NSAC

Network Security and Applied Cryptography Laboratory

http://crypto.cs.stonybrook.edu

Usable PIR

NDSS '08, San Diego, CA

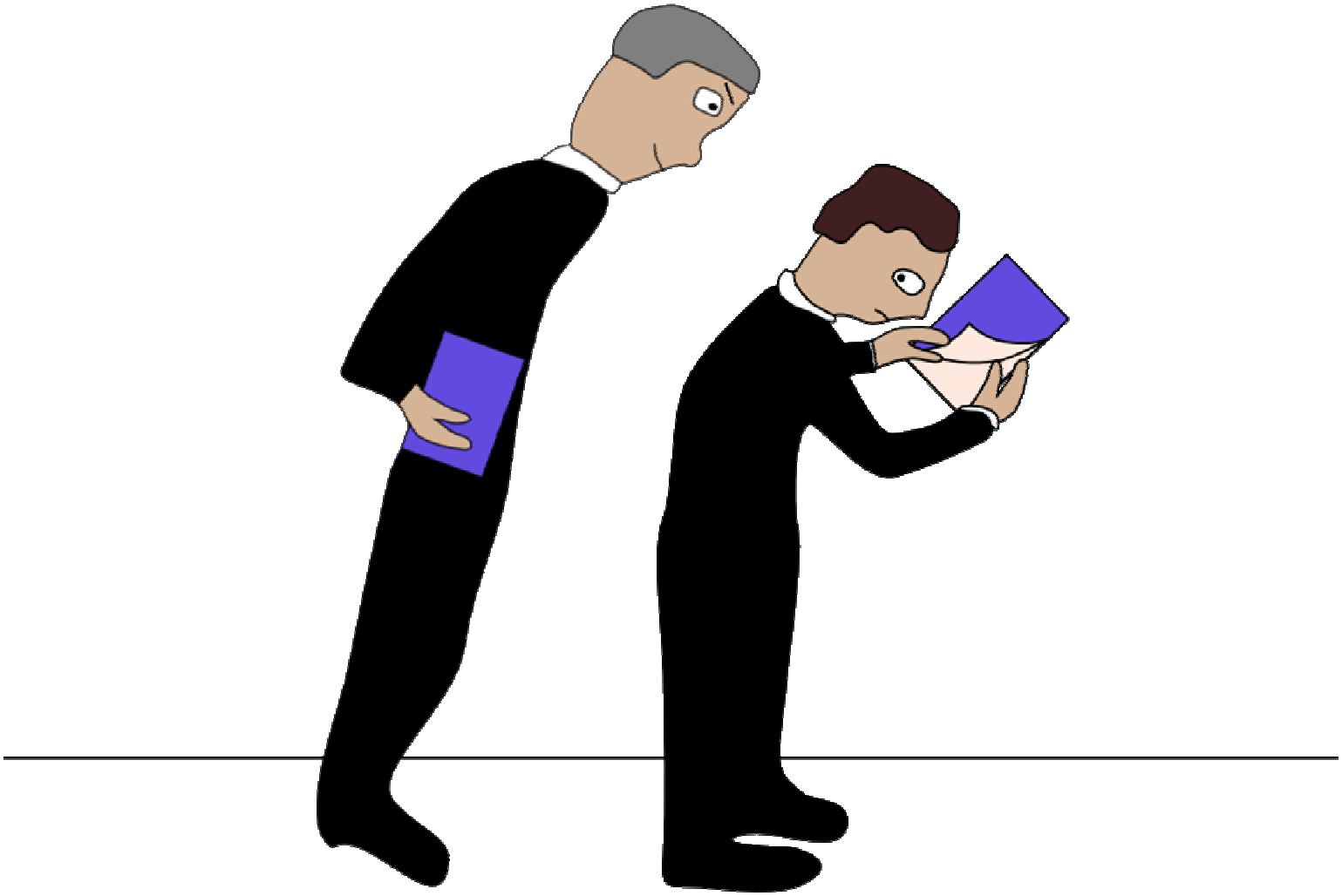
Peter Williams
petertw@cs.stonybrook.edu

Radu Sion
sion@cs.stonybrook.edu

STONY BROOK
COMPUTER SCIENCE

National Science Foundation
WHERE DISCOVERIES BEGIN

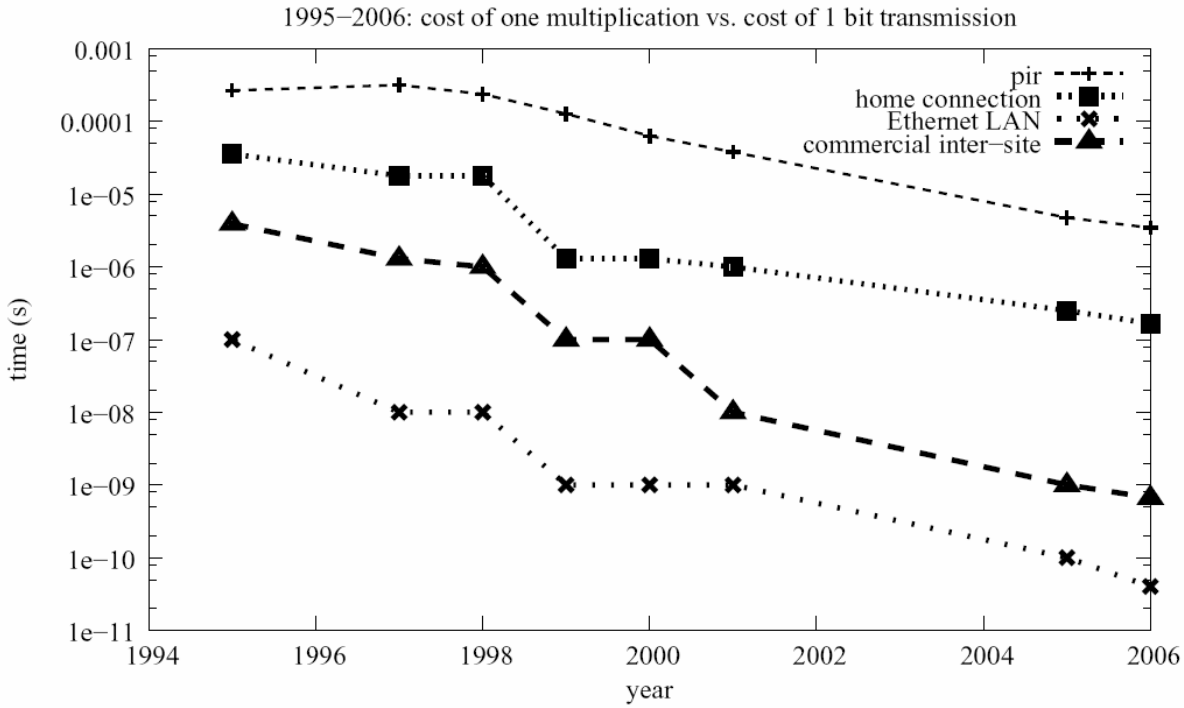
Overview: PIR



Types of PIR

- Trivial - Download entire database
- “Pretty good” PIR
- Information Theoretic PIR
- Multiple non-colluding servers
- Single Server Computational PIR
- Secure hardware

