NSAC

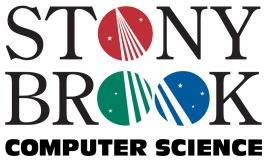
Network Security and Applied Cryptography Laboratory

http://crypto.cs.stonybrook.edu

# On the Practicality of PIR


Radu Sion
Stony Brook NSAC Lab
sion@cs.stonybrook.edu

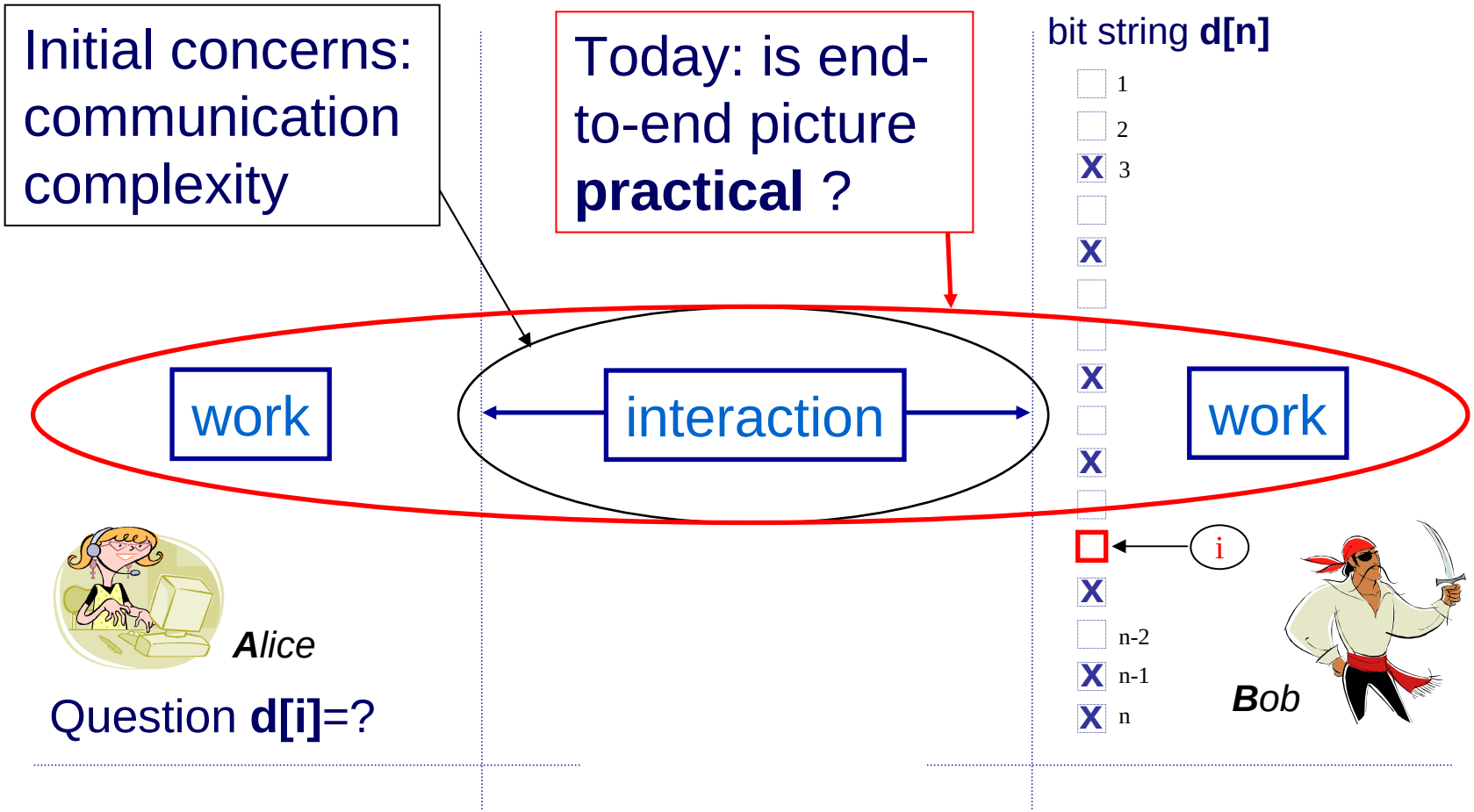
Carbunar Bogdan
Motorola Labs
carbunar@motorola.com


ver. 2.1 (2/25/2007)

# What is PIR ?




Initial concerns: communication complexity
Today: is end-to-end picture **practical** ?
bit string **d[n]**
1
2
3
n-2
n-1
n
i
work
interaction
work
*Alice*
*Bob*
Question **d[i]**=?

# What is “practical” ?



Many things.

But often, in real life, these are defined by execution **time**.

Baseline: a cheaper PIR protocol than trivial database transfer (for now !).

What is cheaper ?

Often, ”not slower”.

**Faster**. **Not always !**

# Time to “illustrate” …



**… we choose:** E. Kushilevitz and R. Ostrovsky, “Replication is not needed: single database, computationally-private information retrieval”, FOCS 1997.

