Query Execution Assurance for Outsourced Databases

Radu Sion
(sion@cs.stonybrook.edu)
http://www.cs.stonybrook.edu/~sion

Secure Systems Lab

Computer Sciences
Stony Brook University

outline

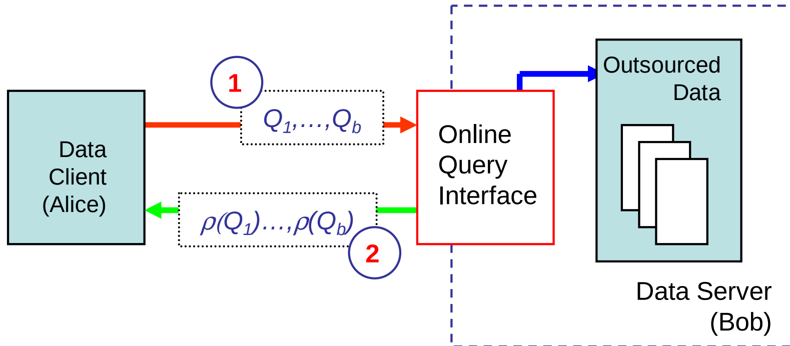
→ ⊕ data outsourcing

⊗ query completeness

⊗ searching

⊗ secure co-processor

data outsourcing



client

- PDA
- personal email user
- file-client

server

- email server
- PostgreSQL
- file-server

data outsourcing: challenges

Untrusted server:

- lazy: incentives to perform less
- curious: incentives to aquire information
- malicious:
  - denial of service
  - incorrect results
  - possibly compromised

Why is this hard ?

- how ?
- arbitrary expressivity
- overheads
  - network
  - computational costs

What do we do ?

- query assurances
- full privacy
  - of queries (even encrypted)
  - of access patterns
- data confidentiality

- ⊗ data outsourcing
- → ⊕ query completeness
- ⊗ searching
- ⊗ secure co-processor

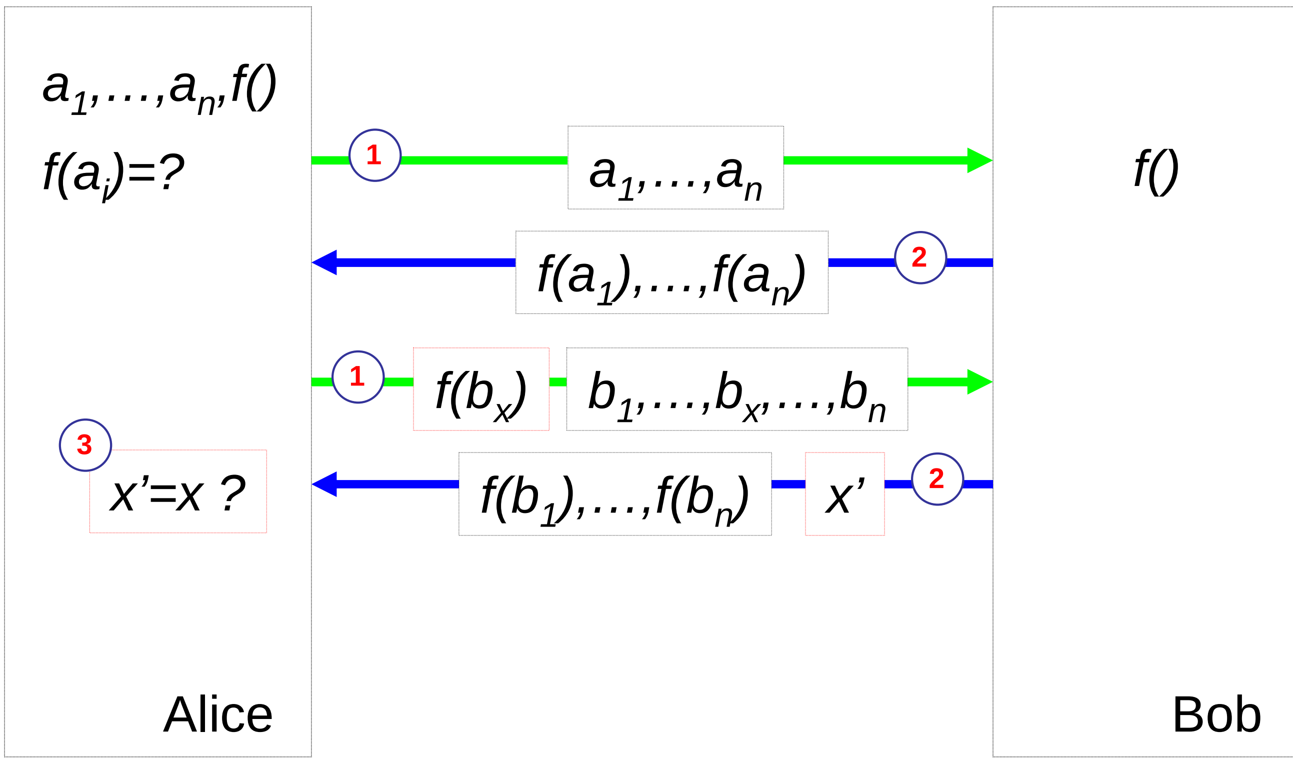
querying with completeness: why ?!