Past: cPIR is impractical

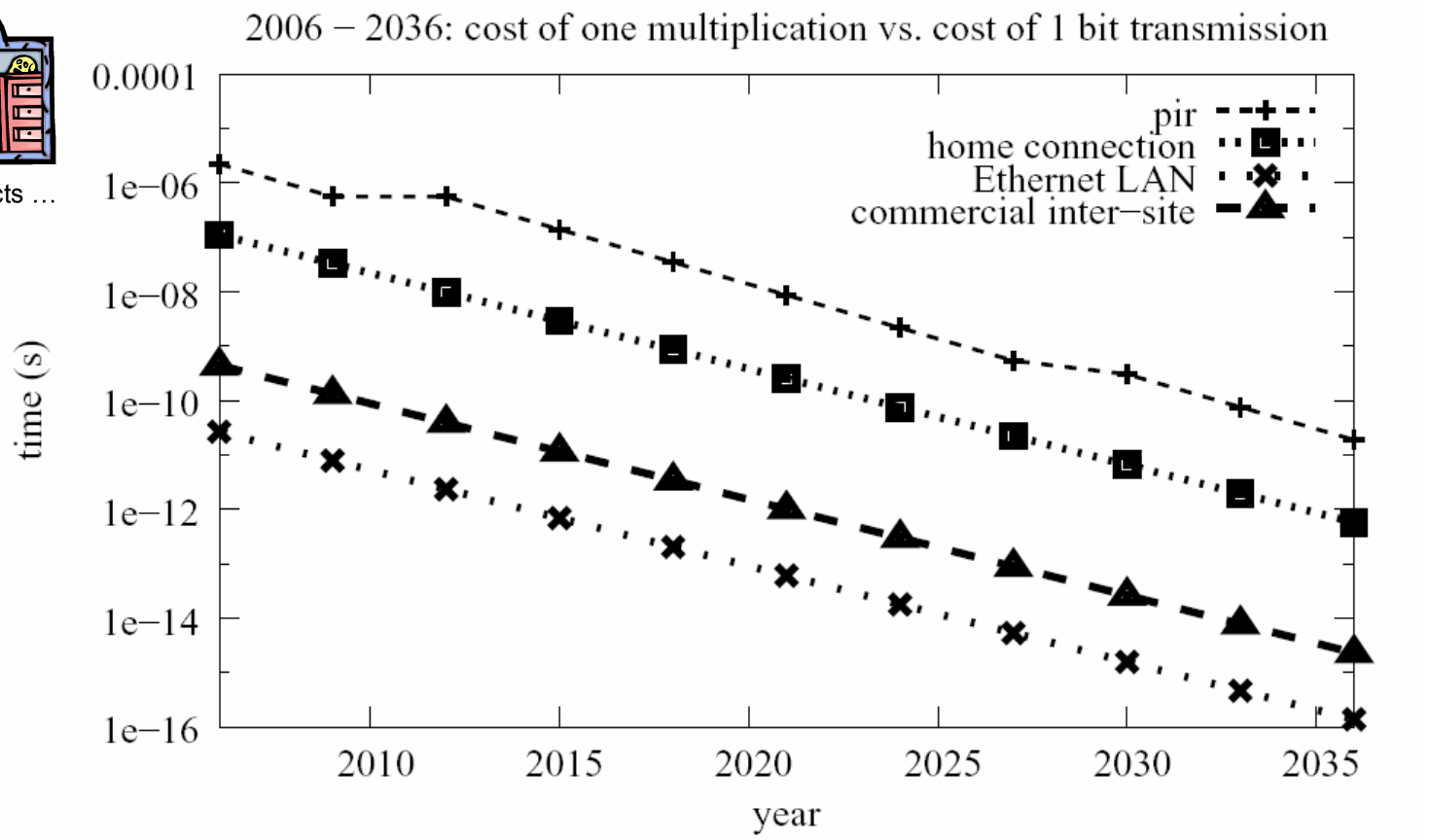


Comparison between the time required to perform PIR and the time taken to transfer the database, between 1995 and 2005. (logarithmic)

