vSQL: Verifying Arbitrary SQL Queries over Dynamic Outsourced Databases

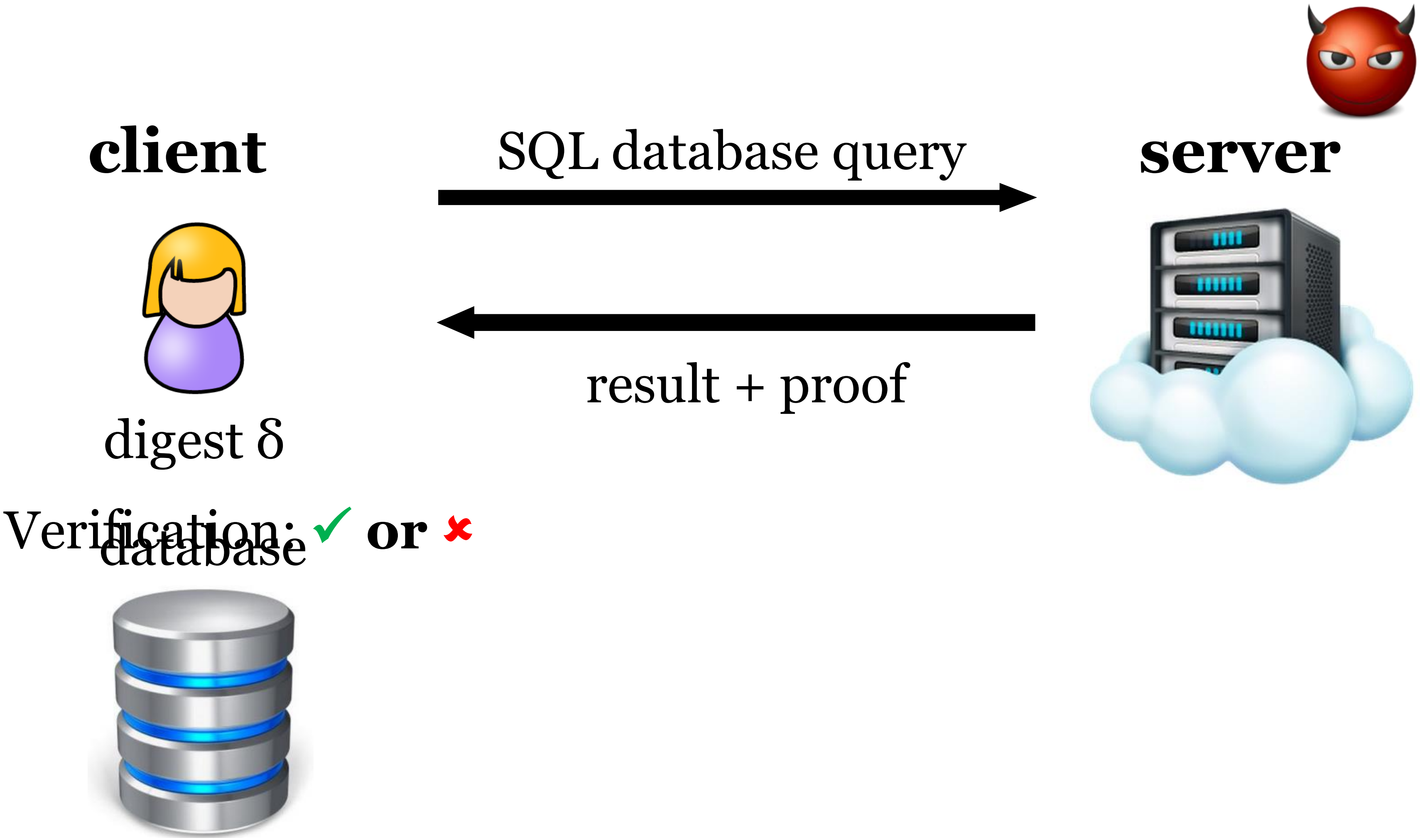
Yupeng Zhang, Daniel Genkin, Jonathan Katz,
Dimitrios Papadopoulos and Charalampos Papamanthou