**Why ?**

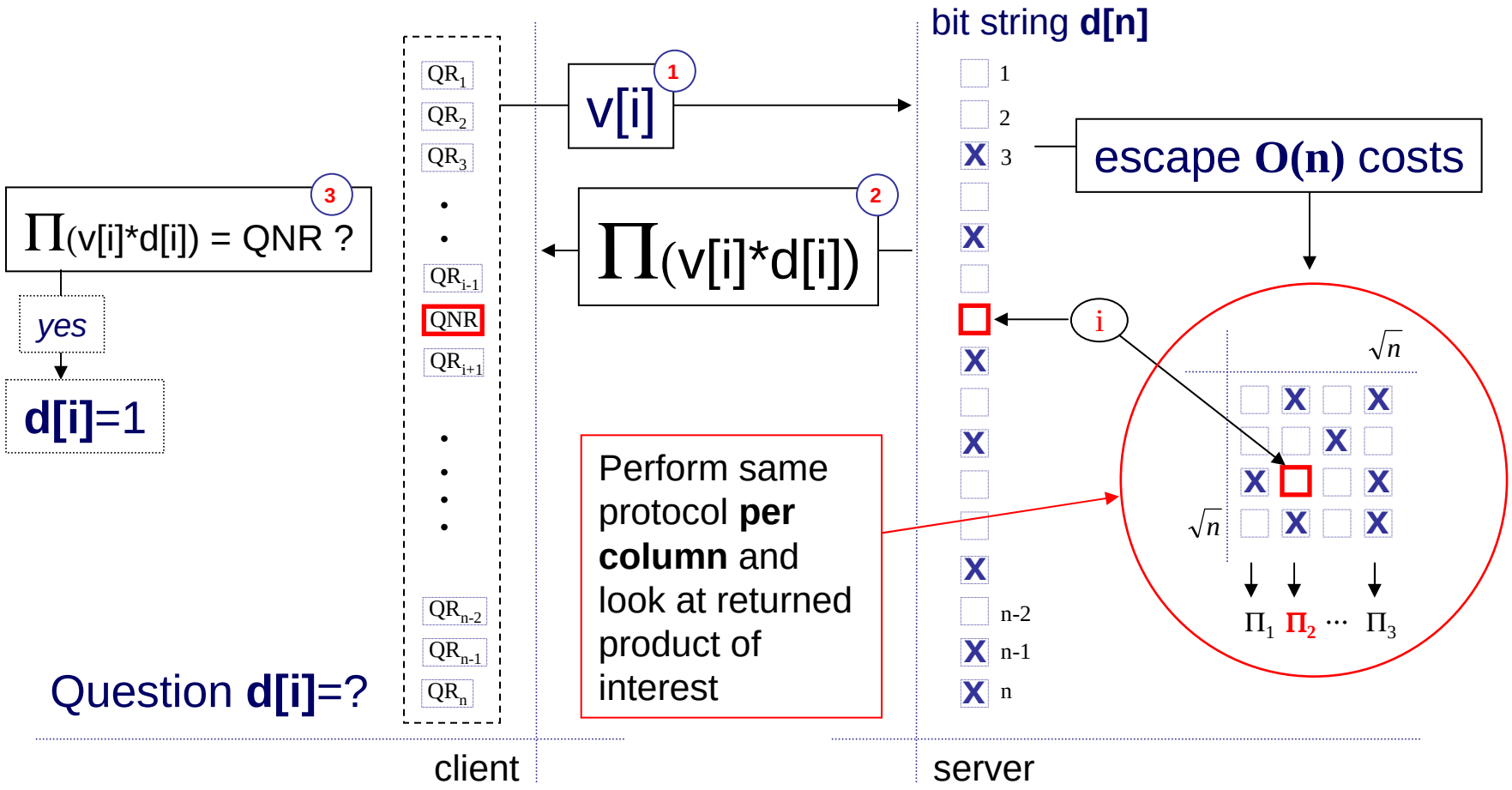
It is the least *computationally* expensive and arguably the *fastest* of the bunch.

The results can be applied to all 7+ single-server computational protocols we looked at (based on well-established intractability assumptions)

They also apply to any protocol with a per-bit cost > fraction (e.g., 1/10) of the cost of a modular multiplication.

# Protocol overview





# Execution time analysis



$$T_{pir} = nt_{mul}(|N|) + 2\sqrt{n}(|N|)t_t + \sqrt{n}t_{qrv}(|N|)$$

PIR-favorable simplification: we ignore anything else but the server-side modular multiplication costs.

$$T_{pir} \approx n \times t_{mul}(|N|)$$

**Conclusion:** PIR is “practical” iff. per-bit server-side complexity is faster than bit transfer.

# RSA Key Size Schedules



| *target* | 1995 | 2000 – 2010 | 2011 – 2030 | 2030– |
|---|---|---|---|---|
| *bits* | 768 | 1024-1536 | 2048 | 3072 |