Client requires quantifiable assurances that query results are complete and correct, for arbitrary query types in the presence of a server that could be …

… lazy (we do this *here*)

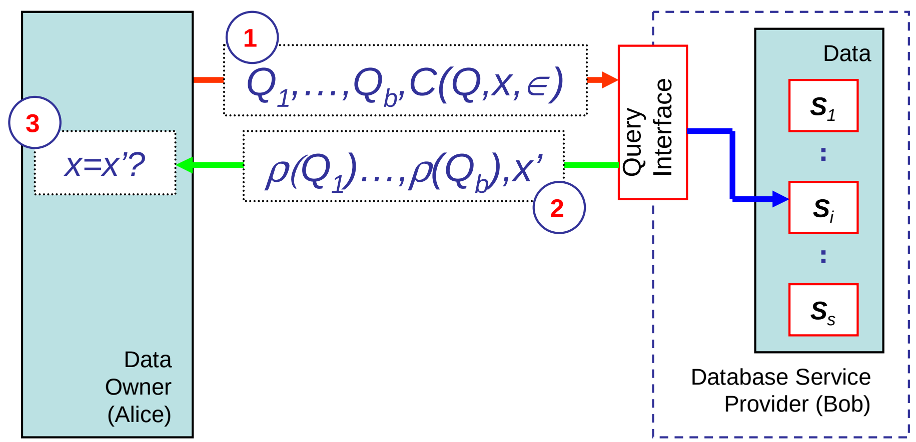
… and/or fully malicious (!)

ringers



P. Golle and I. Mironov, *"Uncheatable Distributed Computations"*, RSA 2001 (Cryptographer's track)

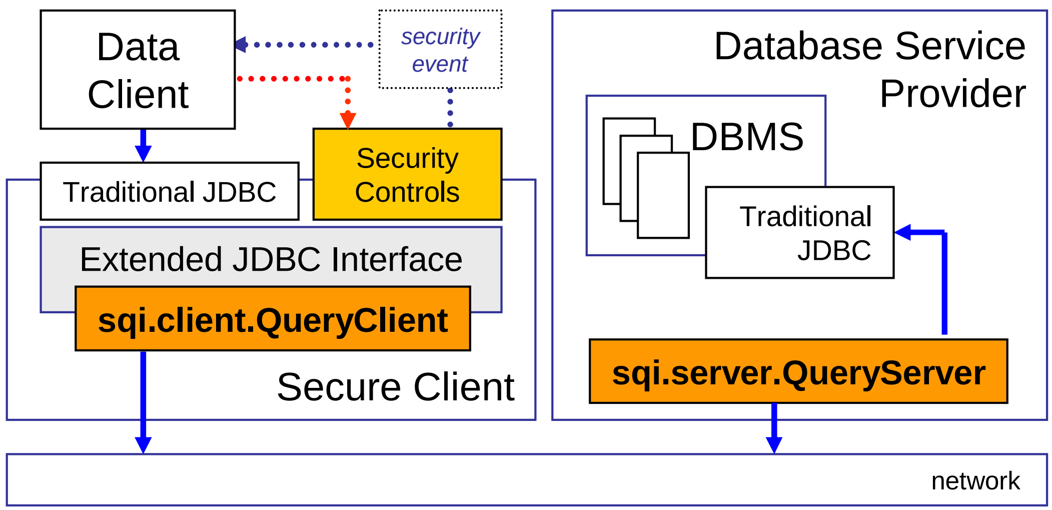
query completeness proofs (lazy server)



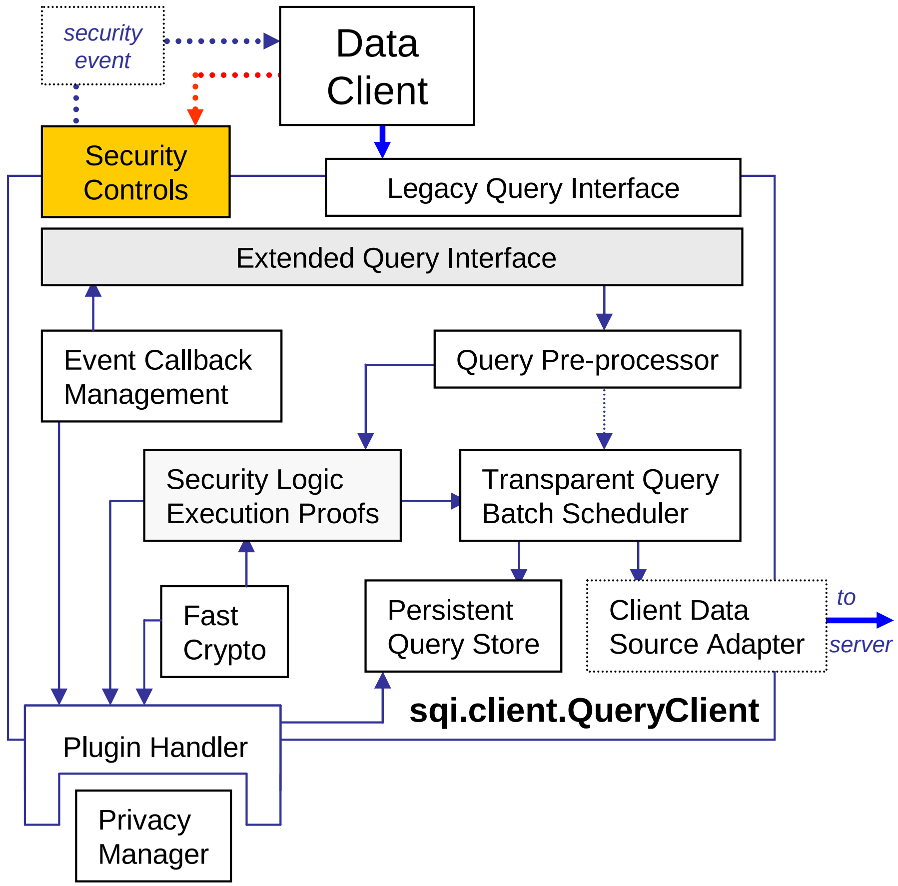
A challenge token (computed by client) is sent together with the batch of queries. Upon return, batch execution is proved by (x==x').

