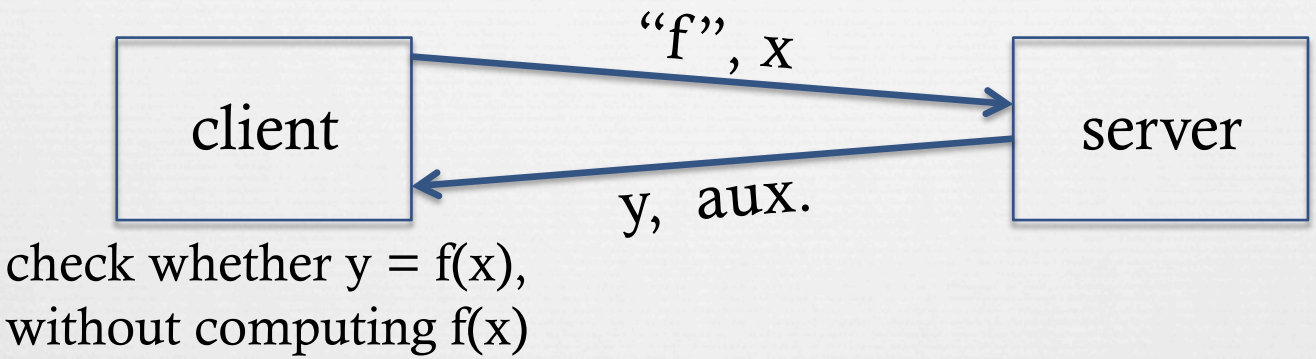
Making argument systems for outsourced computation practical (sometimes)

Srinath Setty, Richard McPherson,
Andrew J. Blumberg, and Michael Walfish

The University of Texas at Austin

By verified outsourced computation, we mean the following:

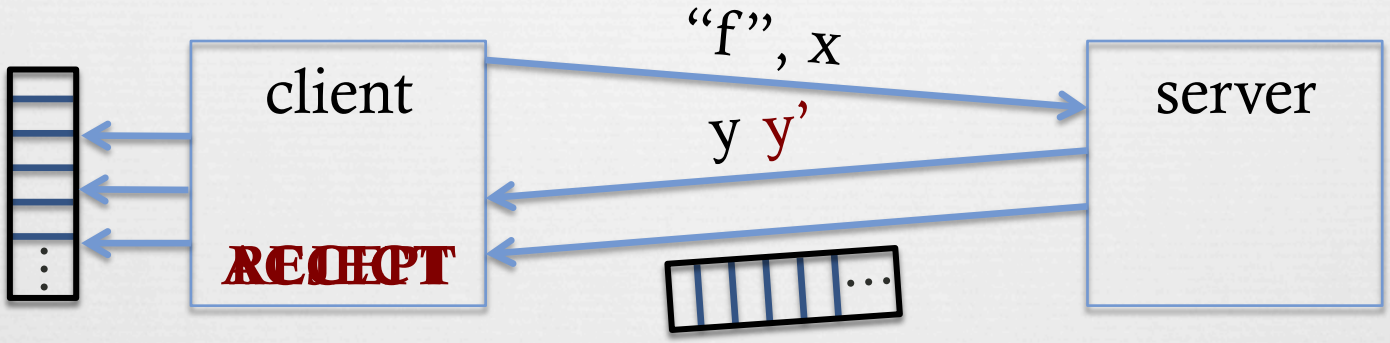


The motivation is 3rd party computing: cloud, volunteers, etc.

We desire the following properties in the above exchange:

1. Unconditional, meaning no assumptions about the server
2. General-purpose, meaning not specialized to a particular f
3. Practical, or at least conceivably practical soon

Theory can supposedly help. Consider the theory of Probabilistically Checkable Proofs (PCPs). [ALMSS JACM98, AS JACM98]



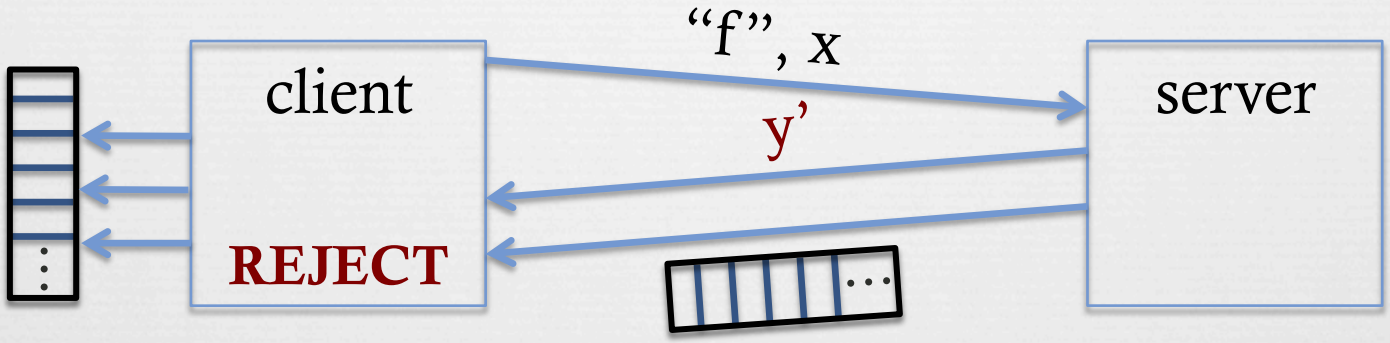
Unfortunately, the constants and proof length are outrageous.

Using a naive PCP implementation, verifying multiplication of 400×400 matrices would take 500 trillion CPU years (seriously).

500 trillion is a big number.

For example, I can beat Michael Jordan in one-on-one basketball only one time out of 500 trillion.