**Recommended RSA key sizes.**

# Past: MIPS Schedule



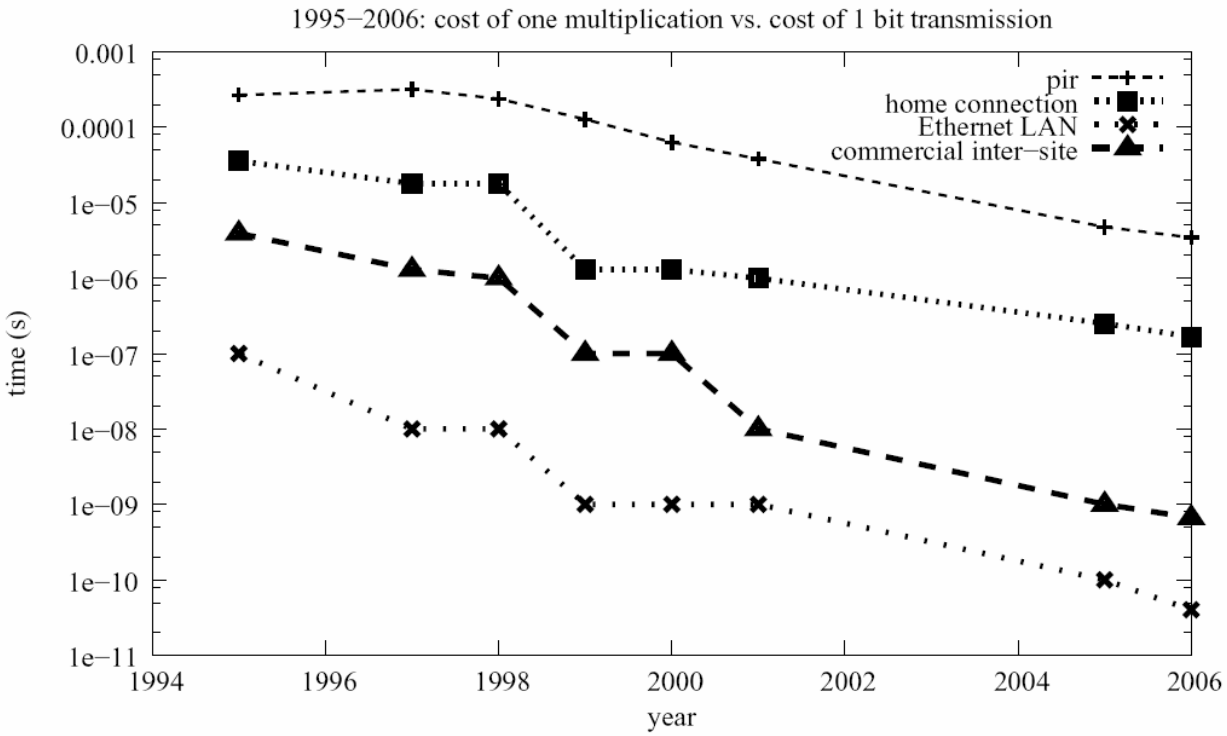
| year | $M$ | $B$ | $B_2$ | $B_3$ |
|---|---|---|---|---|
| 1995 | 200 | 0.028 | 10 | 0.256 |
| 1997 | 300 | 0.056 | 100 | 0.768 |
| 1998 | 400 | | | 1.000 |
| 1999 | 744 | 0.768 | 1000 | 10 |
| 2000 | 1500 | | | |
| 2001 | 2500 | 1.000 | | 100 |
| 2005 | 15000 | 4.000 | 10000 | 1000 |
| 2006 | 25000 | 6.000 | 10000 | 1500 |

**Estimated average values for x86 CPU MIPS, end-user home commodity Internet ($B$), Ethernet LAN ($B_2$) and commercial high-end inter-site ($B_3$) bandwidth (Mbps), between 1995 and 2006.**

**Note: 15MBps/5MBps costs $29.95/mo.**

## Past: 1 bit multiplication vs. transfer






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

# Present: Hardware




**Illustrative baseline.** Results hold within orders of magnitude (e.g., if chip would be ten times faster). Wide spread. Fast ALUs. Setup: 3.6GHz, 1GB RAM. 11000 MIPS (Intel).

## Present: Benchmarks



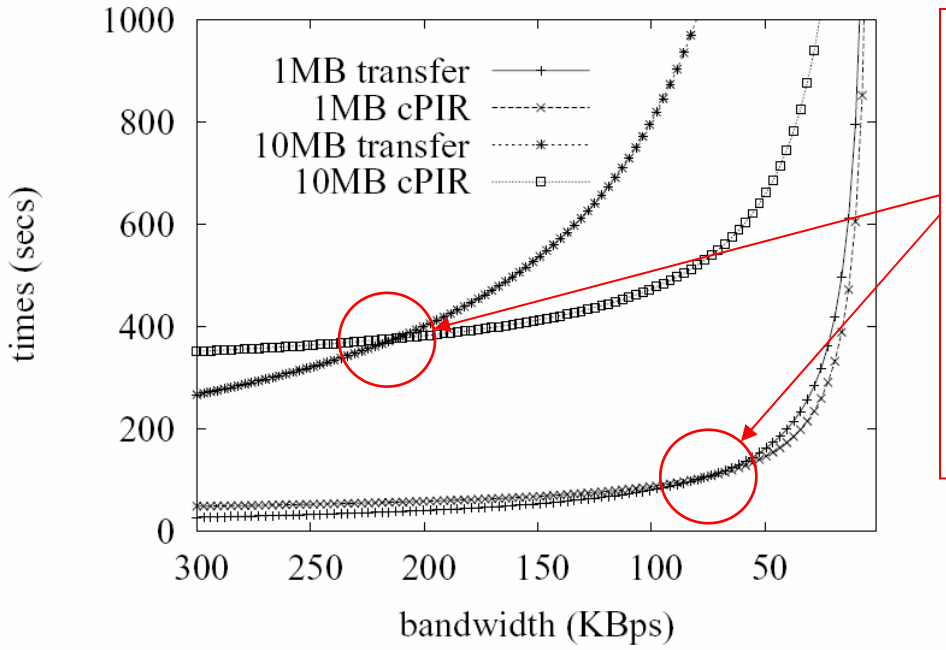
1024 bit values: **273,000 mod. mul. / sec.**
PIR-processing one bit: >>**3700 ns**
10MBps transfer: ~**100-120 ns**

Trivial transfer is **35+ times faster** than PIR.