Sion & Carbunar @ NDSS 2007

Future: cPIR is impractical

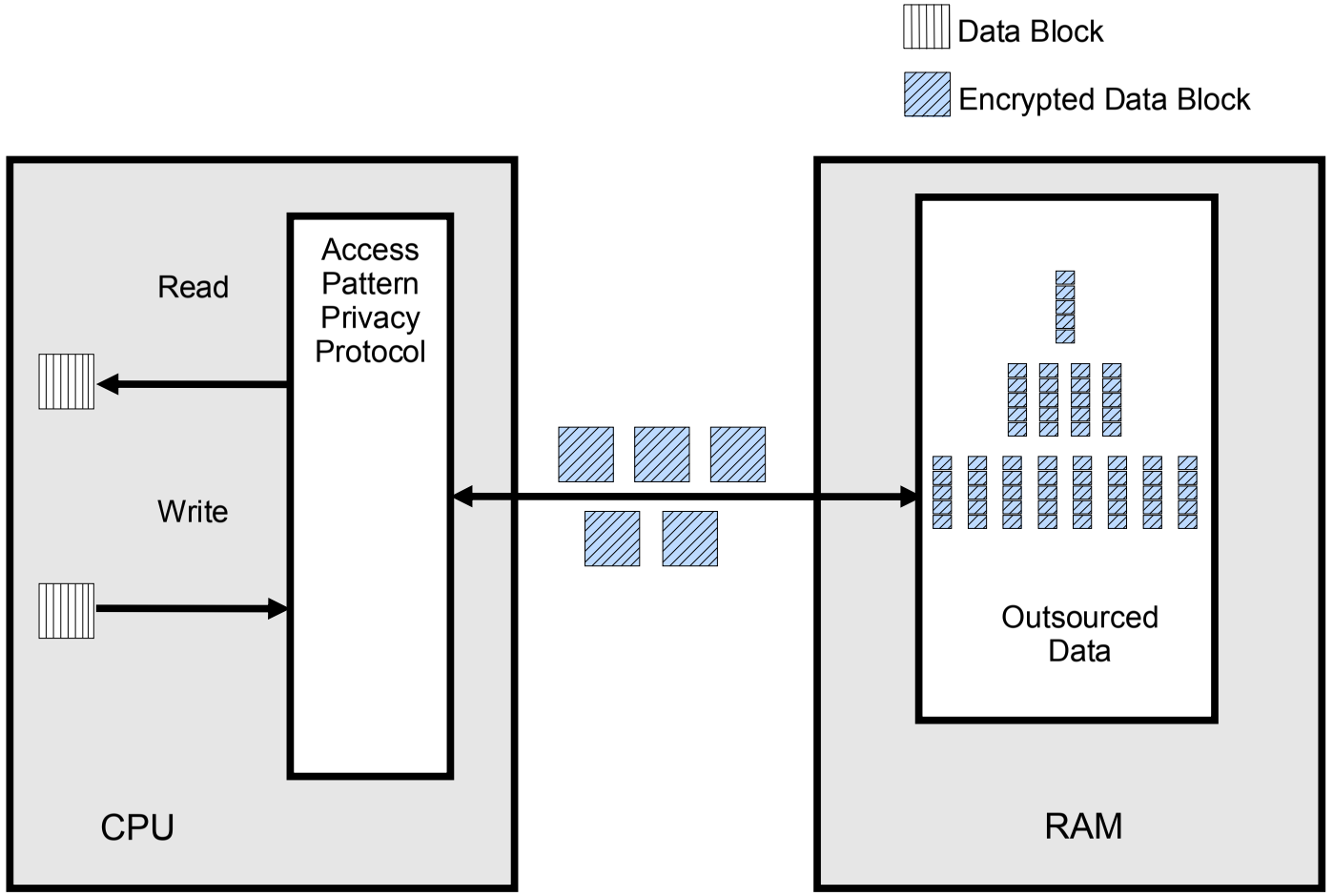
The wizard predicts …



(logarithmic)

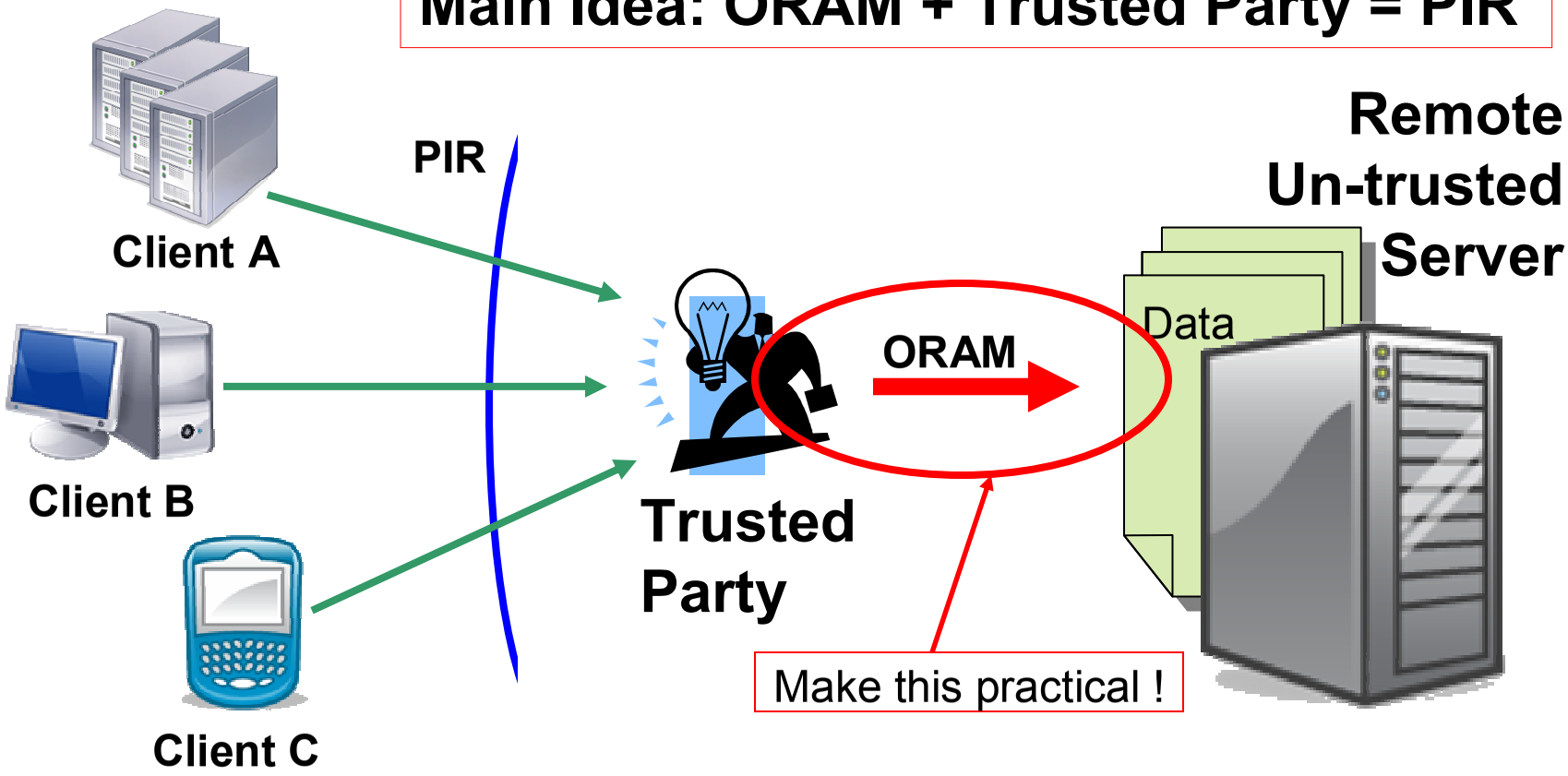
Sion & Carbunar @ NDSS 2007

“Oblivious RAM”



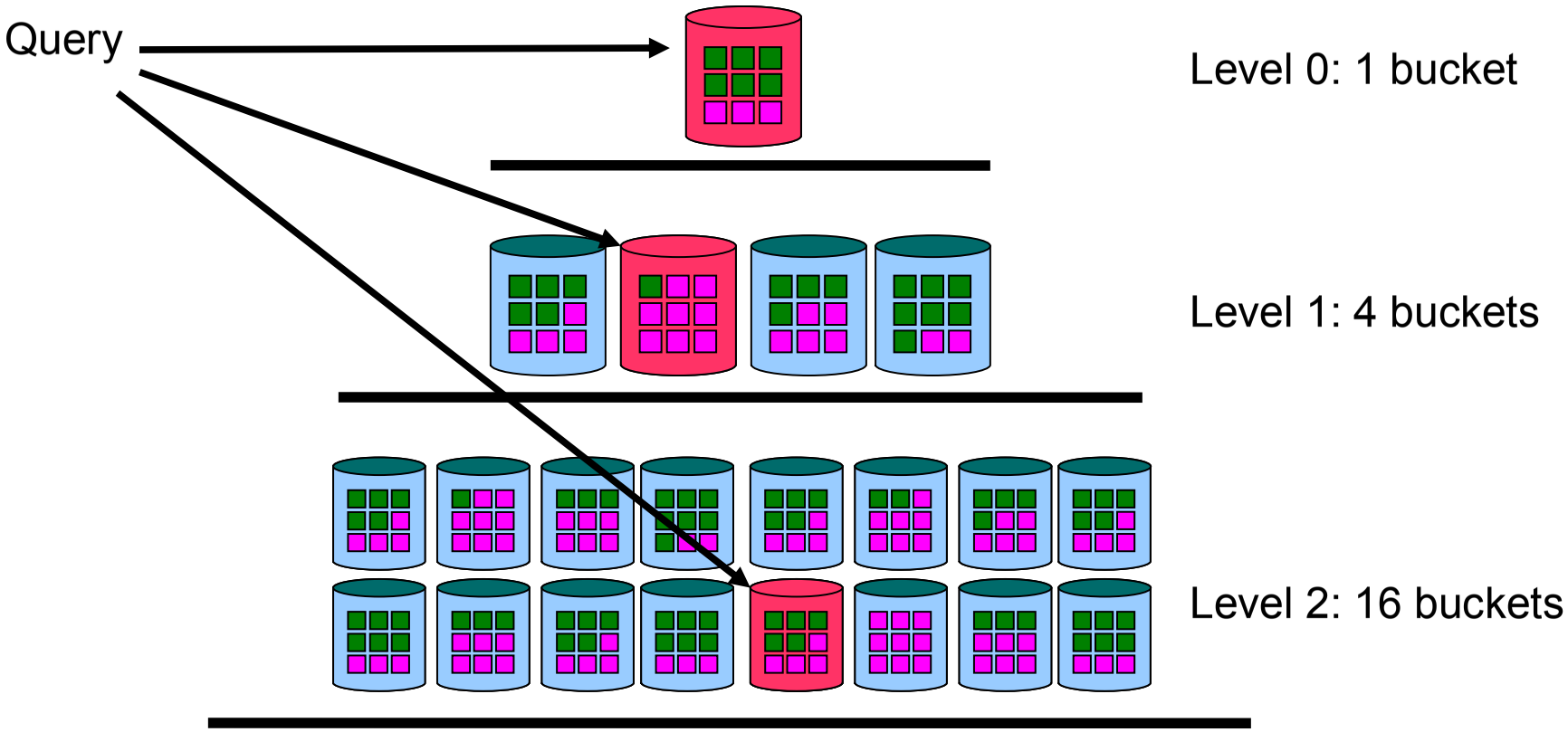
Can we use ORAM ?