Theory can supposedly help. Consider the theory of Probabilistically Checkable Proofs (PCPs). [ALMSS JACM98, AS JACM98]



Unfortunately, the constants and proof length are outrageous.

Using a naive PCP implementation, verifying multiplication of 400×400 matrices would take 500 trillion CPU years (seriously).

We have reduced the costs of a PCP-based argument system by Ishai et al. [CCC07] by 20 orders of magnitude, with proof.

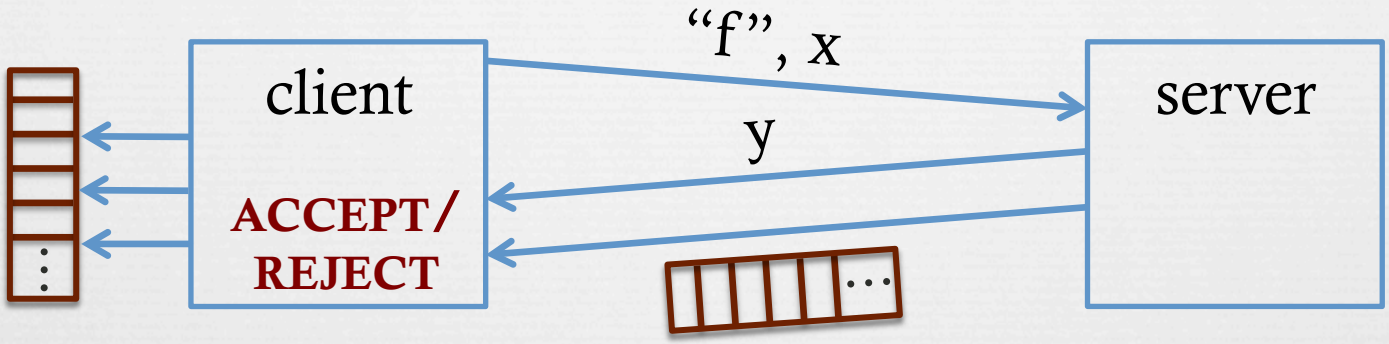
We have implemented the refinements in a system, PEPPER, that is not ready for prime time but is practical in some cases.

Our conclusion is that PCPs are a potentially promising tool for building secure systems.

(1) The design of PEPPER

(2) Experimental results, limitations, and outlook

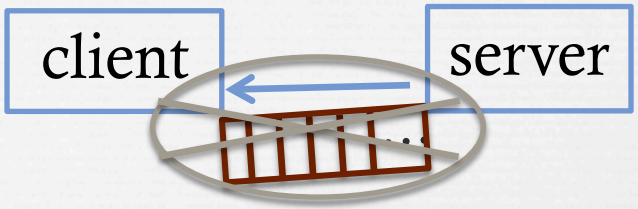
Pepper incorporates PCPs but not like this:



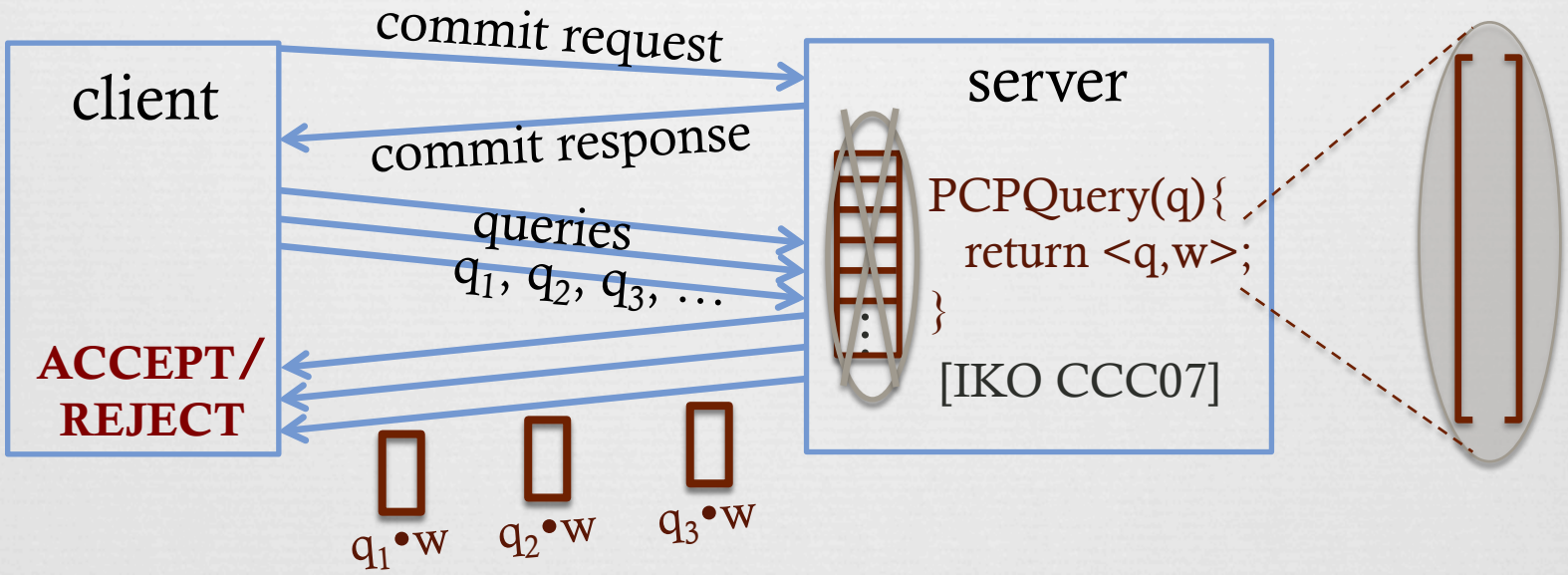
The proof is not drawn to scale: it is far too long to be transferred.