**Question:** But what about faster hardware ?

## Present: Low bandwidth conditions





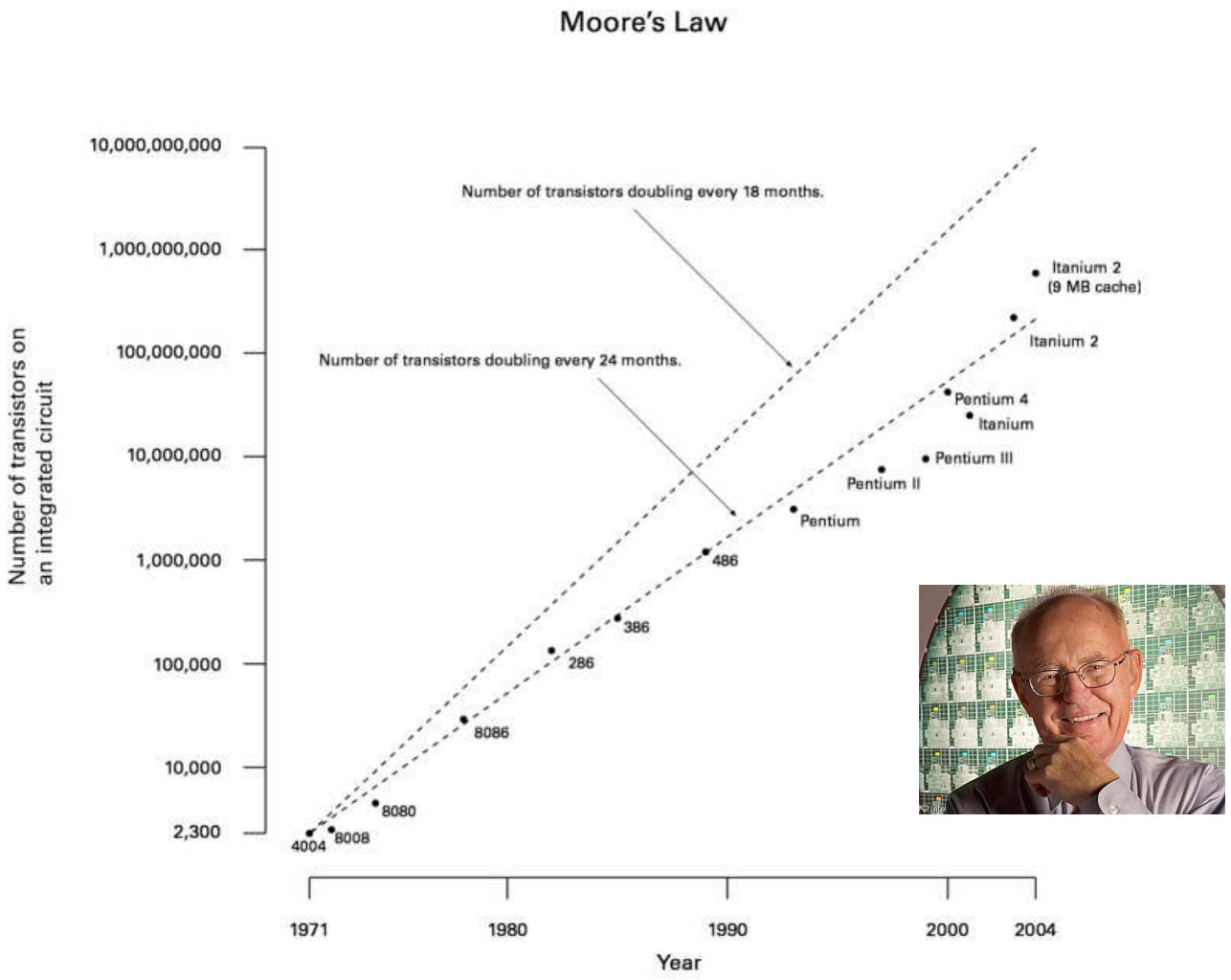

**Conclusion:** Today's PIR protocols become usable for low (tens of KBps) bandwidths.

**But**: … at additional computation costs ☺

Low Bandwidth ($t_{mul} < t_t$, condition (4) does not hold): behavior of execution times for cPIR vs. database transfer times. If its (previously ignored) communication overheads are considered, the bandwidth thresholds below which cPIR becomes useful further decrease.