University of Maryland 1856

Penn University of Pennsylvania

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Verifiable Databases



client

SQL database query

server

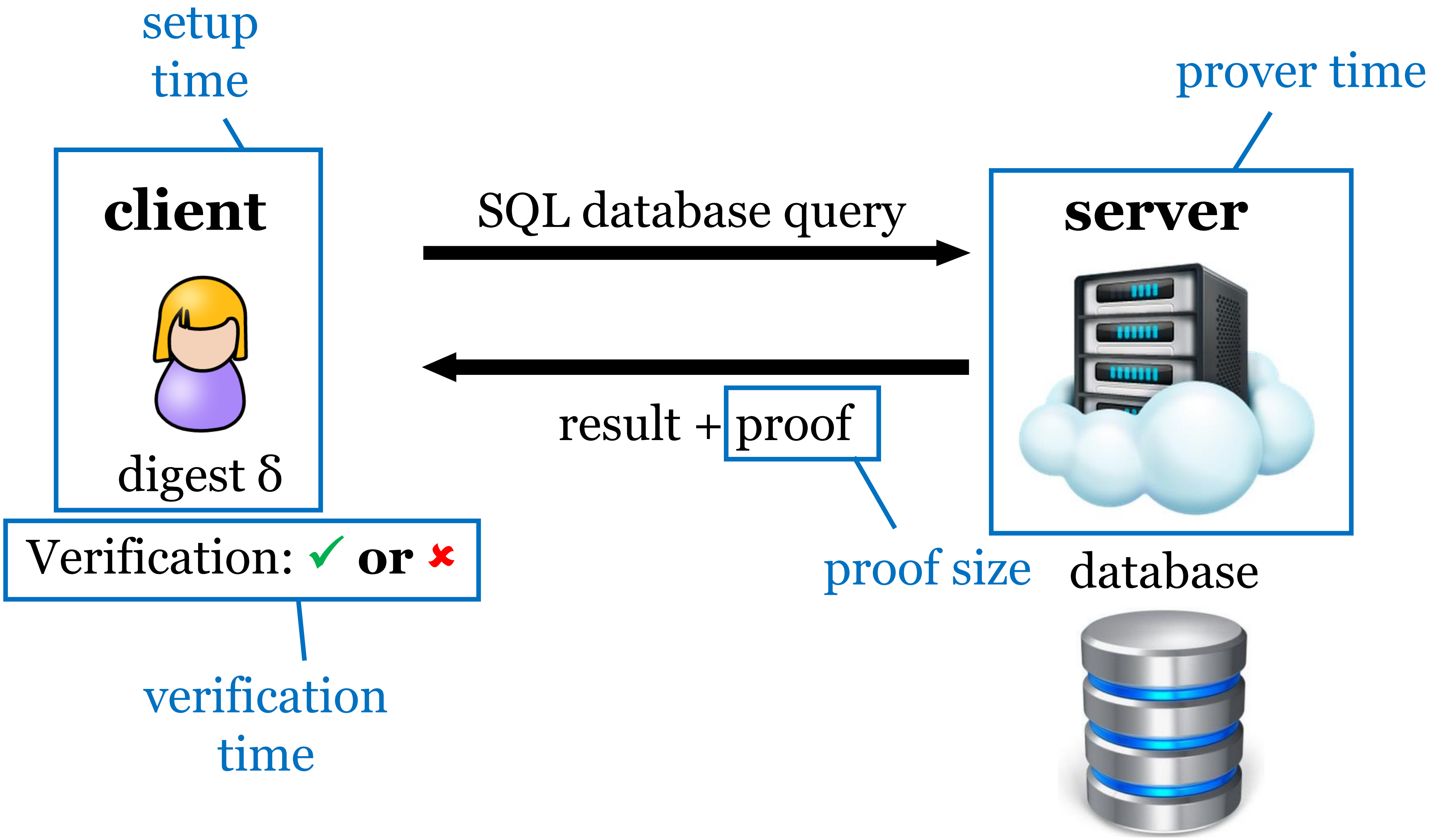
result + proof

digest δ

Verification: ✓ **or** ✘

database

Efficiency Measures of Verifiable Databases



setup time