(Even the asymptotically short PCPs [BGHSV CCC05, BGHSV SIJC06, Dinur JACM07, BS SIJC08] have prohibitive constants.)

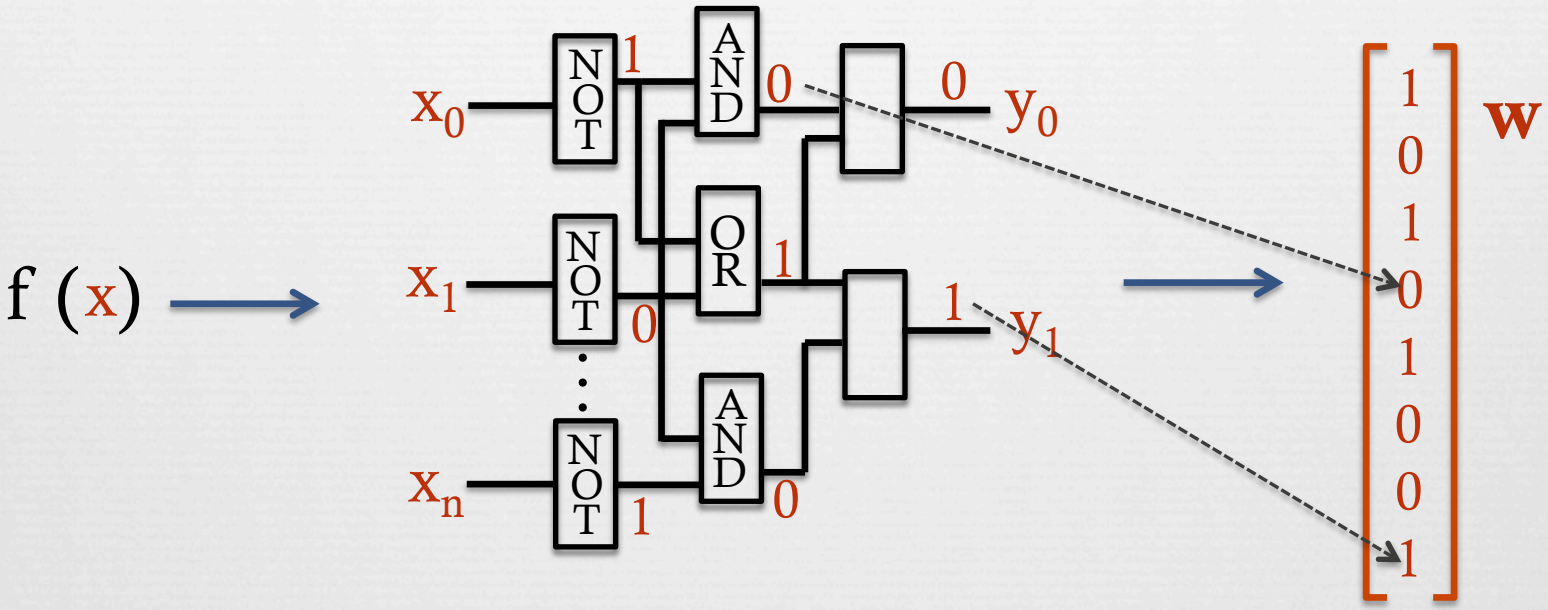
Instead of transferring the PCP …



… Pepper uses an efficient argument [Kilian CRYPTO 92,95]:



The server's vector **w** encodes an execution trace of f(x).

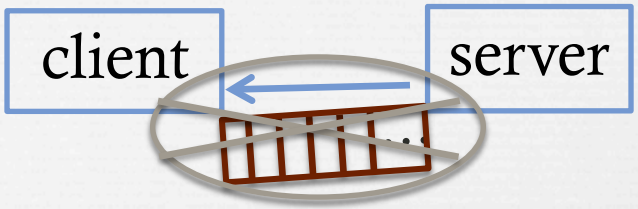


What is in **w**?