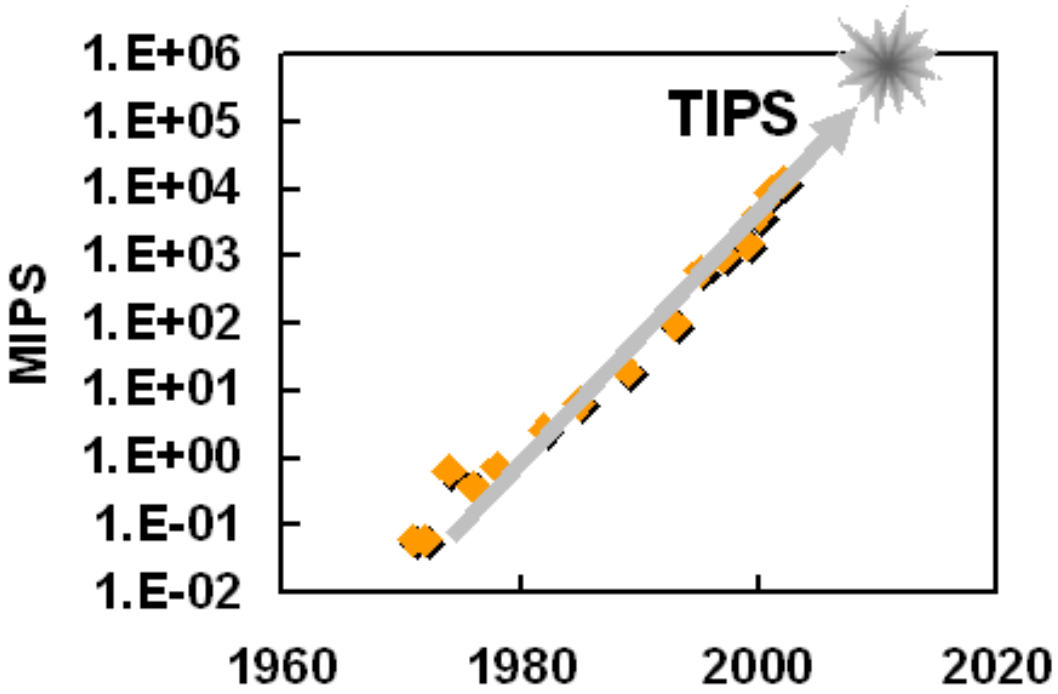
# Future: Moore Says ☺ …





# Future: CPU Speed follows Moore !

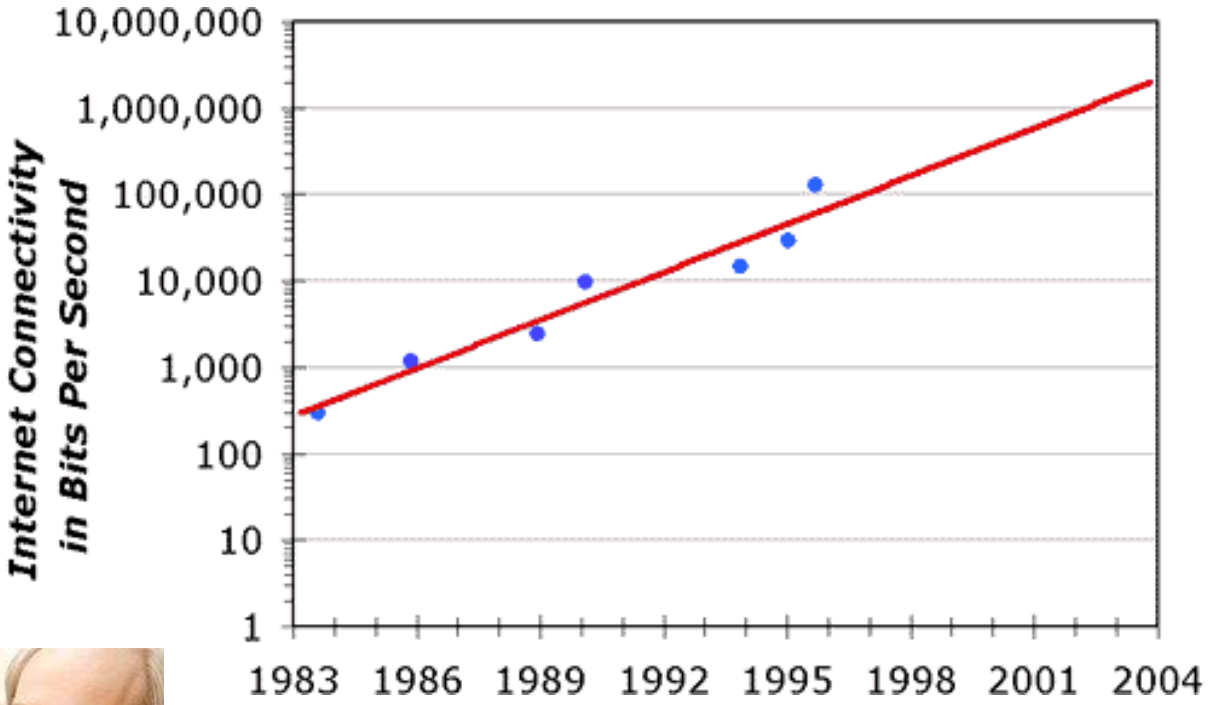





**Source: "Gigascale Integration-Challenges and Opportunities", Shekhar Borkar, Director, Circuit Research, Intel Corp.**

# Network Speed: Nielsen's Law



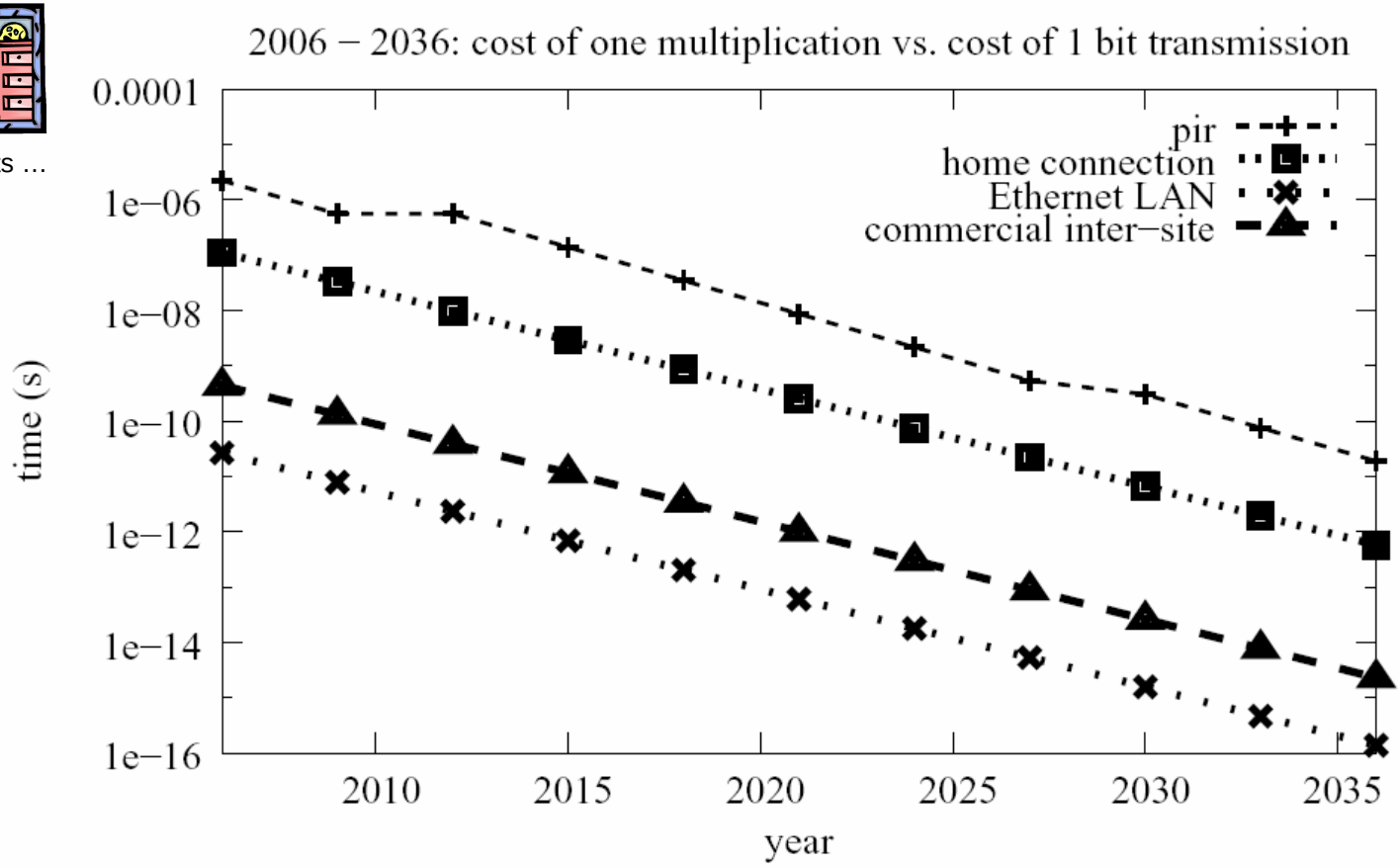



*"high end connection speed grows 50% per year"*

## Future: 1 bit multiplication vs. transfer



The wizard predicts …




(logarithmic)

