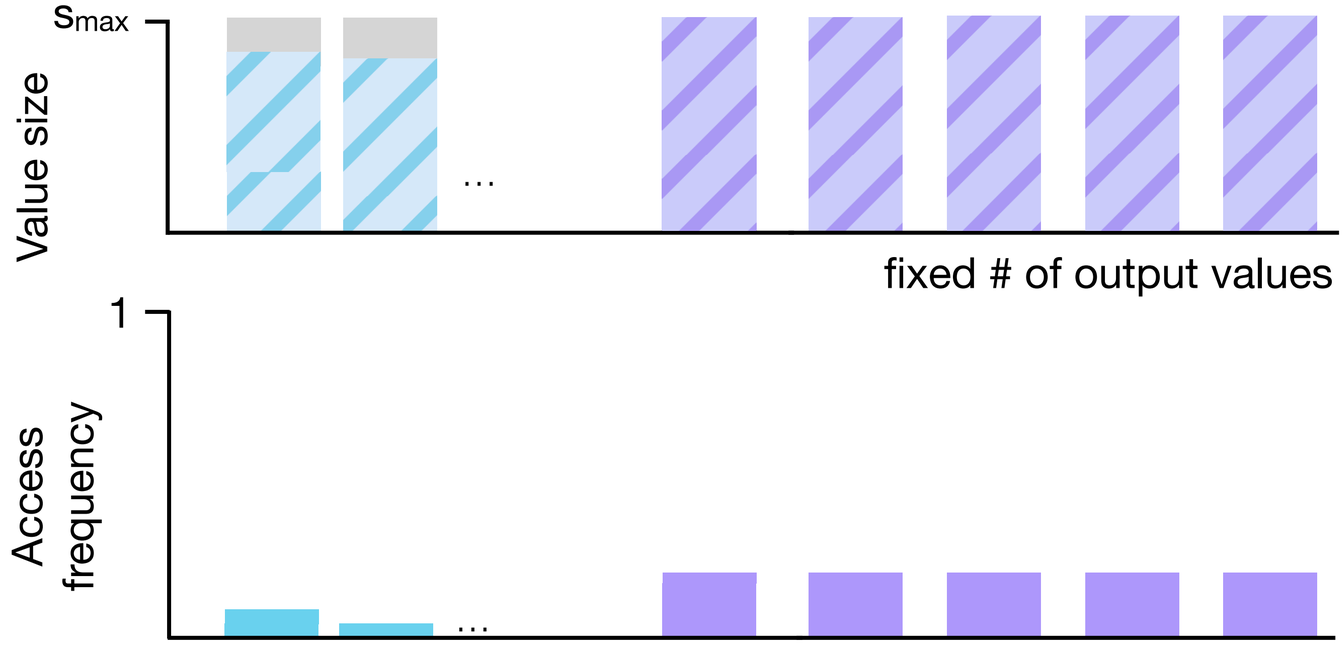
Expensive for datasets with wide range of value sizes,
e.g. 30× bandwidth overhead given 2× storage overhead in evaluation

Benefit of bin-packing for leaking value sizes



ROR-CDLA scheme can now:

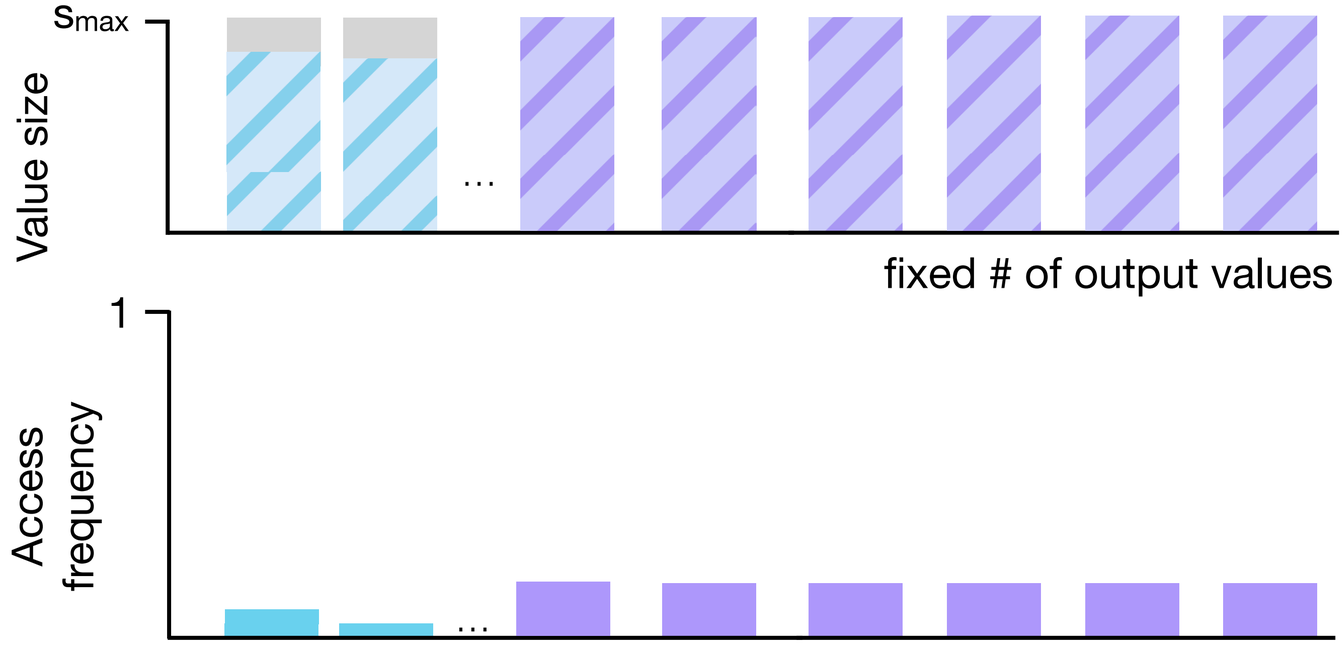
Benefit of bin-packing for leaking value sizes



ROR-CDLA scheme can now:

- Bin-pack unpopular values together instead of padding

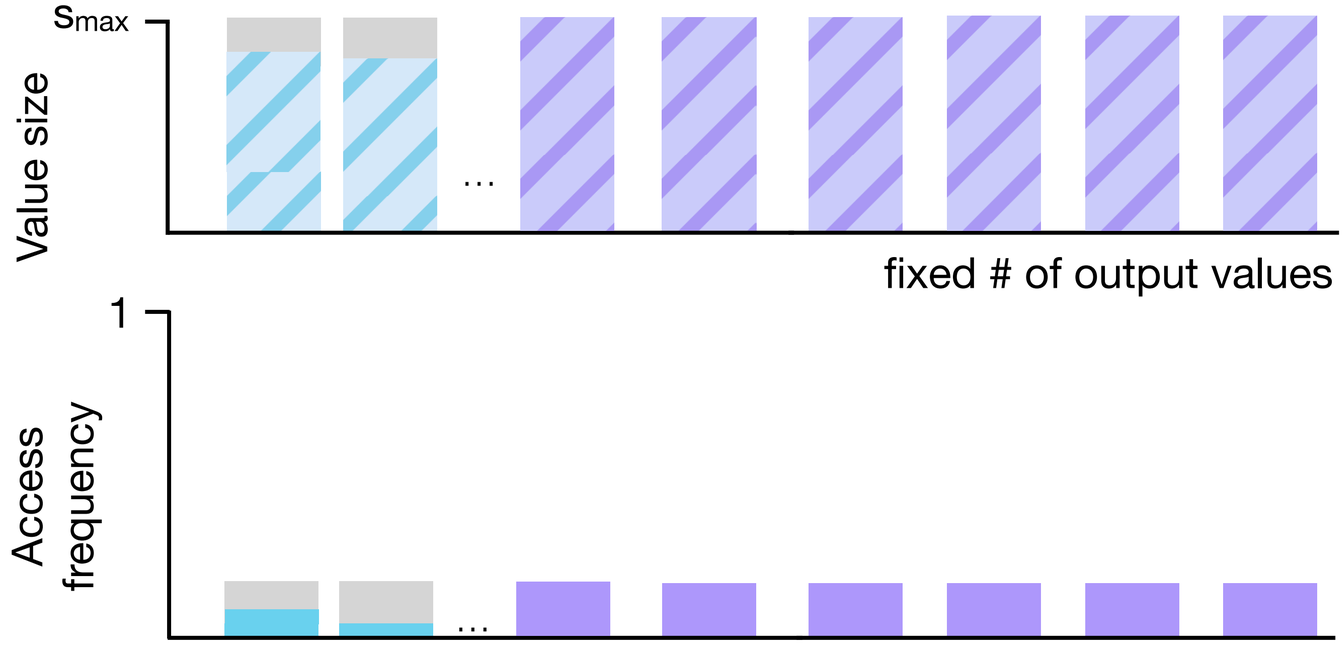
Benefit of bin-packing for leaking value sizes