Main Idea: ORAM + Trusted Party = PIR



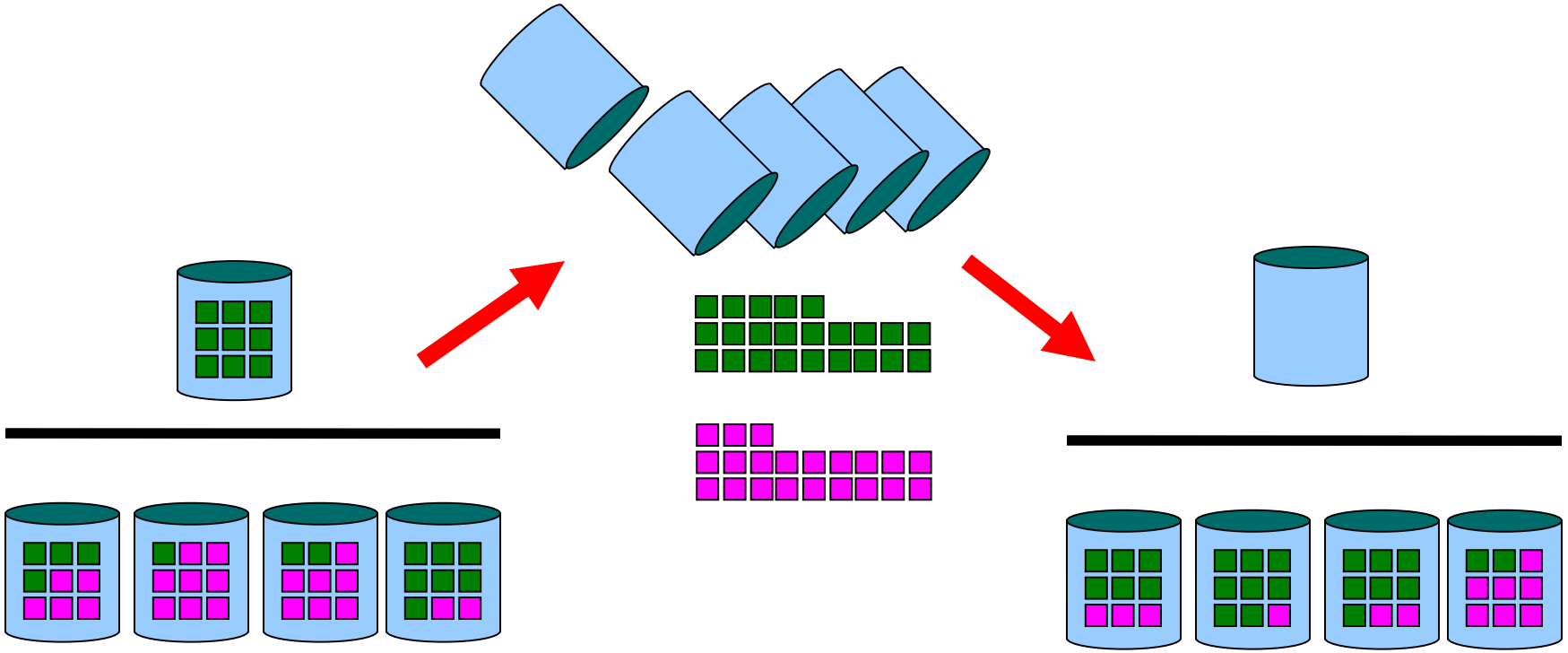
Asonov, Smith and others

ORAM Overview



ORAM: Ostrovsky, 1996

ORAM: Level full ?

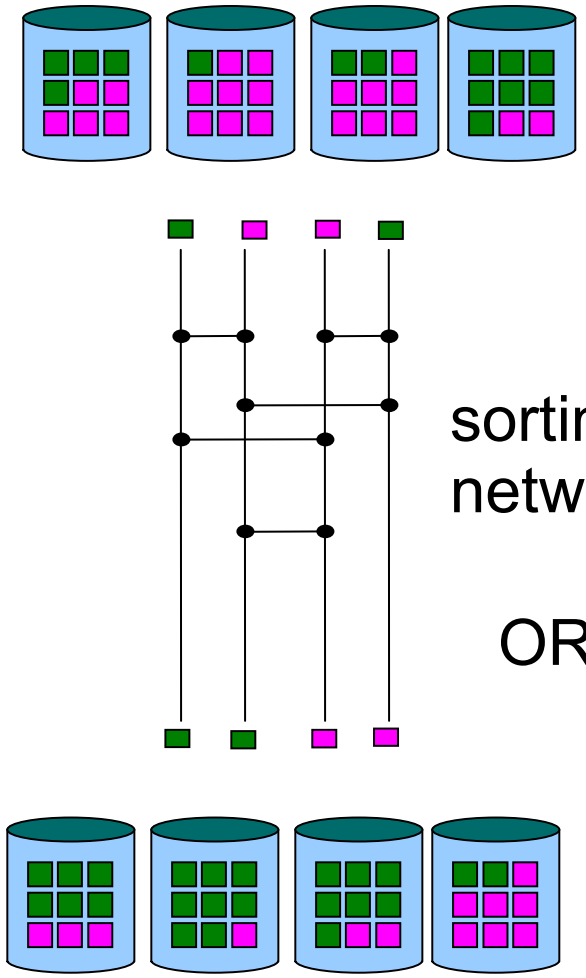


Data Item

Fake “bucket filler”