# What about sPIR ?



*B's input is* $X_1, X_2, \ldots, X_N$, *where each* $X_I \in \{0,1\}^m$ *and* $N = 2^\ell$. *The receiver A would like to learn* $X_I$;

1. *B prepares* $\ell$ *random pairs of keys*

$$(K_1^0, K_1^1), (K_2^0, K_2^1), \ldots (K_\ell^0, K_\ell^1)$$

*where for all* $1 \leq j \leq \ell$ *and* $b \in \{0,1\}$ *each* $K_j^b$ *is a t-bit key to the pseudo-random function* $F_K$.

*For all* $1 \leq I \leq N$ *let* $\langle i_1, i_2, \ldots i_\ell \rangle$ *be the bits of* $I$. *B* $\bigoplus_{j=1}^{\ell} F_{K_j^{i_j}}(I)$.

2. *A and B engage in a 1-out-of-2 OT for each* $1 \leq j \leq \ell$ *on the strings* $(K_j^0, K_j^1)$. *If A would like to learn* $X_I$ *she should pick* $K_j^{i_j}$.

3. *B sends A the strings* $Y_1, Y_2, \ldots Y_N$.

4. *A reconstructs* $X_I = Y_I \oplus \bigoplus_{j=1}^{\ell} F_{K_j^{i_j}}(I)$.

Here do PIR instead: "Naor-Pinkas PIR-SPIR reduction"

[56] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 245–254, New York, NY, USA, 1999. ACM Press.

# What do we do ?

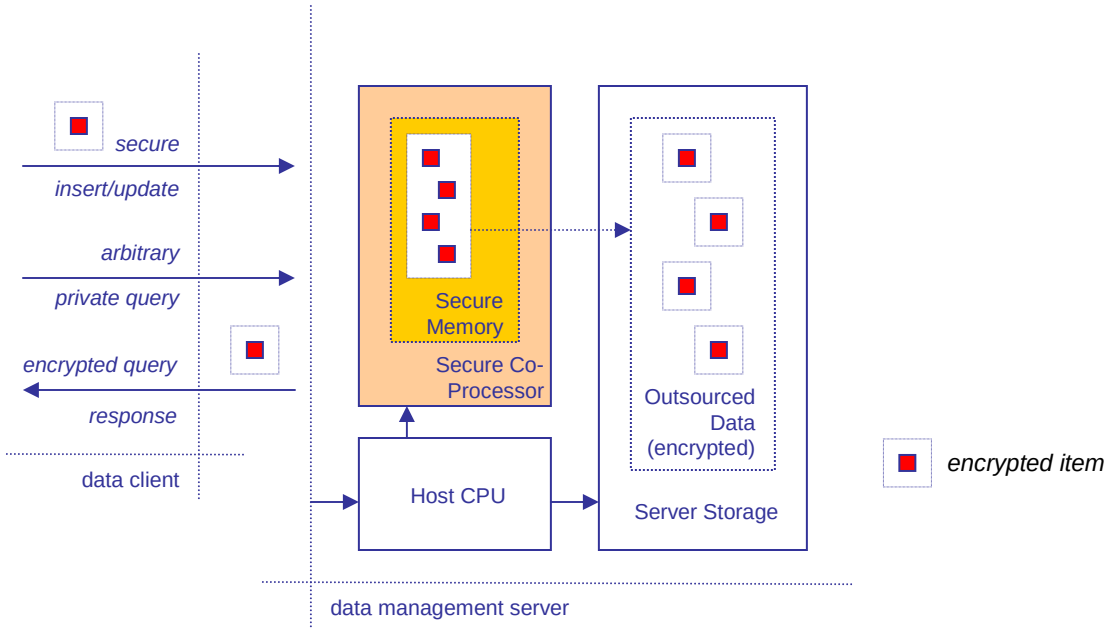


- New PIR protocols
  - Gasarch & Yerukhimovich protocol ?!
  - Gentry & Ramzan ?
- Hardware PIR (Sean Smith @ Darthmouth)
- Weaker privacy metrics (statistical)

- **Important:** use correct baseline for "practical"
  - compare with application requirements, not with trivial transfer (e.g., 4TB database – trivial transfer over 100MBps takes 22+ hrs)

# Trusted Hardware






**A secure co-processor on the data management side may allow for significant leaps in expressivity for queries where privacy and completeness assurance are important.**