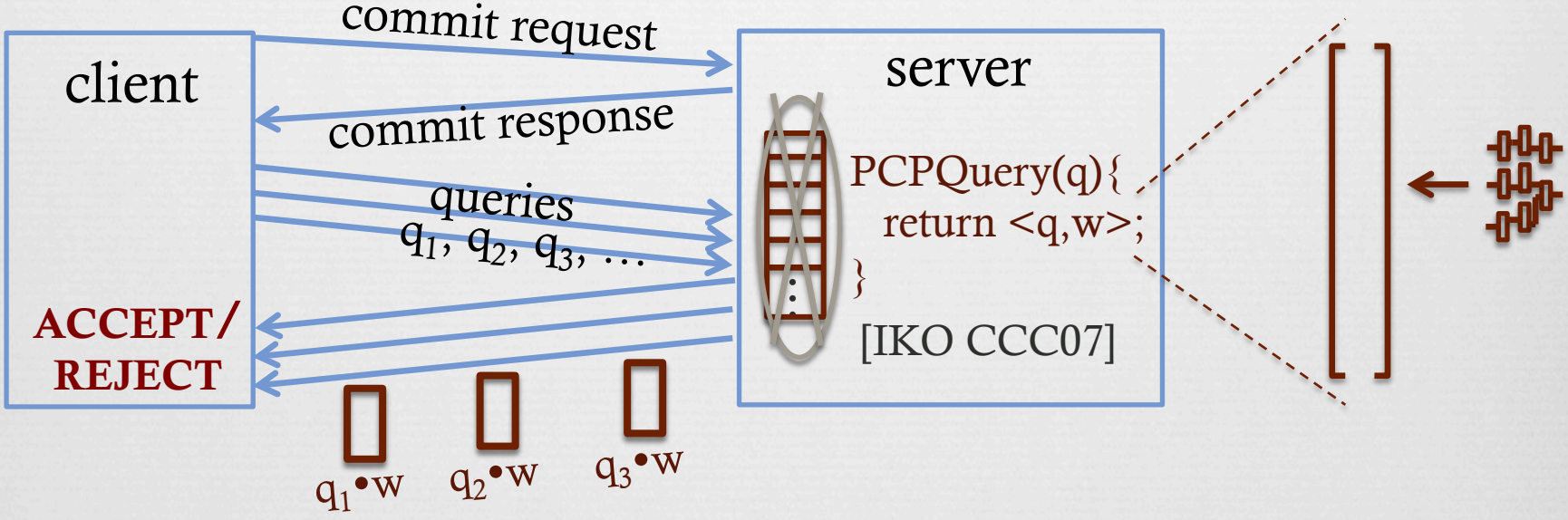
(1) An entry for each wire; and

(2) An entry for the product of each pair of wires.