$$C(\mathbb{Q}, x, \epsilon) = \{H(\epsilon || \rho(Q_x)), \epsilon\}$$

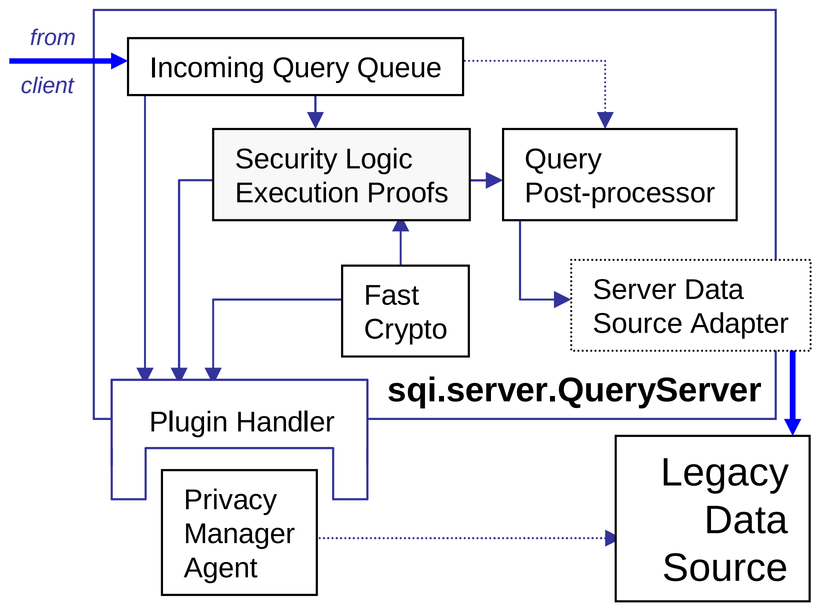
secure query interface (SQi)



SQi: client interface

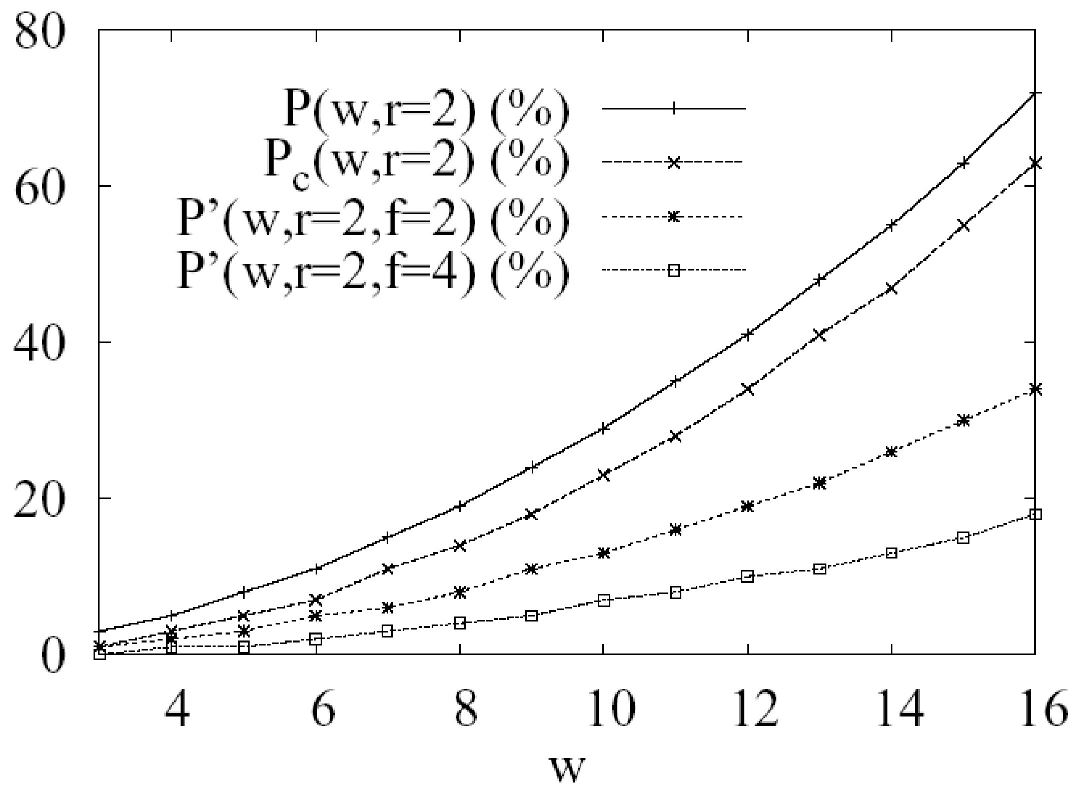


SQi: server



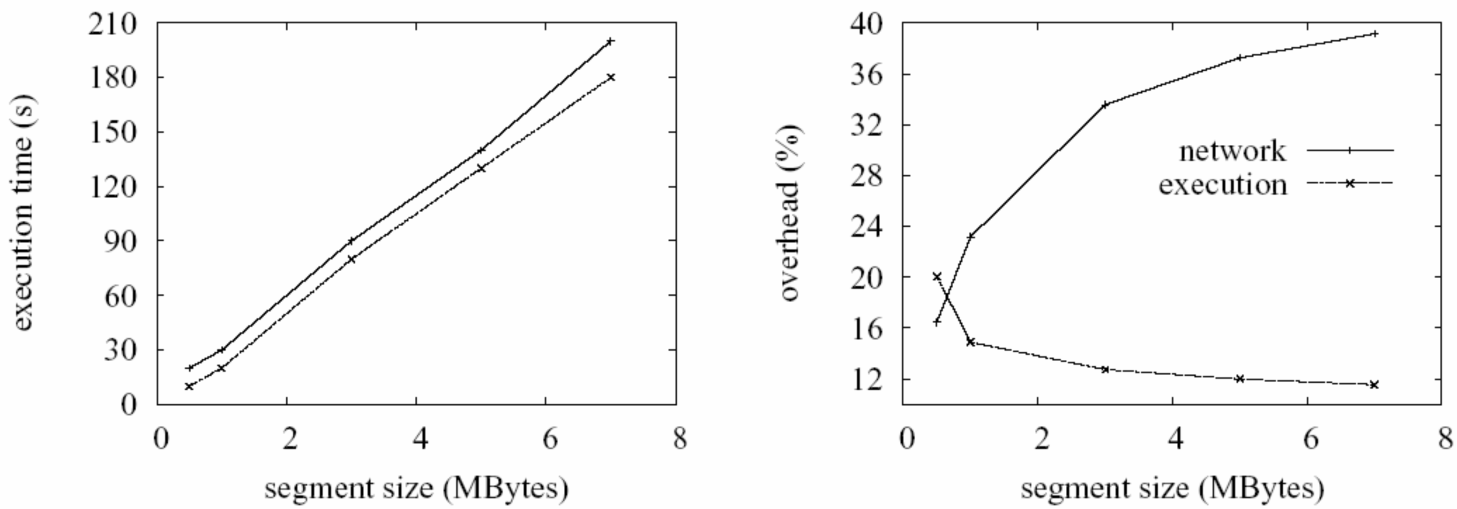
from
client
Incoming Query Queue
Security Logic
Execution Proofs
Query
Post-processor
Fast
Crypto
Server Data
Source Adapter
sqi.server.QueryServer
Plugin Handler
Privacy
Manager
Agent
Legacy
Data
Source