client

digest δ

Verification: ✓ **or** ✗

verification time

SQL database query

result + proof

proof size

prover time

server

database

Prior Work in Verifiable Databases

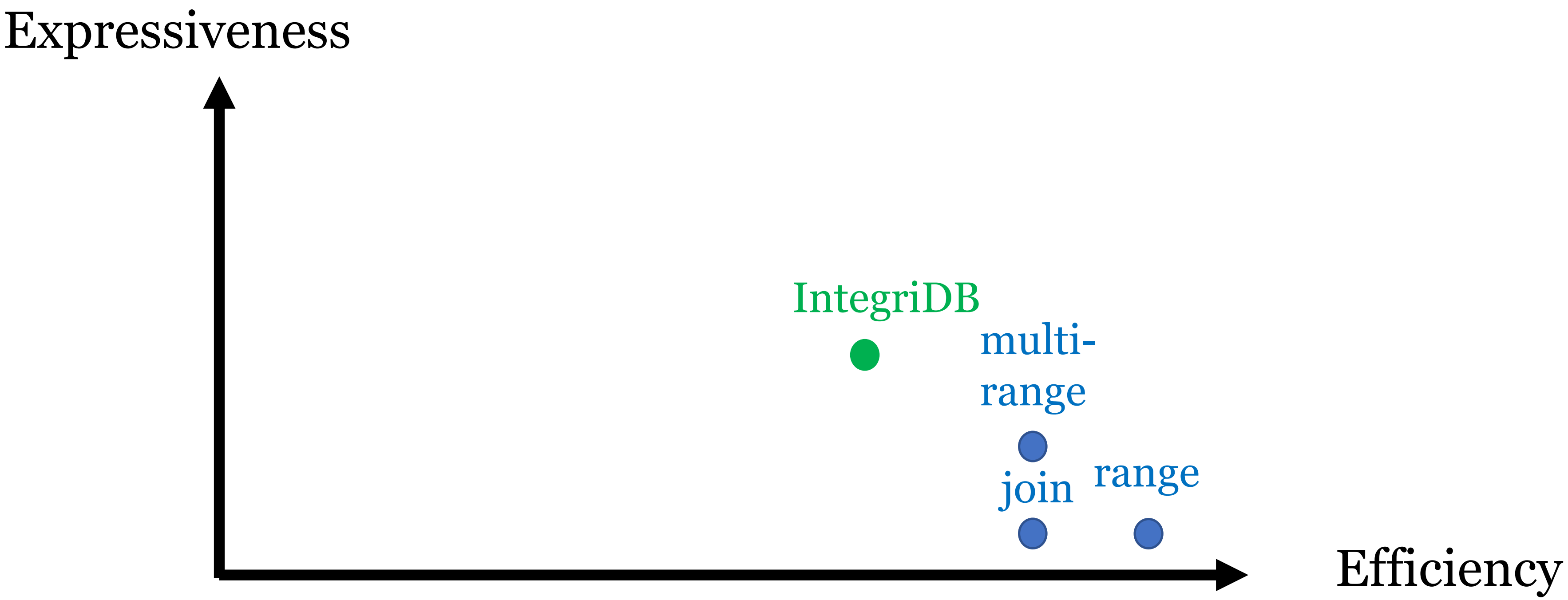
1. Customized Approach (E.g., ADS [Tamassia03])

- Range [LHKR06, MNT06, ...], multi-range [PPT14, ...], join[PJRT05, ...]