Instead of transferring the PCP …

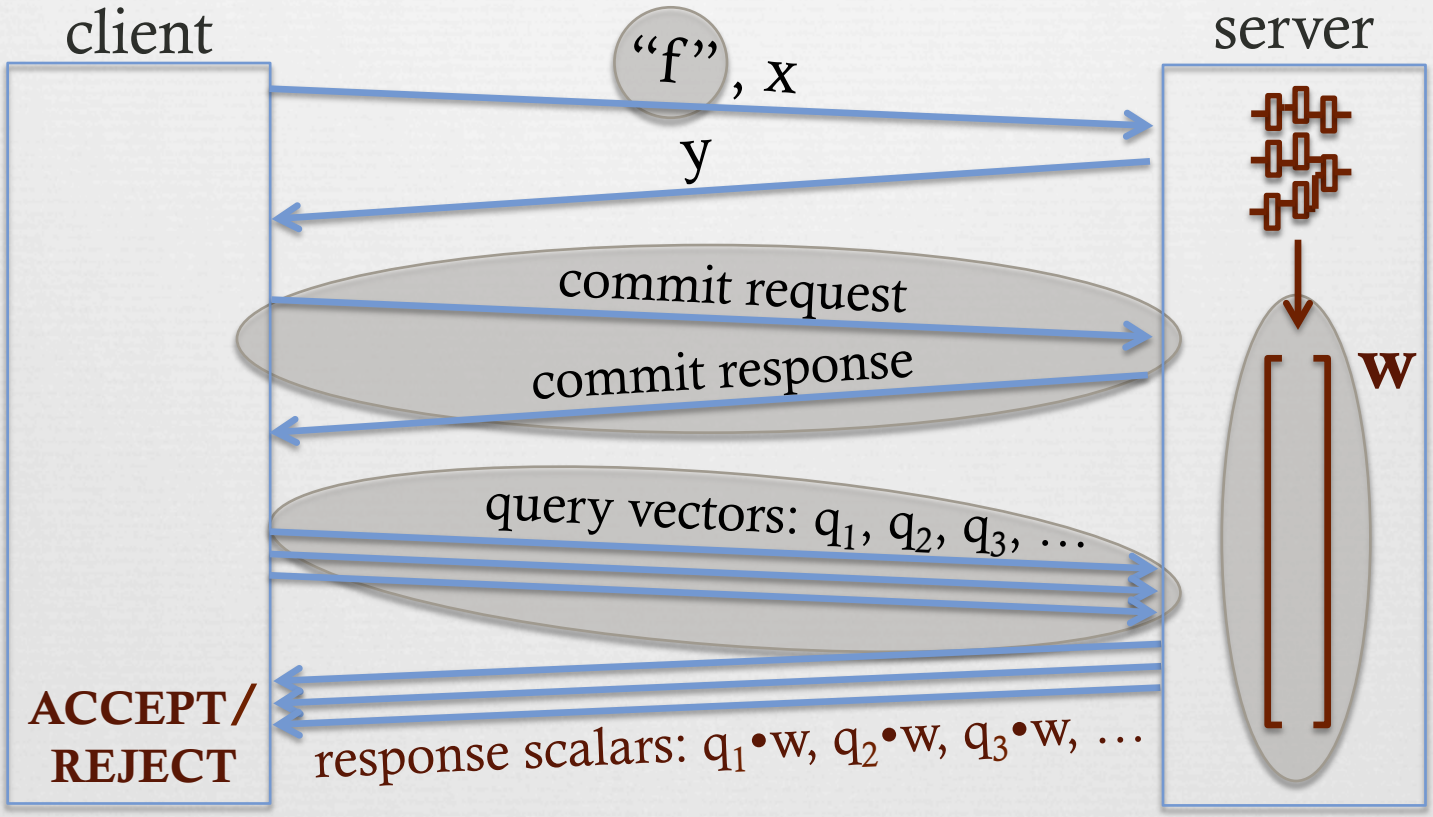


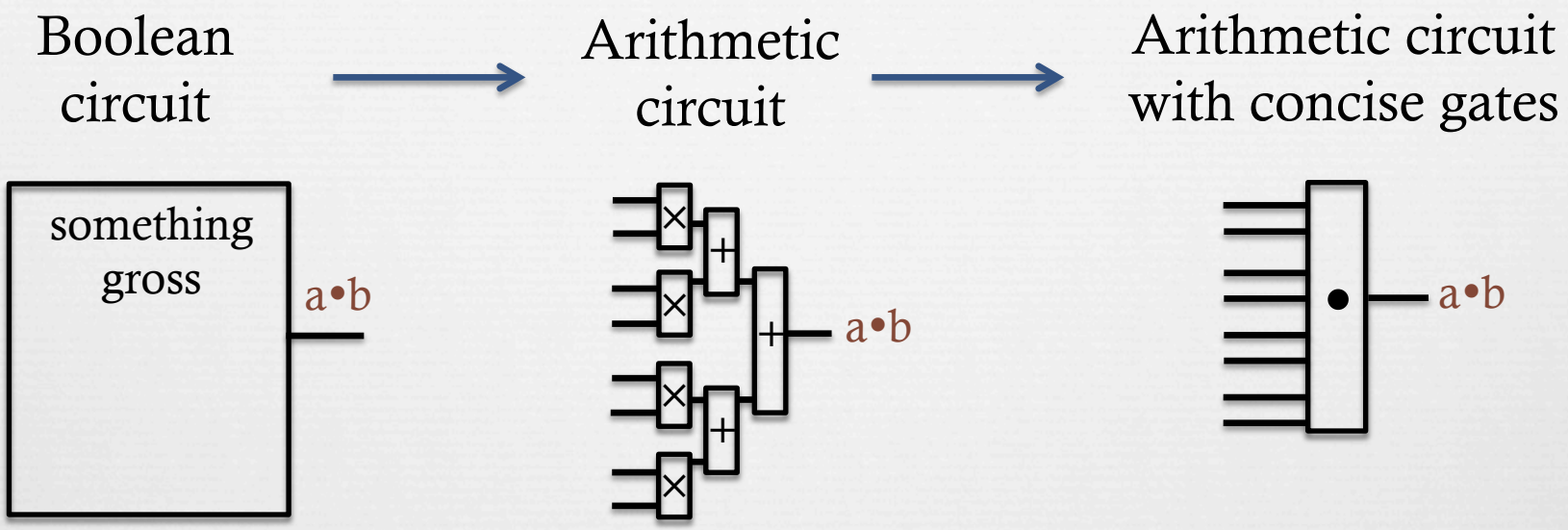
… Pepper uses an efficient argument [Kilian CRYPTO 92,95]:



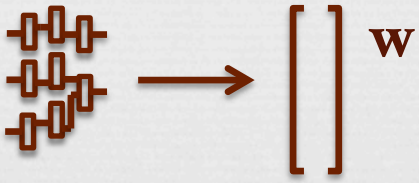
This is still too costly (by a factor of 10^{22}), but it is promising.

PEPPER incorporates four refinements to [IKO CCC07], with proof.



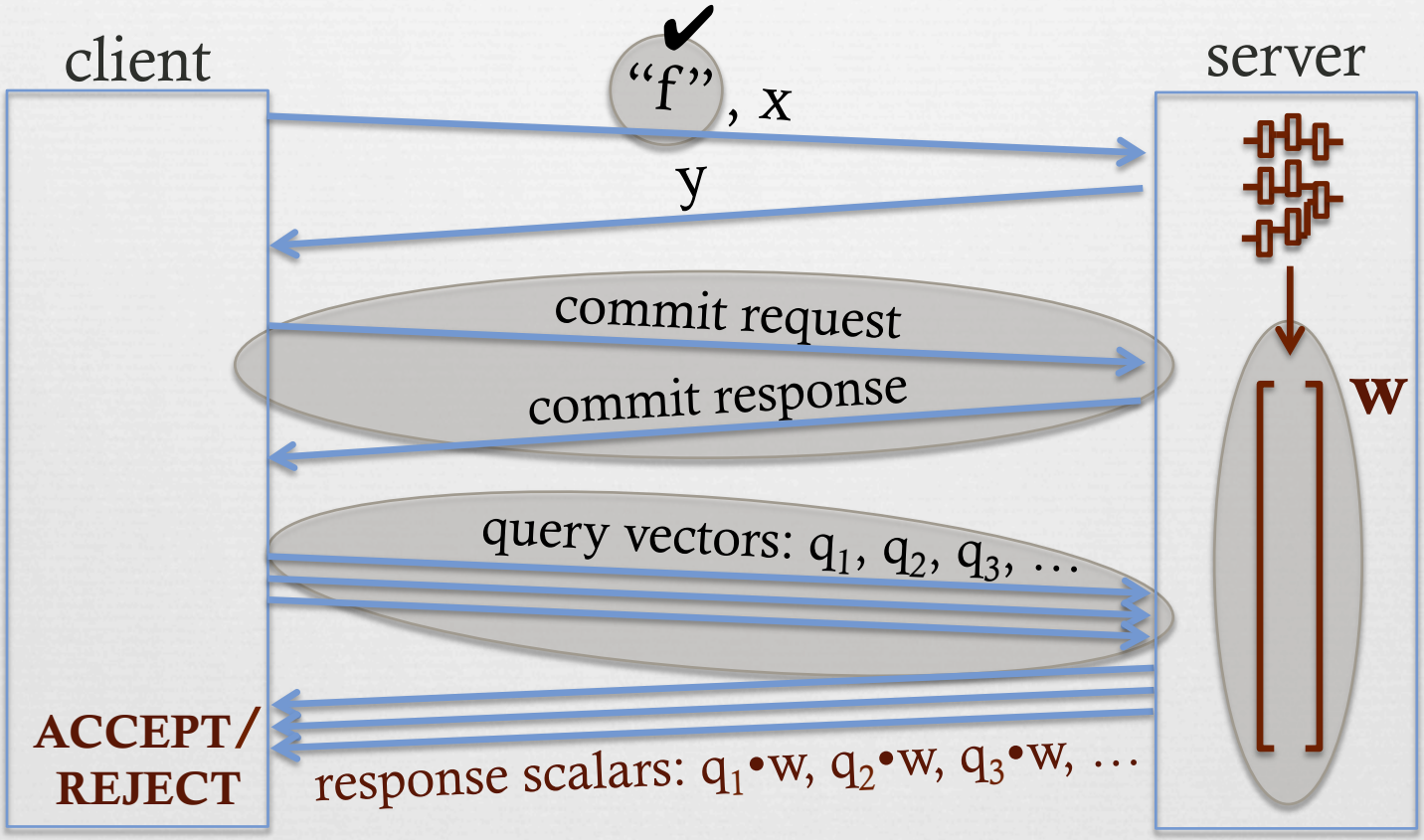


This refinement works best for a restricted class of computations: straight-line, parallelizable, numerical.