# IBM 4764

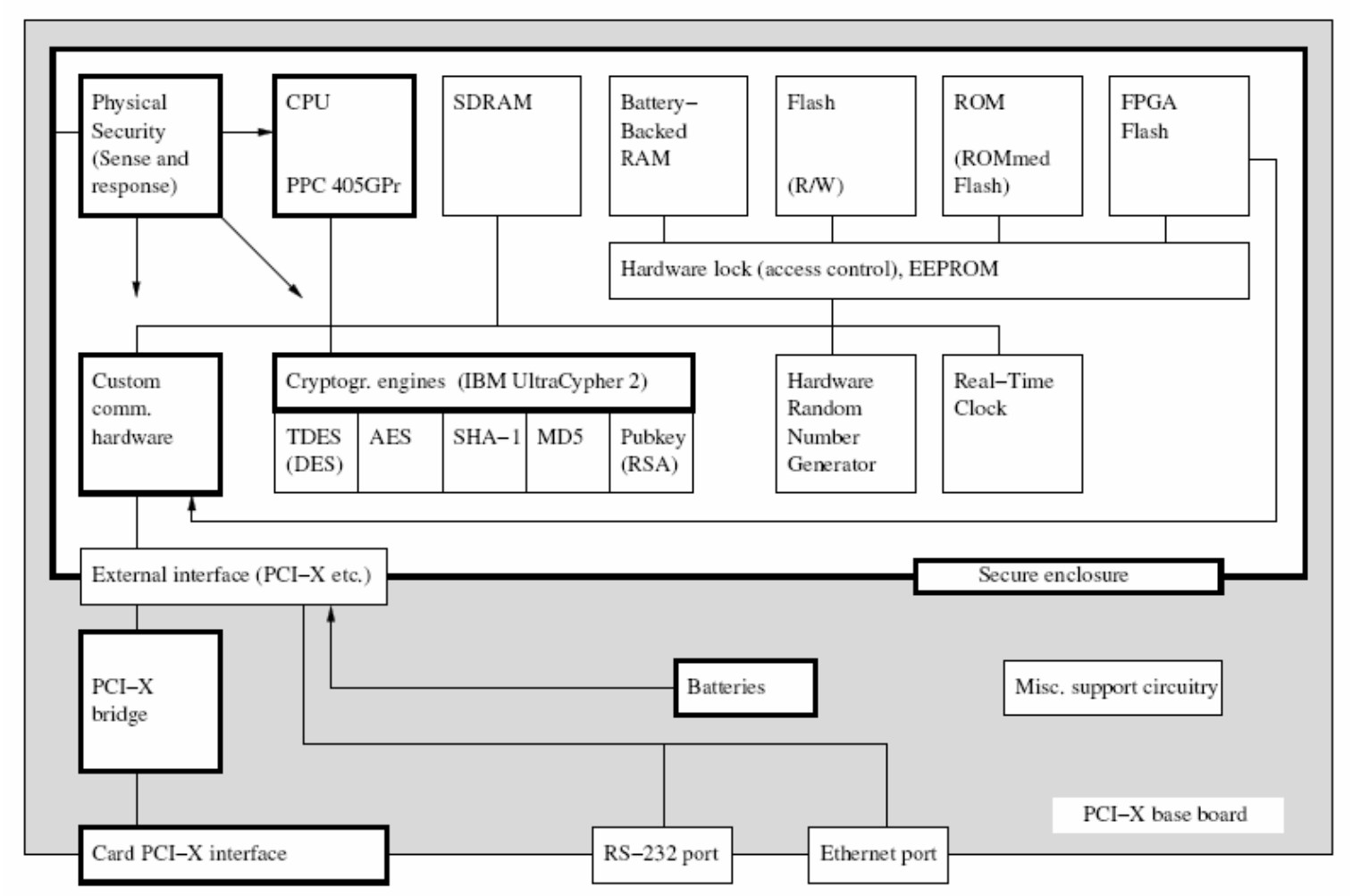


RSA1024 Sign: **848/sec**
RSA1024 Verify: **1157/sec**
3DES: **1-8MB/sec**
DES: **1-8MB/sec**
SHA1: **1-21MB/sec**

IBM 4764-001: 266MHz PowerPC. 64KB battery-backed SRAM storage. Crypto hardware engines: AES256, DES, TDES, DSS, SHA-1, MD5, RSA. FIPS 140-2 Level 4 certified.