ORAM: Ostrovsky, 1996

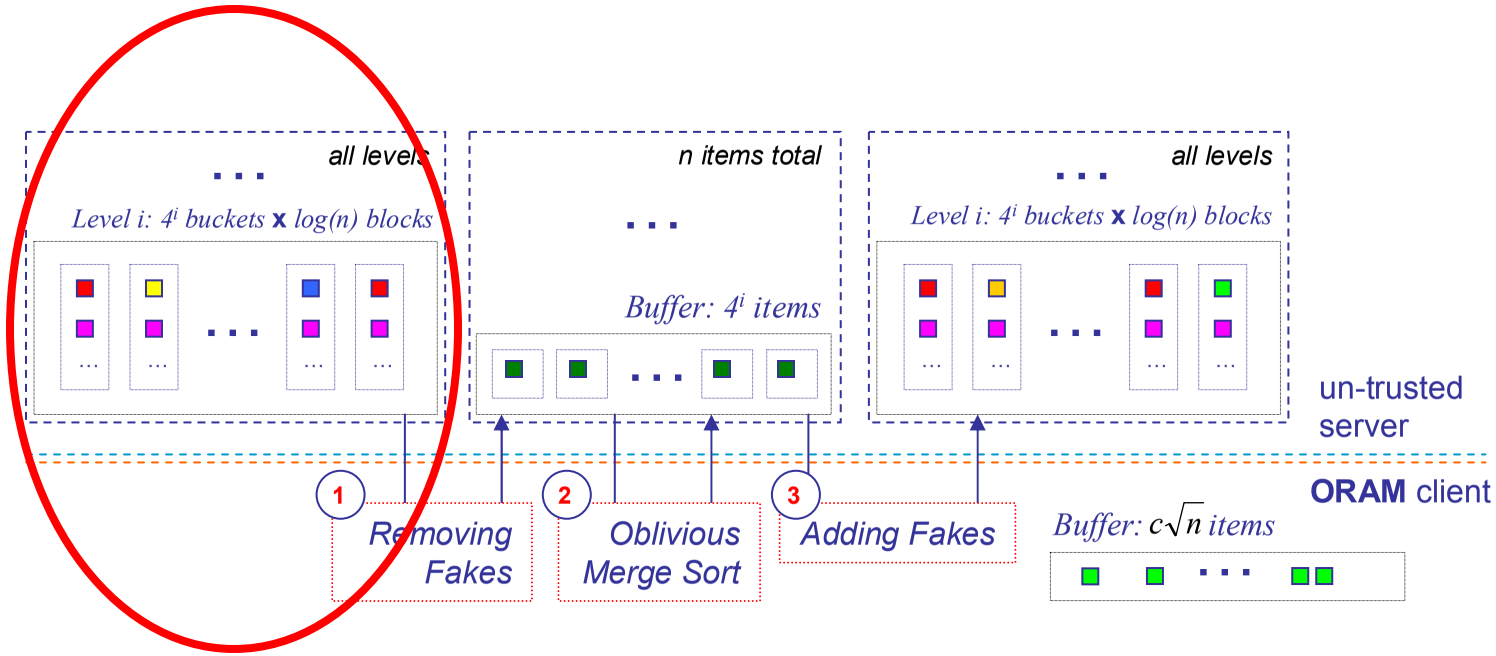
ORAM: How to re-shuffle ?



sorting network

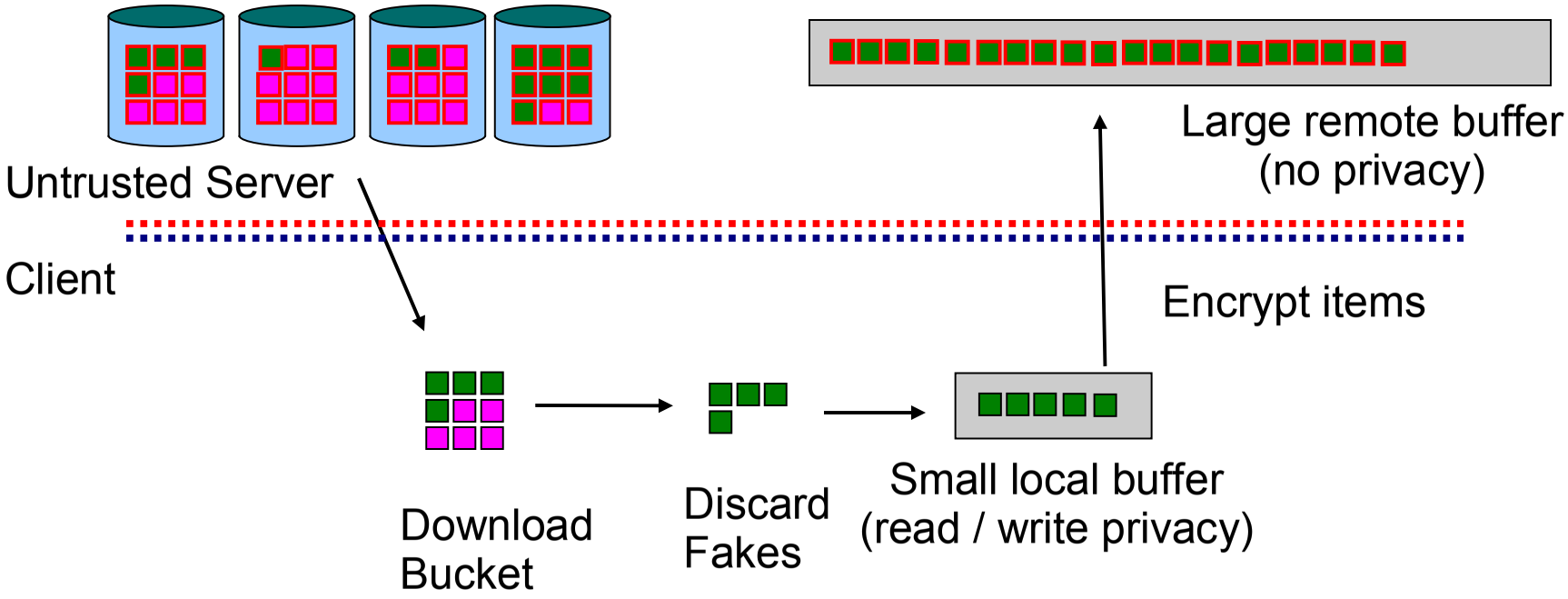
ORAM: Ostrovsky, 1996

Re-shuffle: faster ?