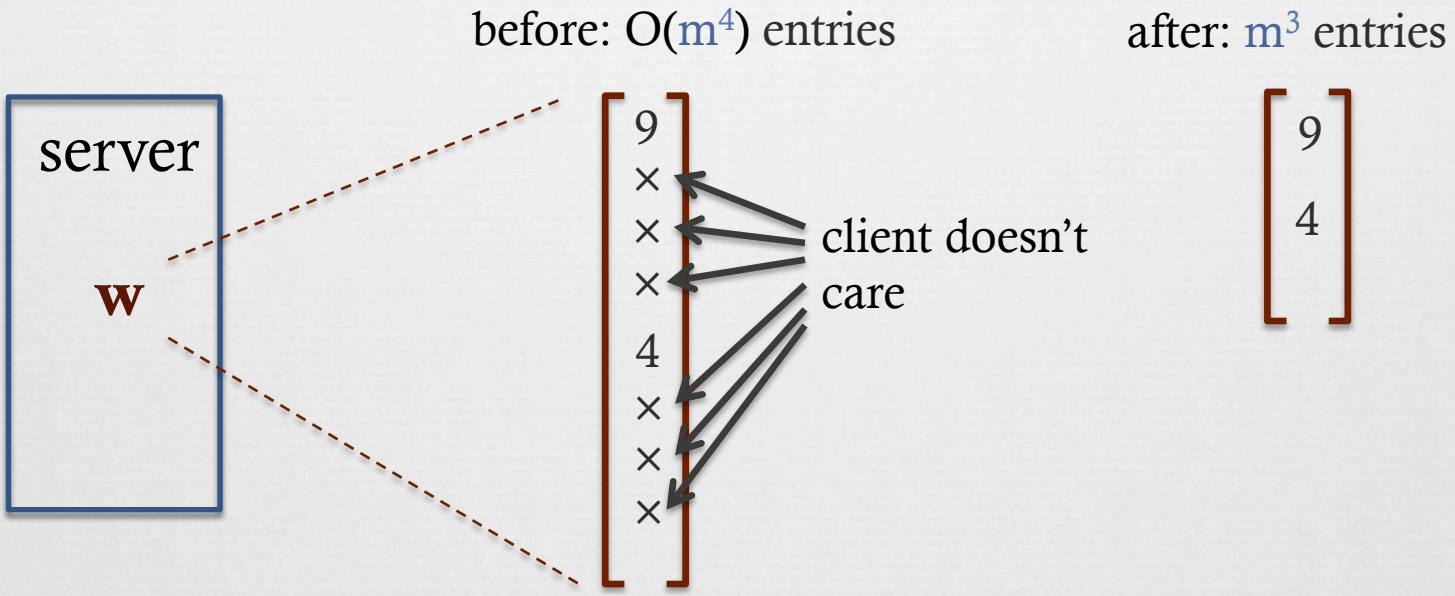
Consider $m \times m$ matrix multiplication as our computation f:

- The Boolean circuit has $O(m^3)$ gates ⟶ **w** has $O(m^6)$ entries
- The new representation has m^2 gates ⟶ **w** has $O(m^4)$ entries