✓ Efficient

× Only support limited operations

- IntegriDB [ZKP15]



Prior Work in Verifiable Databases

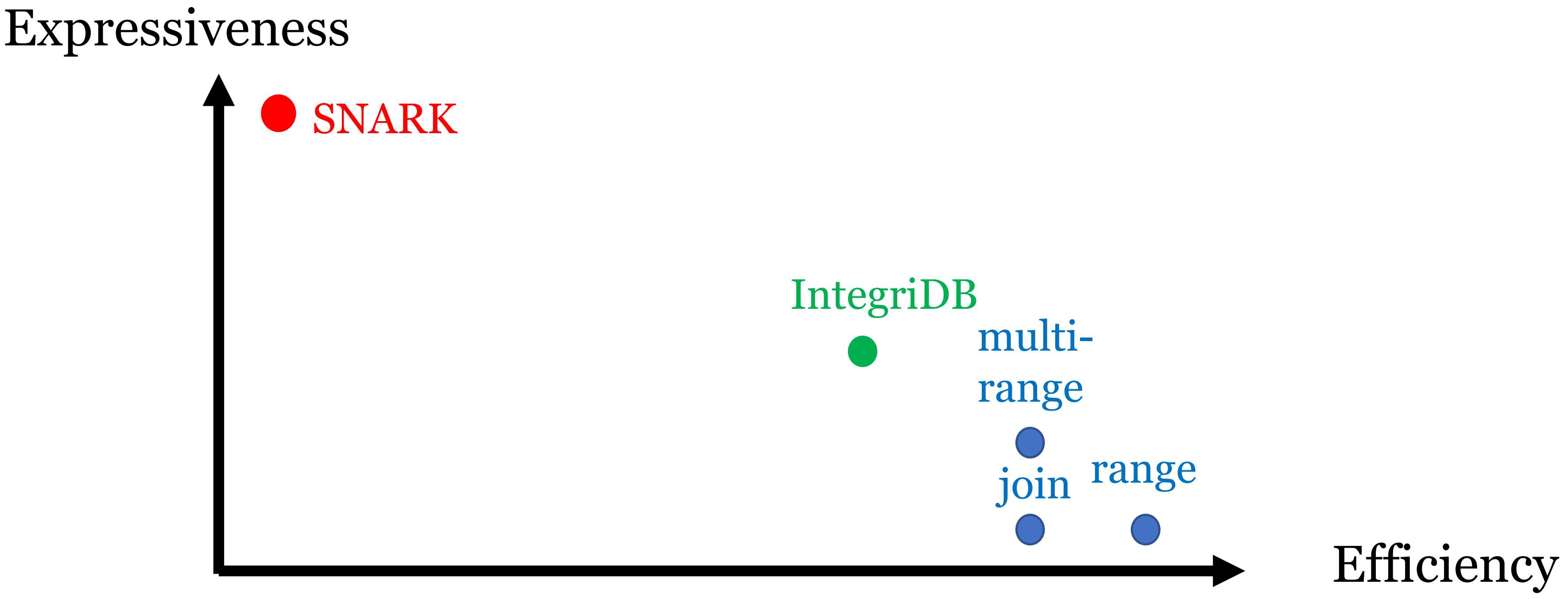
2. Generic Approach (E.g., SNARK [PHGR13, BCGTV13, BFRS+13, …] & PCP [Kilian92, Micali94, ….])