success probability of cheating



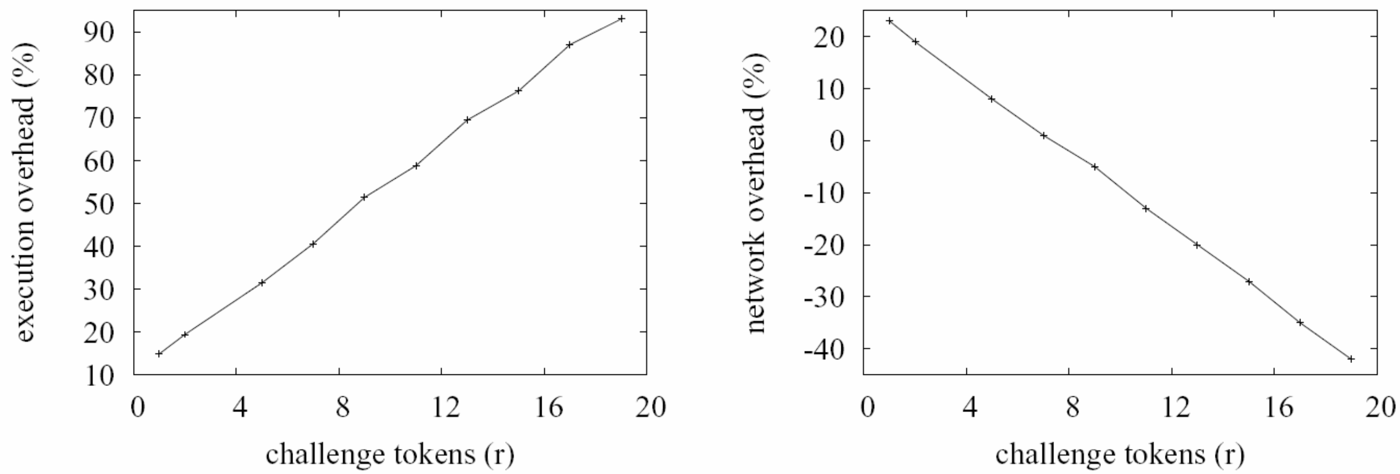
The behavior of $P'(w, r, f)$ (fake tokens) plotted against $P_c(w, r)$ (client-side result checking mechanism) showing that the query execution proof mechanism (with fake tokens) significantly decreases the ability to "get away" with less work.

execution times



(a) Execution times behave naturally linear in the size of the input, (b) Execution time and network overheads behavior with increasing segment size ($\mathtt{r} = 1$).

overheads are reasonable



Overheads with increasing number of challenge tokens per batch: (a) execution overhead increases (b) network overhead decreases and eventually becomes negative.

beyond laziness

- **client-side result checking**
  - weaker assurances of a stronger type ☺
- **secure hardware (we'll see later)**
- **etc. ?**

pointer

- ⊗ data outsourcing
- ⊗ query completeness
- → ⊕ searching
- ⊗ secure co-processor

searching: fun for sure, but important?

Selected scenarios

- compromised server (e.g. network context)
- secure email server
  - do not allow sysadmin to read email ☺
- secure networked file system
  - unable to deploy forensics (without data owner consent)
- secures (from commercial competition):
  - company data
  - data access patterns