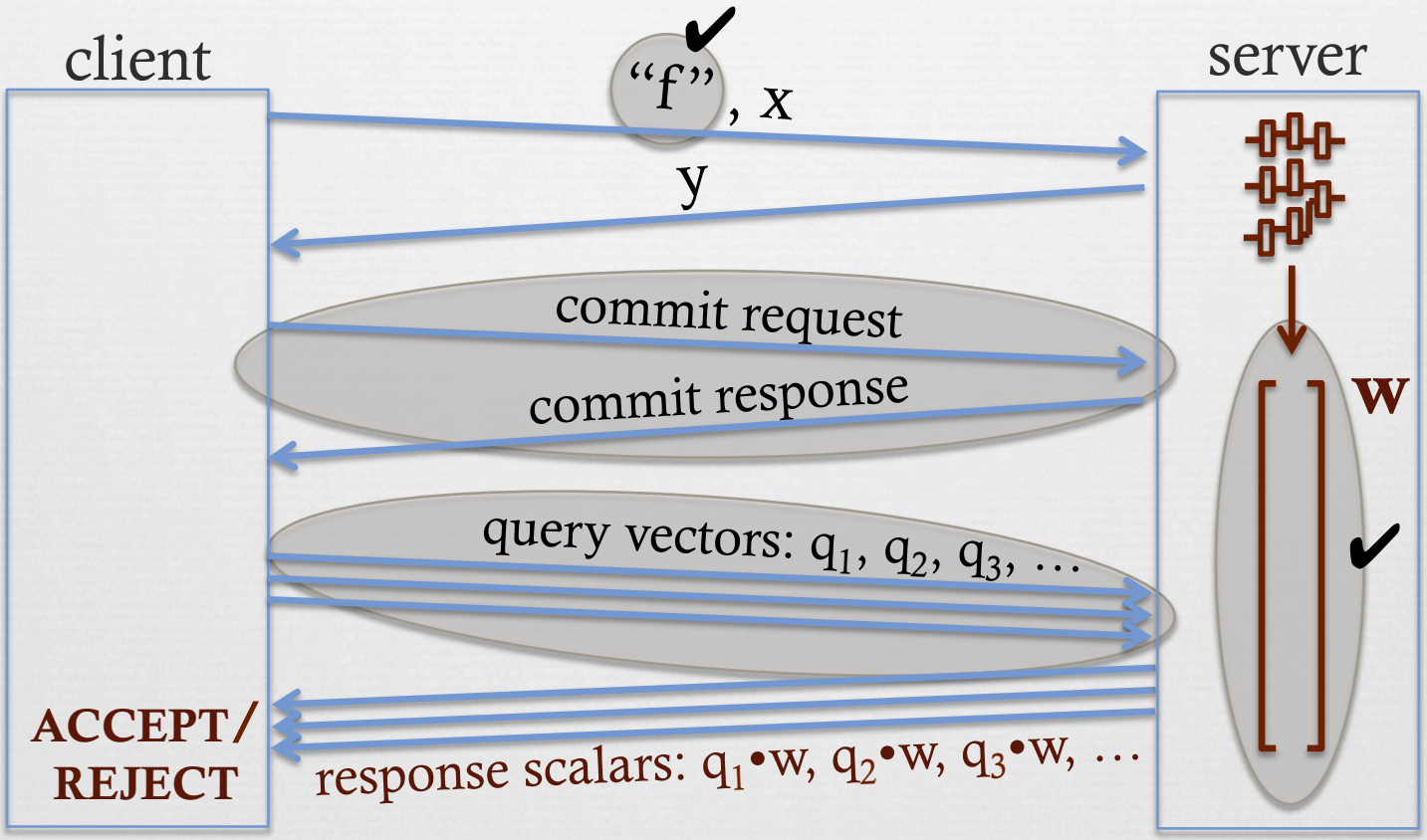

client
server
✔
"f", x
y
commit request
commit response
query vectors: q_1, q_2, q_3, …
w
ACCEPT/
REJECT
response scalars: $q_1 \cdot w$, $q_2 \cdot w$, $q_3 \cdot w$, …

We can sometimes exploit the structure of a computation.

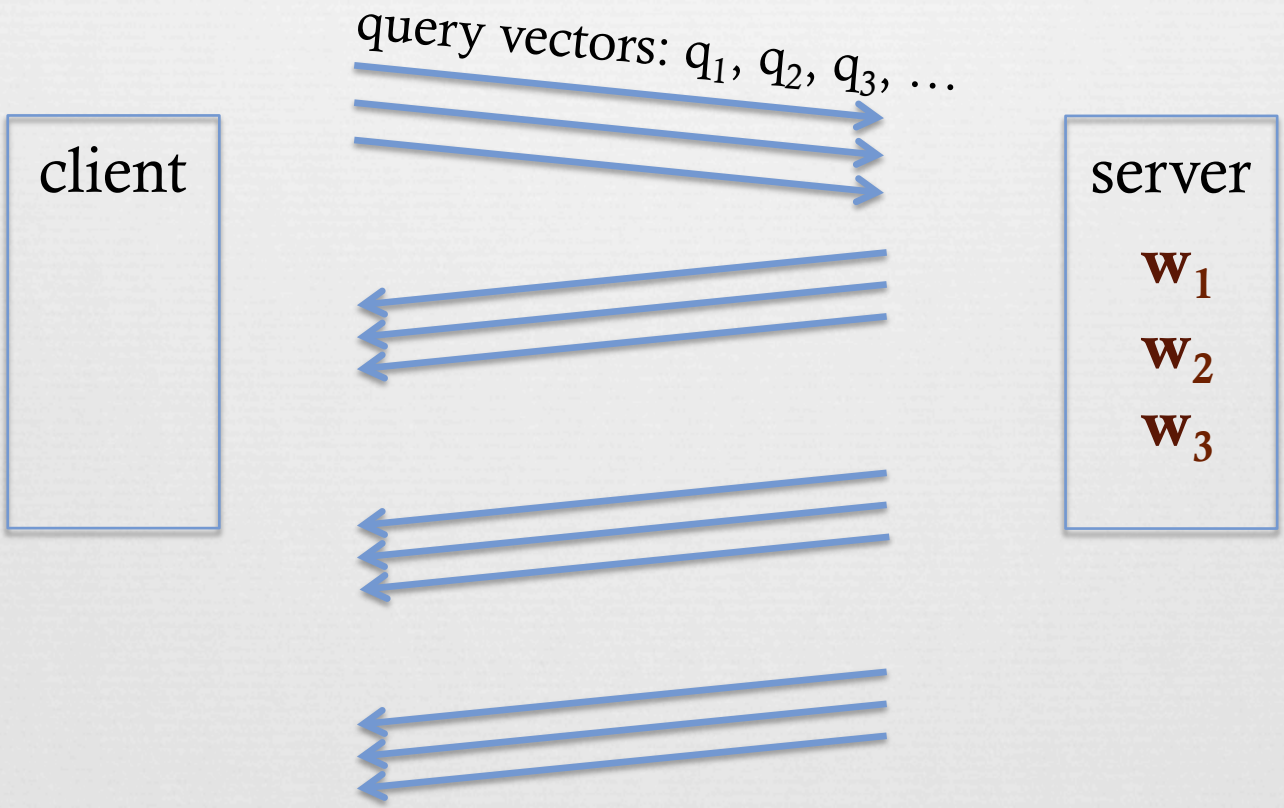
Consider $m \times m$ matrix multiplication as our computation f:



This eliminates the server's PCP-based overhead, and may apply to PCPs more broadly.