# IBM 4764 Architecture





# Comparison: Pentium 4



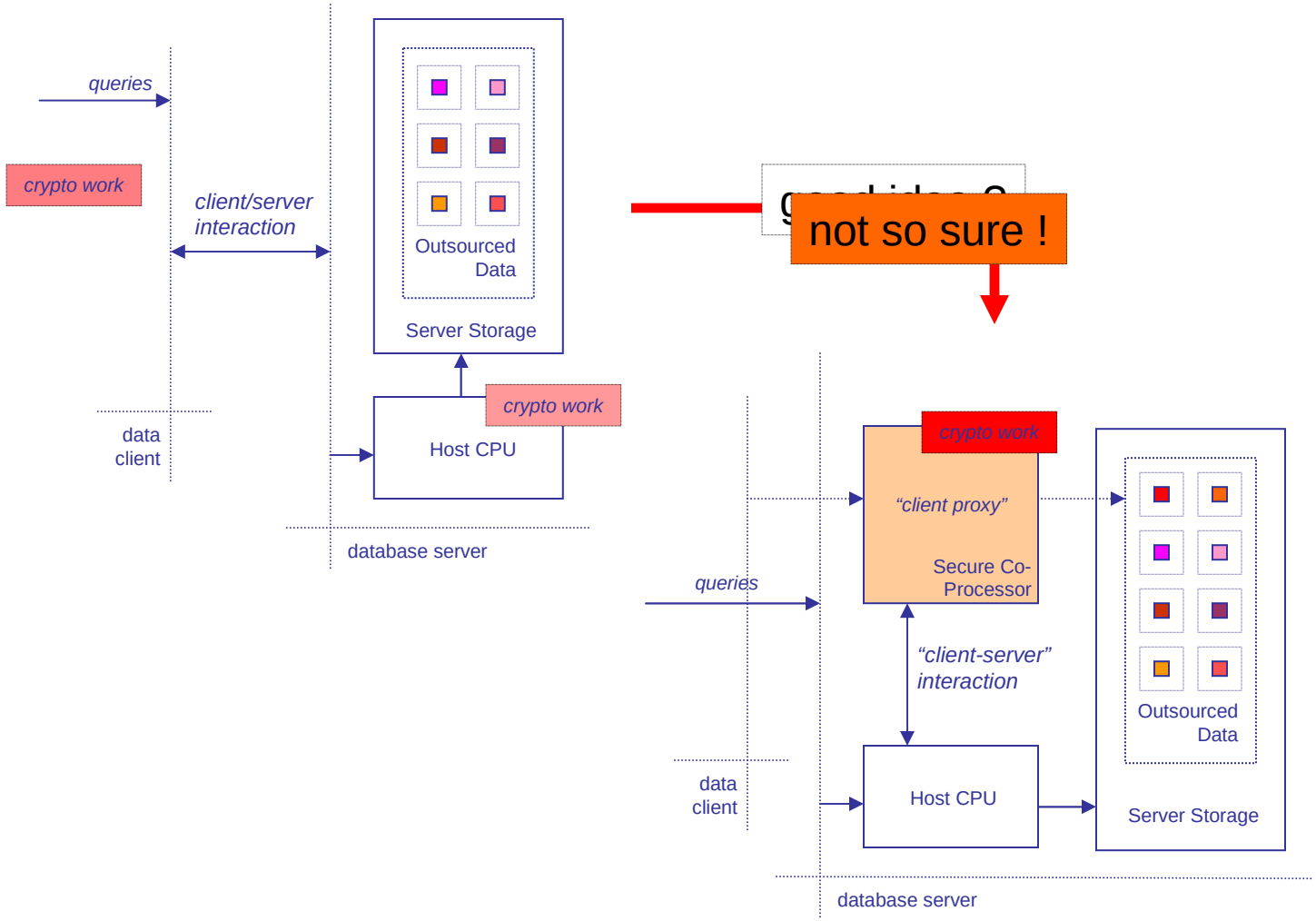

**Illustrative baseline.**
Pentium 4. 3.4GHz.
1GB RAM. 11000 MIPS.
OpenSSL 0.9.7f

DES/CBC: **70MB/sec**
RC4: **138MB/sec**
MD5: **18-615MB/sec**
SHA1: **18-340MB/sec**

Modular MUL 1024: **273000/sec**
RSA1024 Sign: **261/sec**
RSA1024 Verify: **5324/sec**
3DES: **26MB/sec**

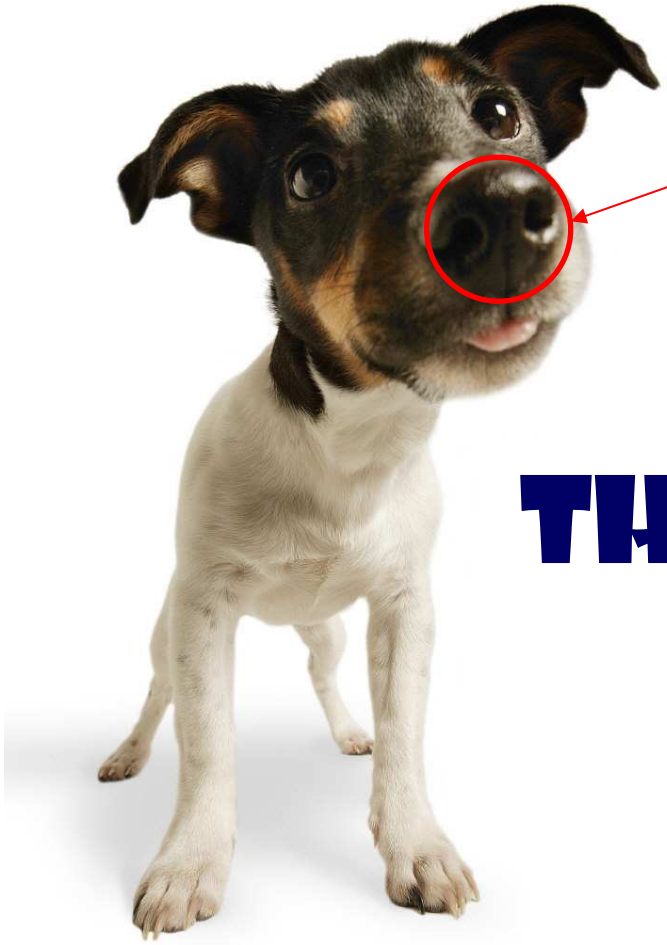
# Sample DON'T




queries
crypto work
client/server interaction
Outsourced Data
Server Storage
crypto work
Host CPU
data client
database server
not so sure !
crypto work
"client proxy"
Secure Co-Processor
queries
"client-server" interaction
Outsourced Data
data client
Host CPU
Server Storage
database server

# /bin/yes > /dev/lunchtime






THANK YOU !

# Protocol overview



The $n$ bits of the database are organized logically at the server as a bi-dimensional matrix $M$ of size $\sqrt{n} \times \sqrt{n}$. To retrieve bit $M(x, y)$ with computational privacy, the client:

- randomly chooses two prime numbers $p$ and $q$ of similar bit length, computes their product, $N = pq$ and sends it to the server.
- generates $\sqrt{n}$ numbers $s_1, s_2, \ldots, s_{\sqrt{n}}$, such that $s_x$ is a quadratic non-residue (QNR) and the rest are quadratic residues (QR) in $\mathbb{Z}_N^*$.
- sends $s_1, s_2, \ldots, s_{\sqrt{n}}$ to the server.

For each "column" $j \in (1, \sqrt{n})$ in the $\sqrt{n} \times \sqrt{n}$ matrix, the server:

- computes the product $r_j = \prod_{0<i<\sqrt{n}} q_{ij}$ where $q_{ij} = s_i^2$ if $M(i, j) = 1$ and $q_{ij} = s_i$ otherwise [2].
- sends $r_1, \ldots, r_{\sqrt{n}}$ to the client

The client then simply checks if $r_y$ is a QR in $\mathbb{Z}_N^*$ which implies $M(x, y) = 1$, else $M(x, y) = 0$.