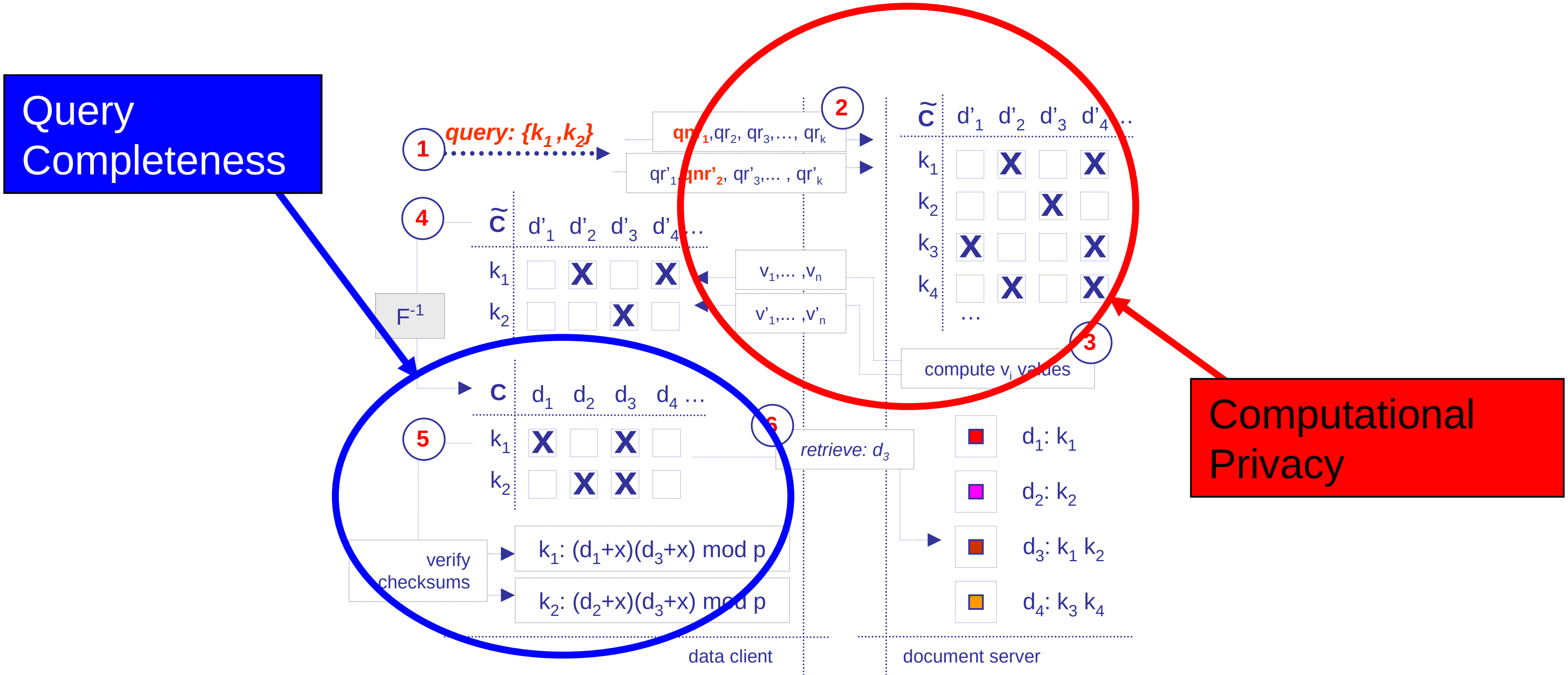
sample: "return **all emails containing** 'John' and 'lunch'"

searching: fun for sure, but important?!

Challenges

- result assurances
  - completeness
  - correctness
- confidentiality of data
- obliviousness
  - privacy of searches
  - no correlation leaks
- overheads
  - computational
  - network
  - storage constrained client
- dynamic (updates)

searching: helicopter overview



Deploying a modified version of computational PIR targeted at a server-side indexing structure to achieve complete privacy.