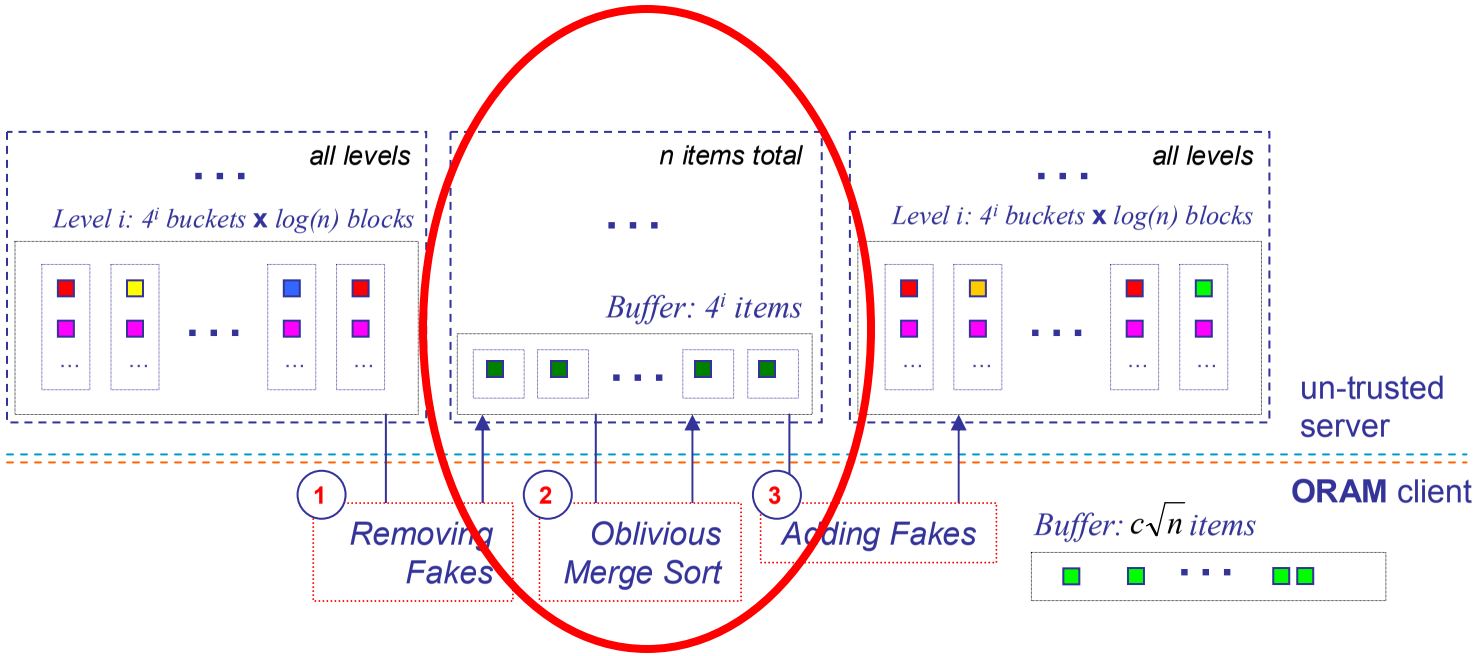


Remove fakes *obliviously*

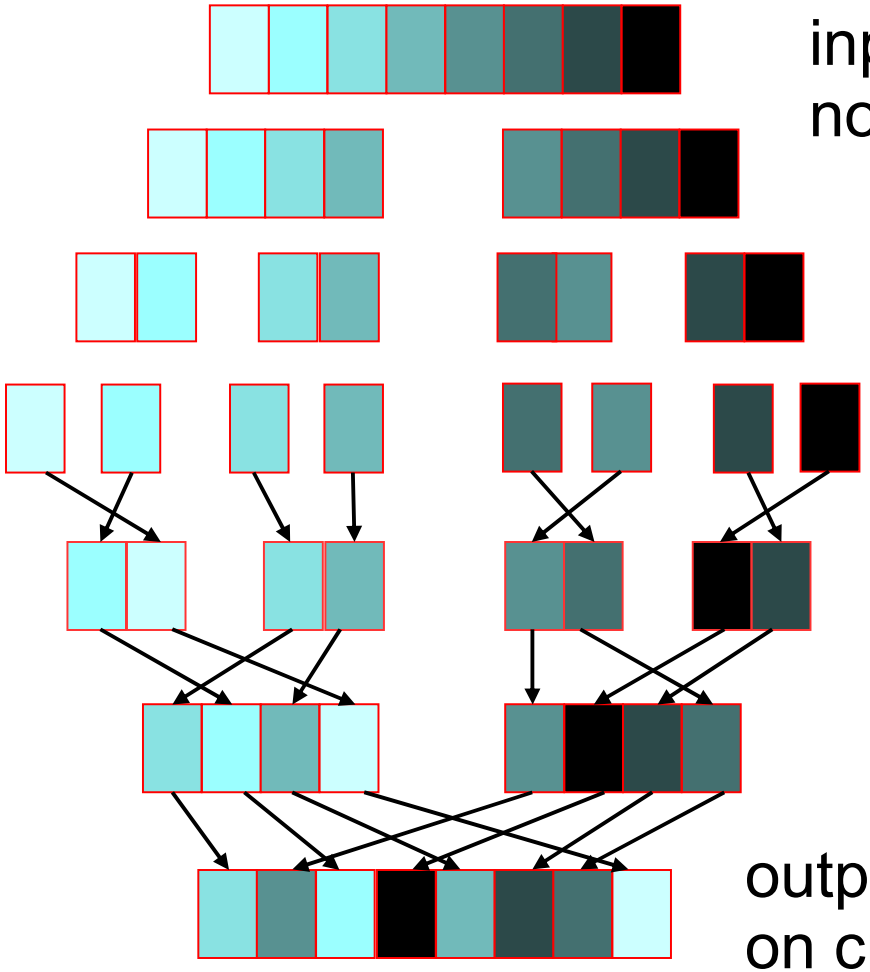
- Discard fakes without revealing their locations
- But: ***how big of a buffer do we need ?***