ROR-CDLA scheme can now:

- Bin-pack unpopular values together instead of padding
- Make more replicas of popular keys

Benefit of bin-packing for leaking value sizes



ROR-CDLA scheme can now:

- Bin-pack unpopular values together instead of padding
- Make more replicas of popular keys
- Inject less traffic to unpopular keys

Resulting scheme “Stuffed Pancake” achieves bandwidth lower bound!

Benefit of bin-packing for leaking value sizes

Lower bound: Given storage footprint $c \cdot s_{max}$, bandwidth footprint of any ROR-CDLA scheme must be $\geq d^* \cdot s_{max}$ where $d^* \leq d$

smaller multiple, depending on value sizes

Value size

Access frequency

- Make more replicas of popular keys

Significant improvement for evaluated scenarios: 30× to 6× bandwidth overhead

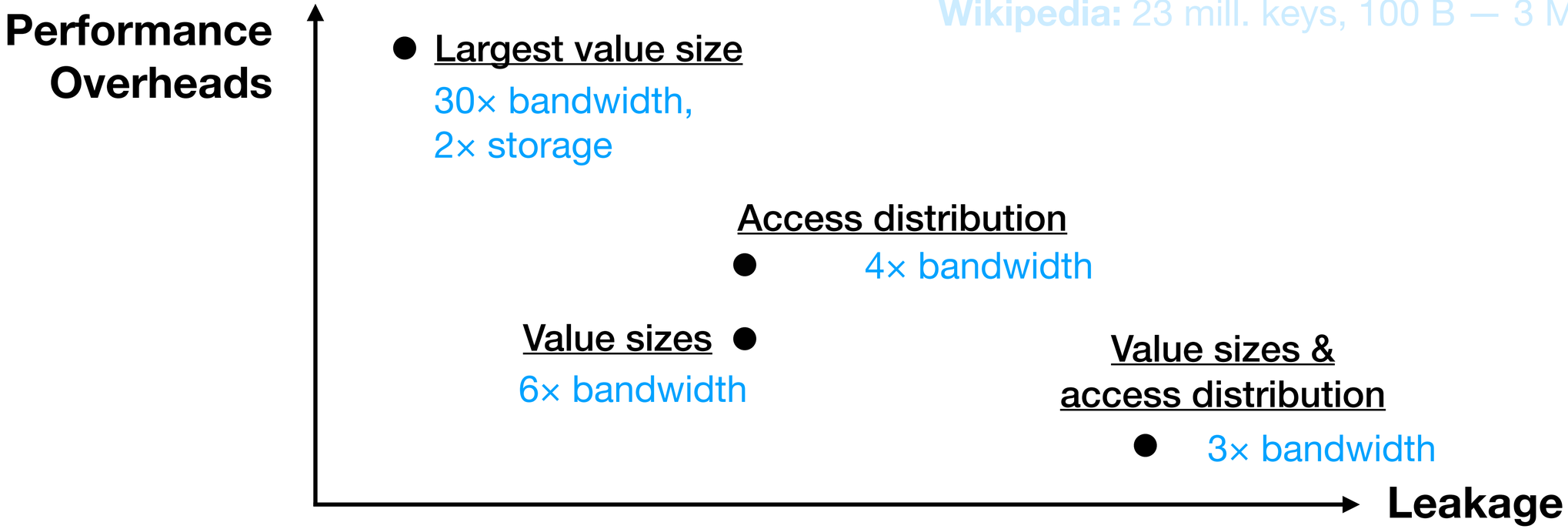
Resulting scheme “Stuffed Pancake” achieves bandwidth lower bound!

Bridging theory and practice

Evaluation on real datasets

Twitter: 3 mill. keys, 100 — 300 B values

Wikipedia: 23 mill. keys, 100 B — 3 MB values



Conclusion

New **security model** combining access pattern and length leakage

New **three-way tradeoff** between **security, bandwidth, and storage**

Performance lower bounds under given security

Secure constructions that achieve lower bounds

Future work

- Attacks exploiting leakage profiles
- Other leakage profiles
- Extension to ORAM, compression, etc.