pointer

⊗ data outsourcing

⊗ query completeness

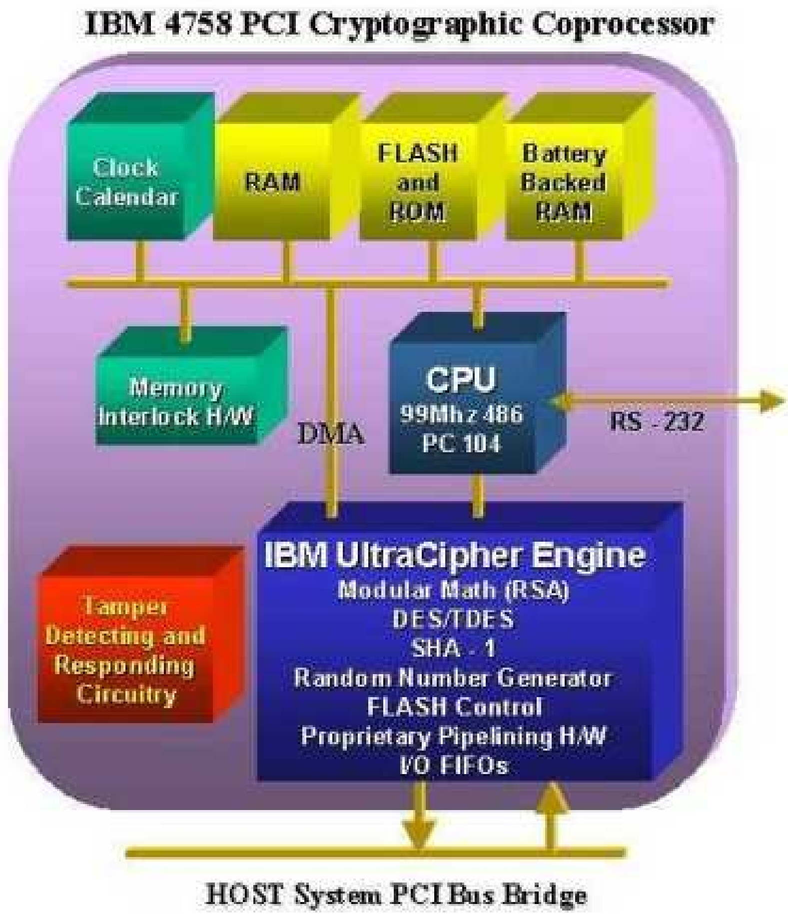
⊗ searching

→ ⊕ secure co-processor

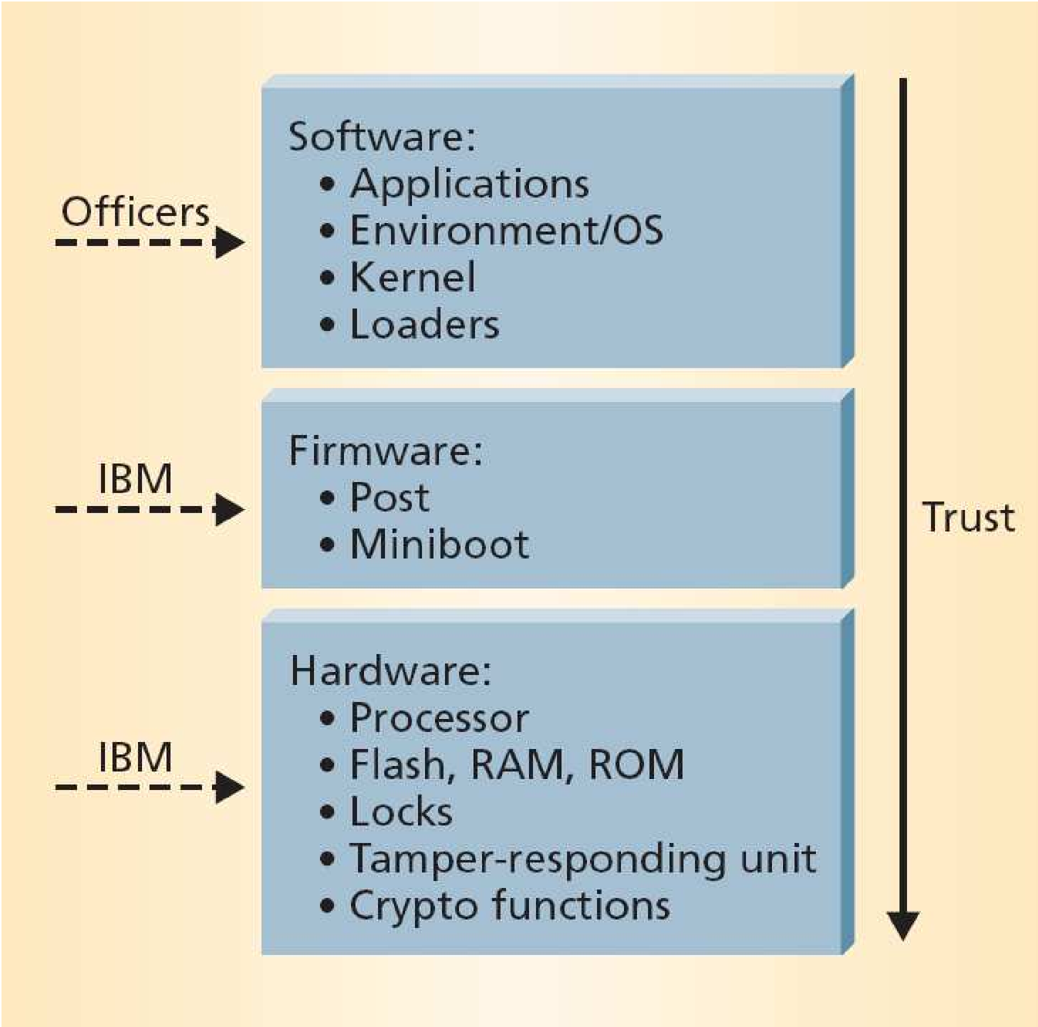
e.g. IBM 4758 (4764)

STONY BROOK
STATE UNIVERSITY OF NEW YORK

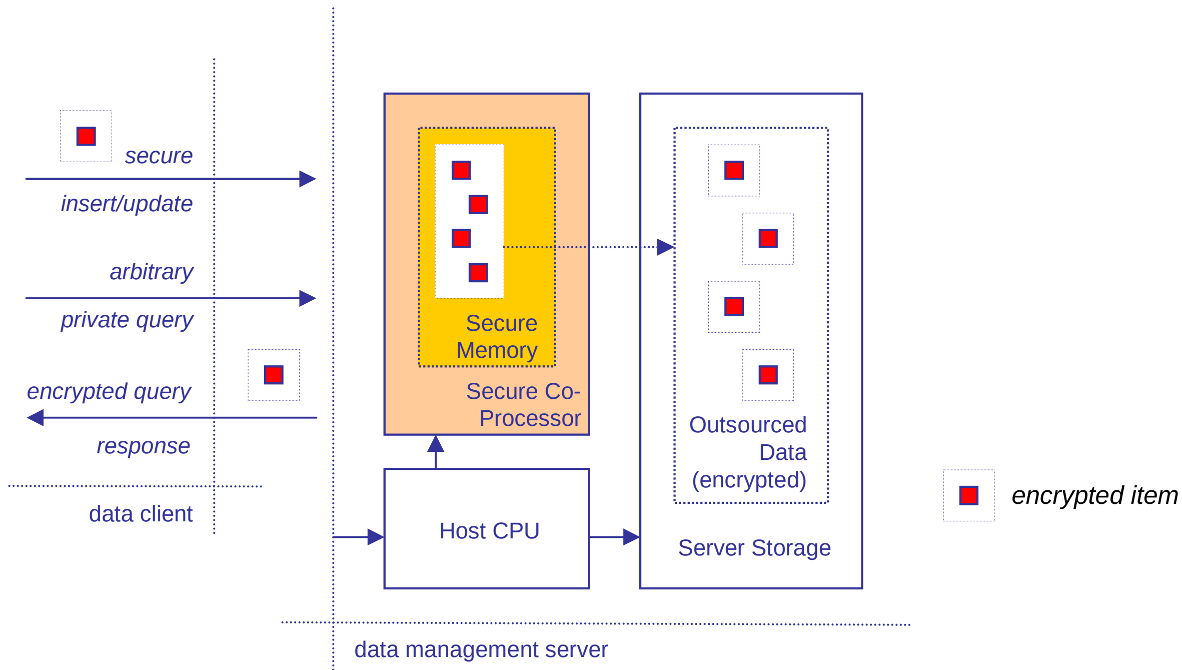
architecture overview (4758)



trust propagation (4758)