Re-shuffle: merge sort



Merge sort on random keys

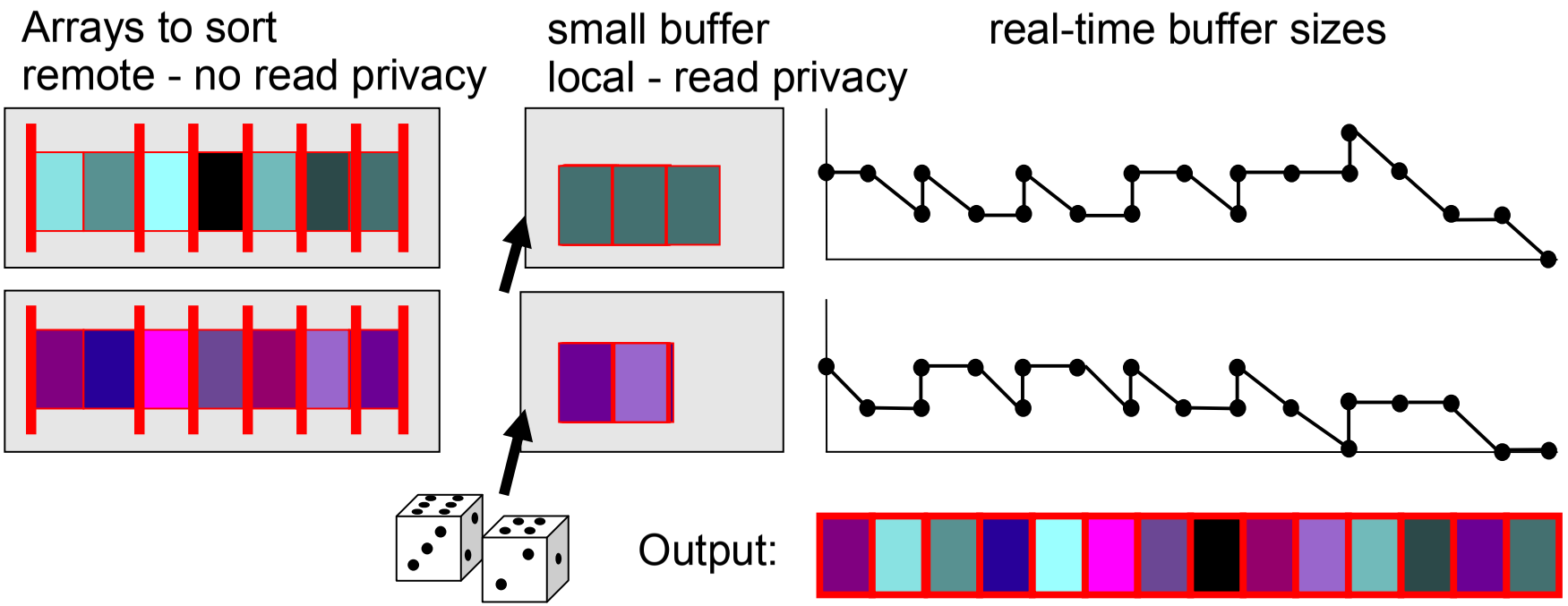


input: items,
no more fakes

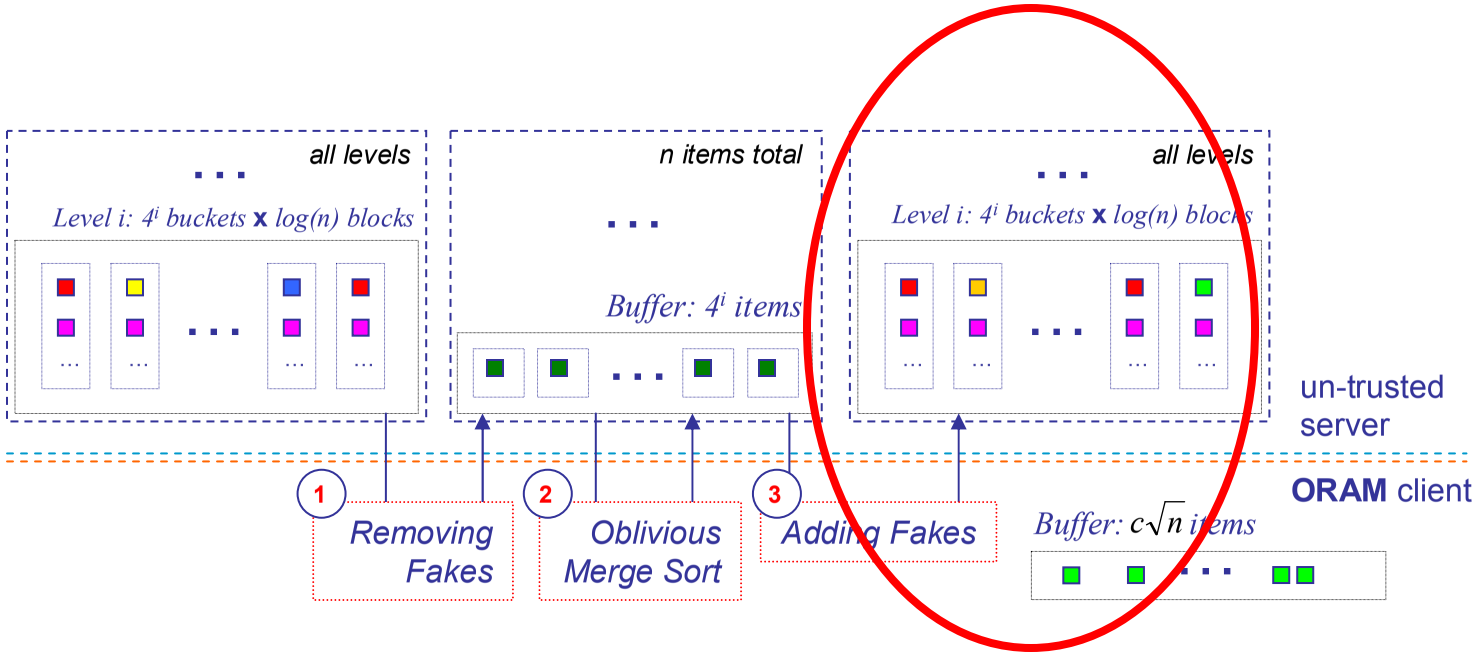
output: sorted
on crypto hash

Sort *obliviously*

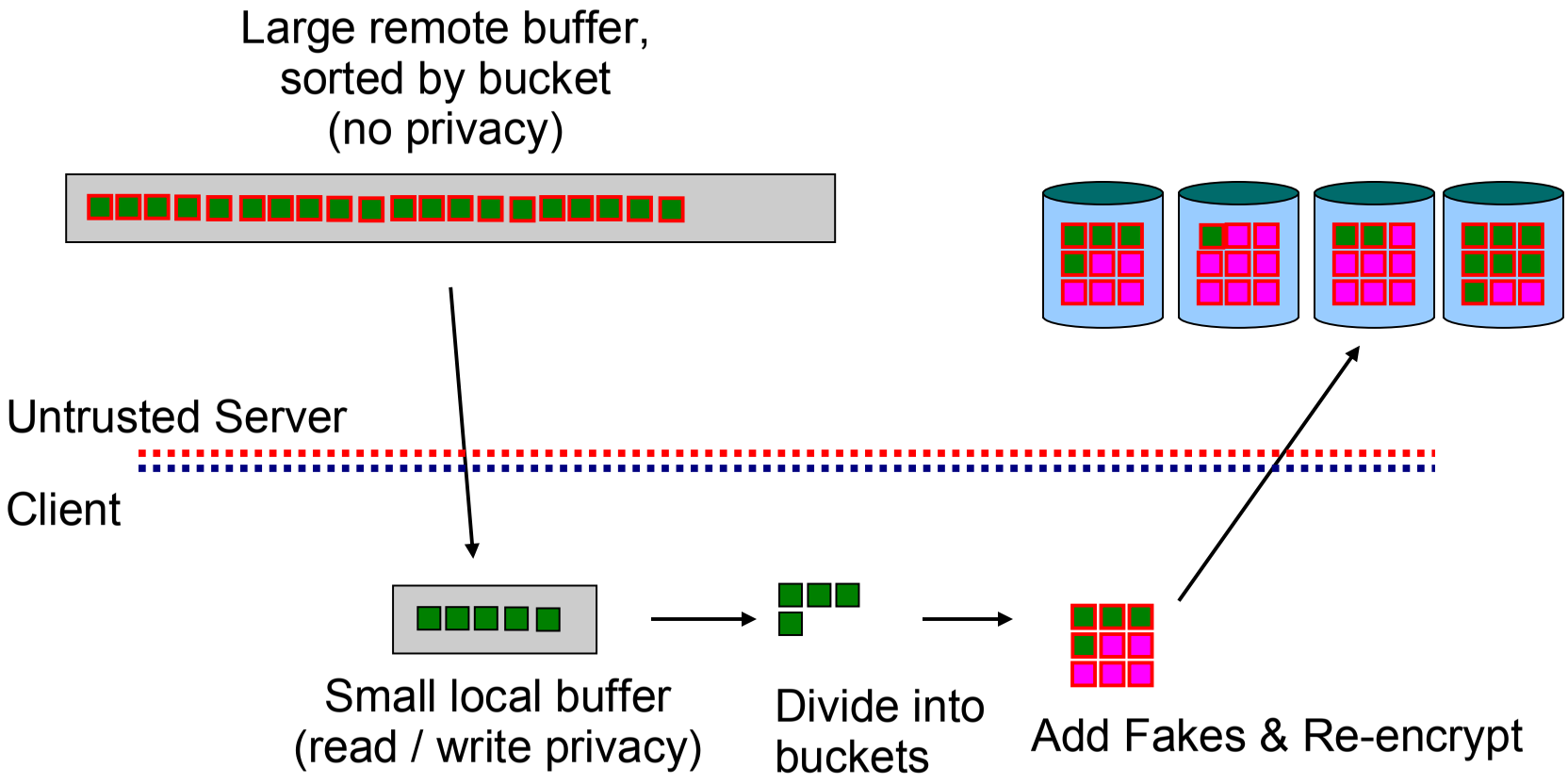
- Idea: Buffer reads to hide the permutation
- Key: Cursors remain close, since keys random