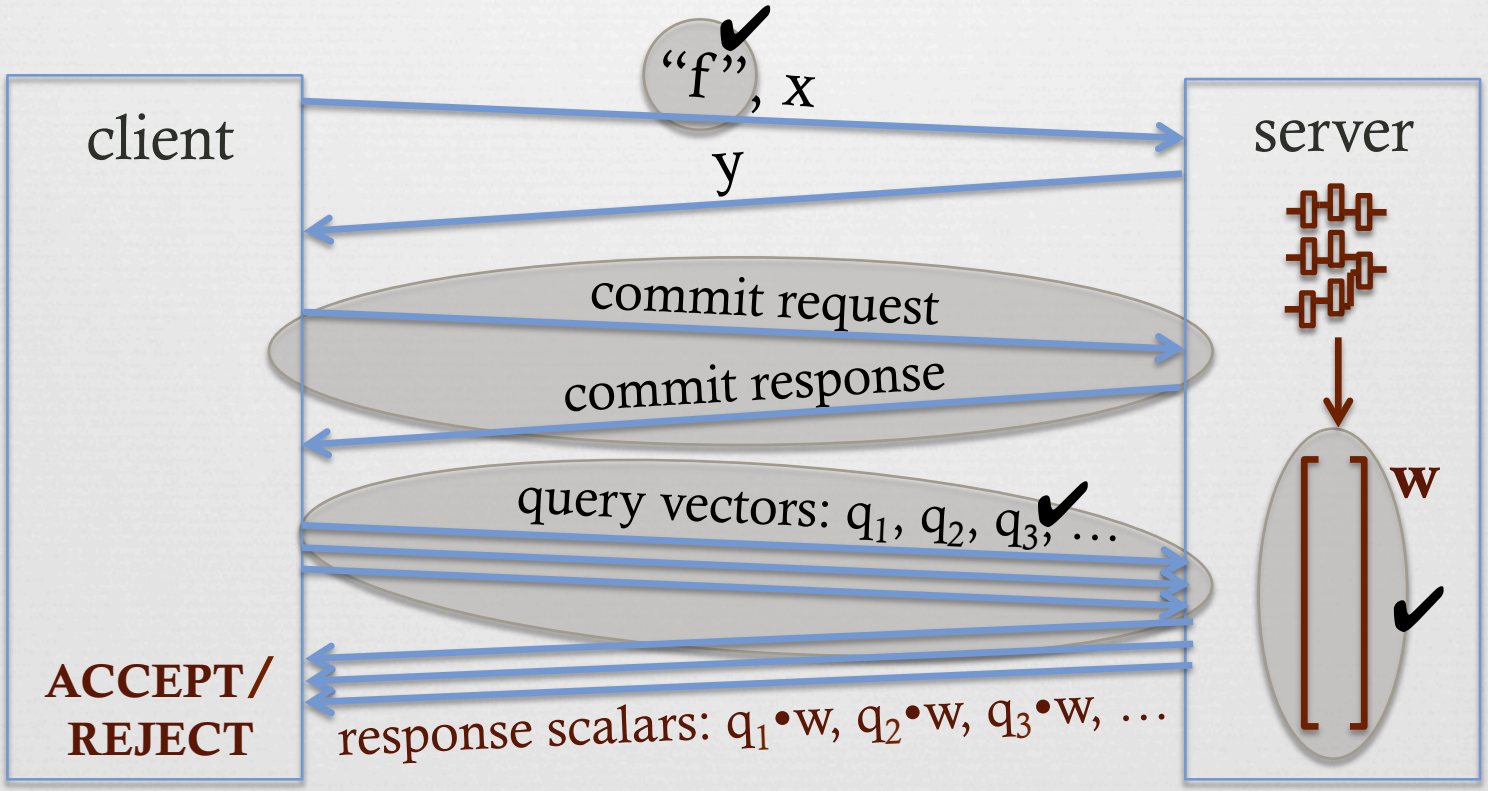

client
server
"f", x
y
commit request
commit response
query vectors: q_1, q_2, q_3, …
w
ACCEPT/
REJECT
response scalars: $q_1 \cdot w$, $q_2 \cdot w$, $q_3 \cdot w$, …

The client amortizes its overhead by reusing queries over multiple runs. Each run has the same f but different input x.



PEPPER generalizes the commitment primitive of Ishai et al. [CCC07].

With the new primitive, the client can issue multiple queries for the price of encrypting only a single query.



✔ (1) The design of PEPPER

(2) Experimental results, limitations, and outlook

Consider amortized costs for multiplication of 400×400 matrices:

	Under the theory, naively applied	Under PEPPER
client CPU time	>100 trillion years	1.1 seconds
server CPU time	>100 trillion years	1.6 hours

(assumes 2.4 Ghz CPU)

However, the batch size is large, so these numbers are not ideal.

PEPPER is not ready for prime time, for several reasons:

1. The client breaks even only for large batch sizes.
2. The server's burden is too high, still.
3. The approach is plausible for only a class of computations.

We relate PEPPER to prior work in terms of our three goals.

1. General-purpose and practical; gives up unconditional

- Replication ([Castro & Liskov TOCS02]), trusted hardware ([Chiesa & Tromer ICS10, SSW TRUST10]), auditing ([DJMM ICDCS04, HKD SOSP07, Kissner & Song ACNS04, MWR NDSS99])

2. Unconditional; gives up being general-purpose

- [BGV CRYPTO11, Boneh & Freeman EUROCRYPT11, Golle & Mironov RSA01, Sion VLDB05, THHSY PET09, WRW INFOCOM11, Atallah & Frikken ASIACCS10, Freivalds MFCS79]
- Toward practical Interactive Proofs [CMT ITCS12, GKR STOC08]

3. Unconditional and general-purpose; gives up practicality

- Fully homomorphic encryption, secure multi-party computation [CKV CRYPTO10, GGP CRYPTO10, AIK ICALP10]

We have reduced the costs of a PCP-based argument system by Ishai et al. [CCC07] by 20 orders of magnitude, with proof.

We have implemented the refinements in a system, PEPPER, that is not ready for prime time but is practical in some cases.

Our conclusions are that PCPs are a potentially useful tool for real systems, and that the research area is promising.