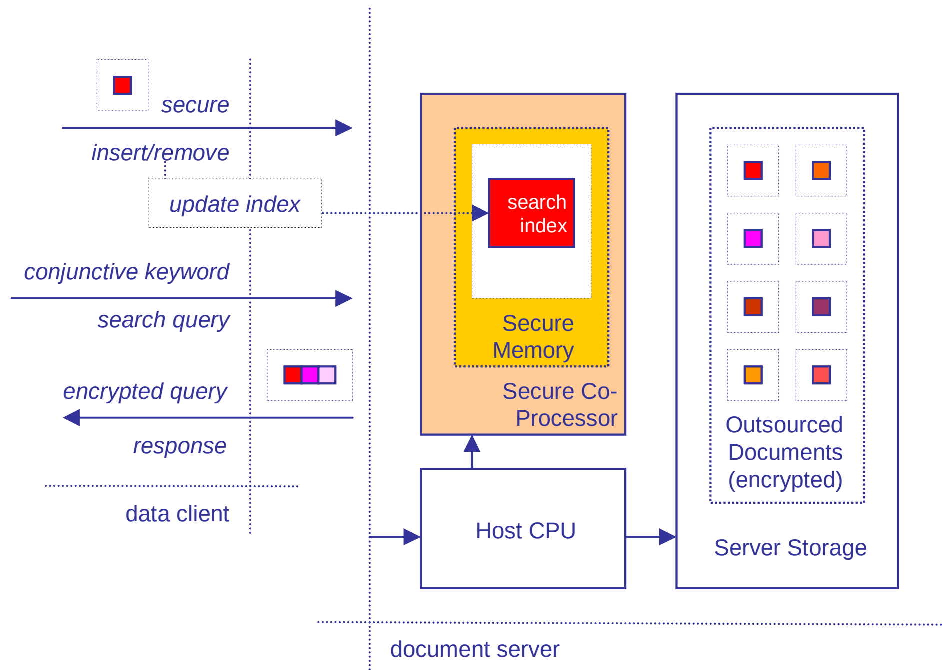


scpu: possible benefits



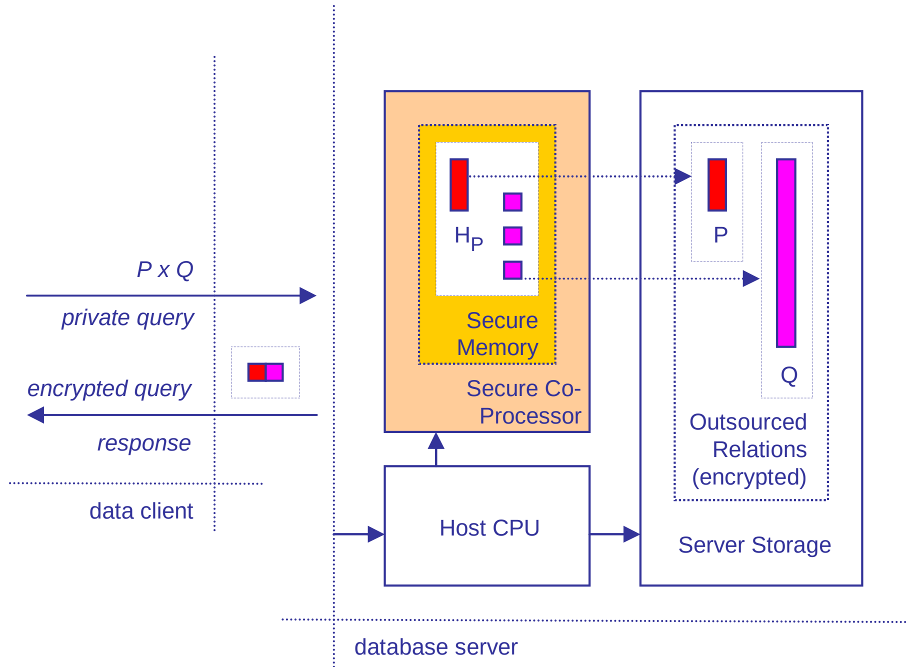
A secure co-processor on the data management side may allow for significant leaps in expressivity for queries where privacy and completeness assurance are important.

scpu: searching with privacy



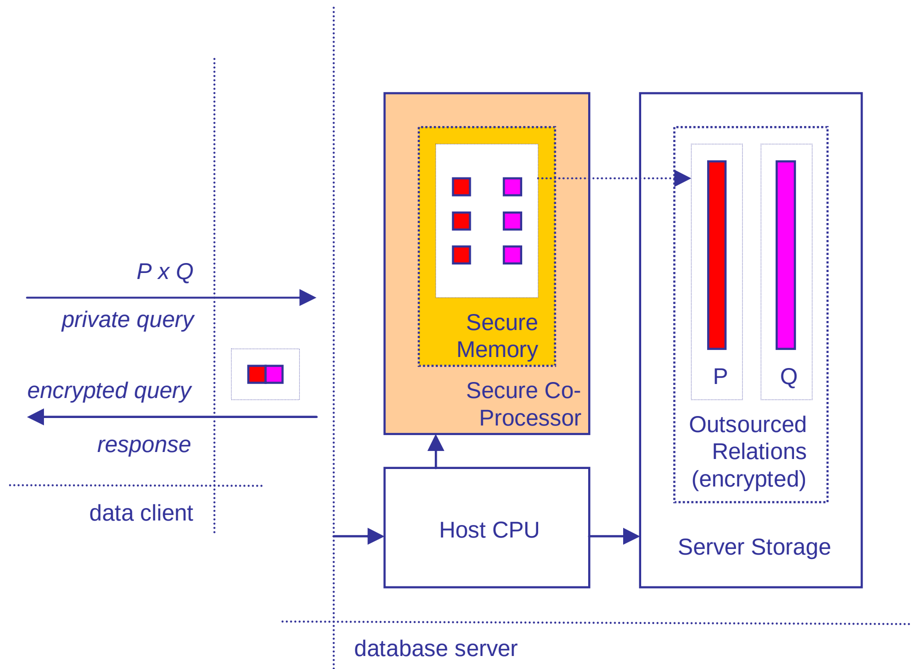
For conjunctive keyword searches on document (email, files) servers, oblivious search index structures could be queried in secure memory achieving a novel zero-leak query model.

scpu: hash-join (with privacy)