Re-shuffle: add new fakes



Add new fakes *obliviously*



Costs

- Database size n consists of $\log(n)$ levels
- Level i is reshuffled once every 4^i accesses
- Reshuffle of i costs $O\left(4^i \log 4^i\right)$
- Amortized cost per query for reshuffling:

$$\sum_{i=1}^{\log(n)} \frac{O\left(4^i \log 4^i\right)}{4^i} = \sum_{i=1}^{\log(n)} O(i) = O\left(\log^2(n)\right)$$

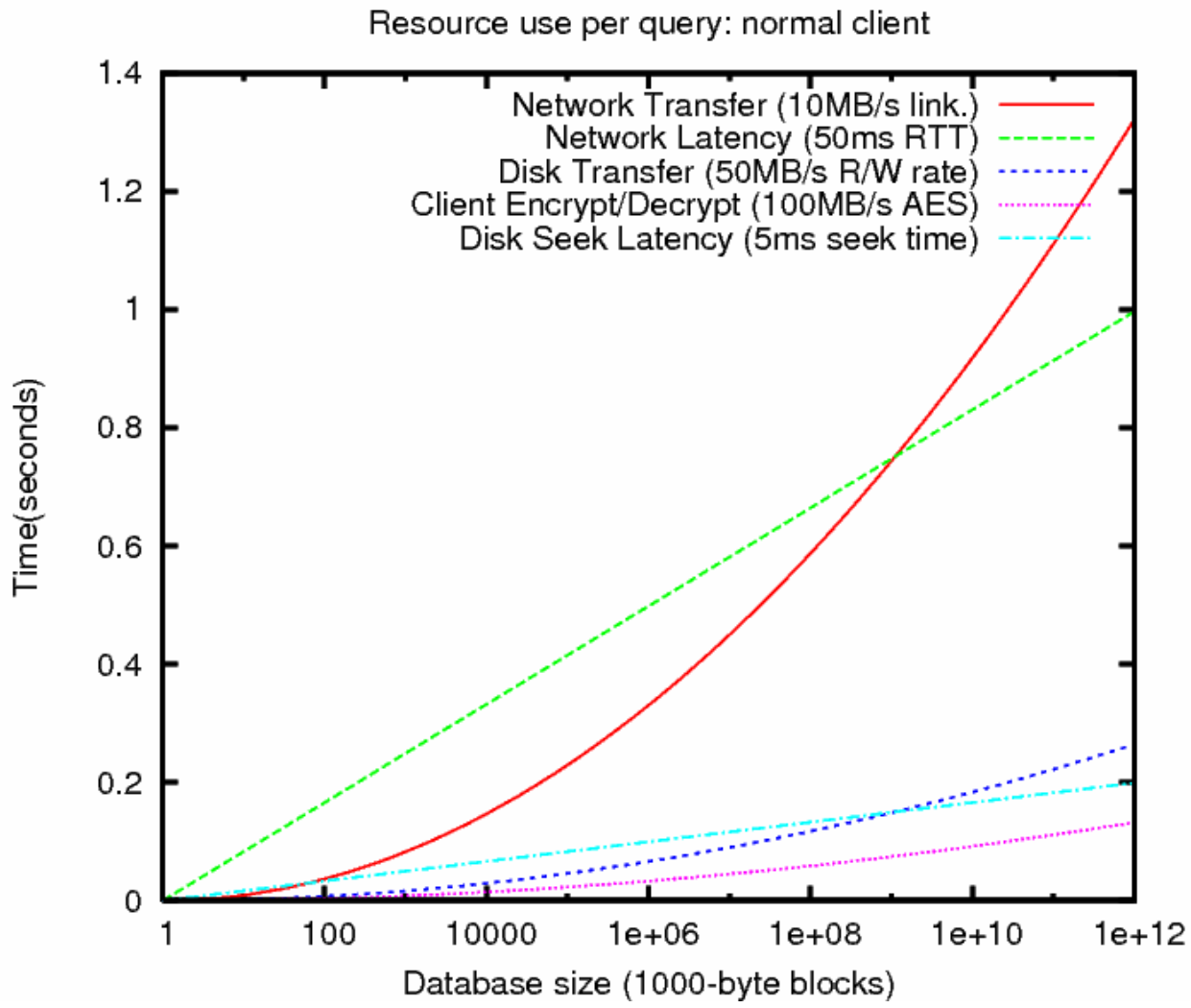
- Online cost per query: $\log(n)$ levels x $O(\log(n))$ bucket size = $O\left(\log^2(n)\right)$

Existing work

For client storage $O(\sqrt{n})$

- Goldreich/Ostrovsky ORAM 1996 - $O(\log^4 n)$
- Smith/Illiev 2004 - $O(\sqrt{n}\log n)$
- Wang et al. ESORICS 2006 - $O(\sqrt{n})$
- This protocol - $O(\log^2 n)$

How fast can we run ?



Conclusions

Practical Private Information Retrieval Protocol
Several queries per second over large data sets
Full computational privacy

Future Work

De-amortize re-shuffle costs
Reduce server storage overhead - eliminate use of fakes
New mechanism with $O(\log n \log\log n)$ overhead


Thank you!