Hash-JOIN could be naturally accomodated.

scpu: merge-join (with privacy)



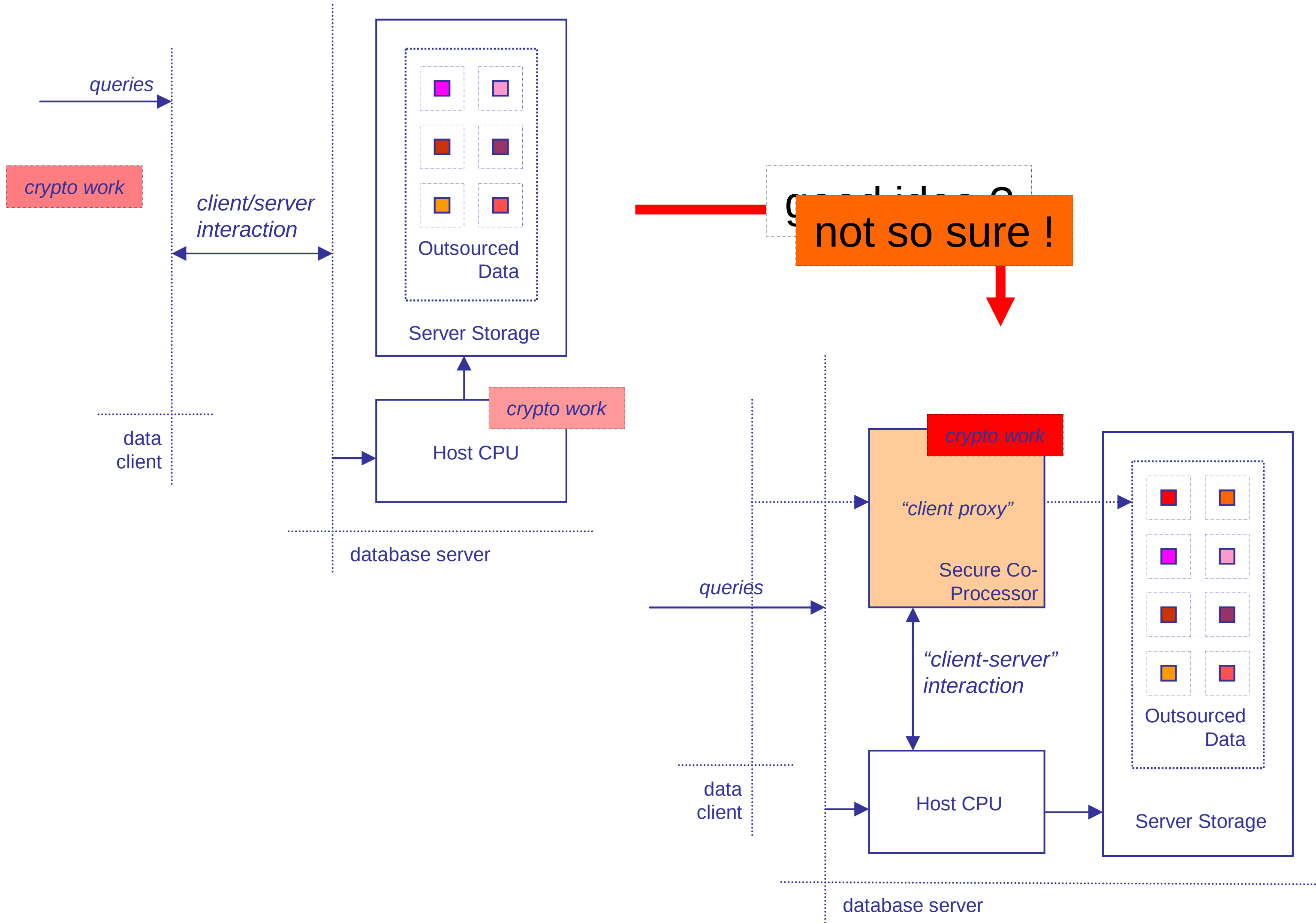
For Merge-JOIN, order-preserving encryption primitives could be deployed to minimize the amount of data parsing required in the sorting phase.

scpu: what about general semantics ?

How do we aproach the problem of arbitrary query expressivity with strong computational (at least) privacy ?

Let's look at things we <u>don't</u> "believe" in ☺ …

sample "wouldn't do": SCPU=client proxy



scpu: some things we are afraid to do

- Process entire queries on SCPU (!)
- Dedicate (one) SCPU per query or equivalent
  - e.g., limit TPS by SCPU TPS
- Synchronize CPU with SCPU
  - e.g., block main CPU until SCPU completes
- Transfer >= O(n) on SCPU-CPU bus (!)
- Anything else un-smart ☺

selected related research (SCPU)

Kenneth Goldman, Enriquillo Valdez: “Matchbox: Secure Data Sharing”, IEEE Internet Computing 2004

“Practical server privacy with secure coprocessors”, IBM Systems Journal 2001, S. W. Smith, D. Safford

J. Marchesini, S.W. Smith, “SHEMP: Secure Hardware Enhanced MyProxy” Technical Report TR2005-532, Department of Computer Science, Dartmouth College, February 2005.

A. Iliev, S.W. Smith, "Protecting Client Privacy with Trusted Computing at the Server", IEEE Security and Privacy, March/April 2005

A. Iliev, S.W. Smith, "Private Information Storage with Logarithmic-space Secure Hardware.", 3rd Working Conference on Privacy and Anonymity in Networked and Distributed Systems.

A. Iliev, S.W. Smith, "Prototyping an Armored Data Vault: Rights Management on Big Brother's Computer.", Privacy-Enhancing Technology 2002

E. Mykletun and G. Tsudik, “On using Secure Hardware in Outsourced Databases”, International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems, January 2005

Related research at IBM TJ Watson (Bishwaranjan Bhattacharjee a.o.)

cat /proc/lunchtime

Thank You ☺