✓ Supports all functions that can be modeled as arithmetic circuits

✓ Constant proof size, fast verification time

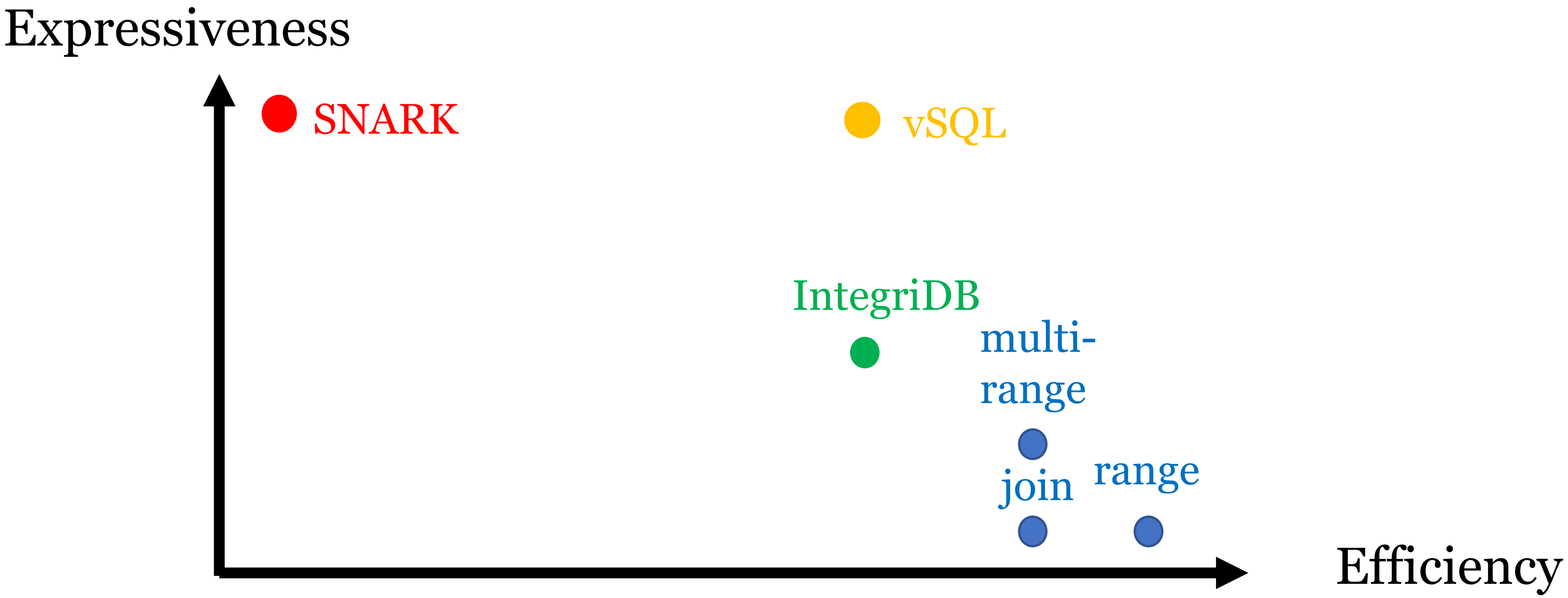
× Large setup time & prover time

× Function specific setup



Our Contribution: vSQL

- Supports arbitrary SQL queries
- Comparable prover time to IntegriDB, faster setup time
- Up to 2 orders of magnitude faster than SNARKs
- No function specific setup



Example

```
SELECT SUM (l_extendedprice * (1 - l_discount)) AS revenue FROM lineitem, part
WHERE
( p_partkey = l_partkey
AND p_brand = ‘Brand#41’
AND p_container IN (‘SM CASE’, ‘SM BOX’, ‘SM PACK’, ‘SM PKG’)
AND l_quantity >= 7 AND l_quantity <= 7 + 10
AND p_size BETWEEN 1 AND 5
AND l_shipmode IN (‘AIR’, ‘AIR REG’)
AND l_shipinstruct = ‘DELIVER IN PERSON’ )
OR
( p_partkey = l_partkey
AND p_brand = ‘Brand#14’
AND p_container IN (‘MED BAG’, ‘MED BOX’,‘MED PKG’, ‘MED PACK’)
AND l_quantity >= 14 AND l_quantity <= 14 + 10
AND p_size BETWEEN 1 AND 10
AND l_shipmode IN (‘AIR’, ‘AIR REG’)
AND l_shipinstruct = ‘DELIVER IN PERSON’ )
OR
( p_partkey = l_partkey
AND p_brand = ‘Brand#23’
AND p_container IN (‘LG CASE’, ‘LG BOX’, ‘LG PACK’, ‘LG PKG’)
AND l_quantity >= 25 AND l_quantity <= 25 + 10
AND p_size BETWEEN 1 AND 15
AND l_shipmode IN (‘AIR’, ‘AIR REG’)
AND l_shipinstruct = ‘DELIVER IN PERSON’ );
```

Query #19 of the TPC-H benchmark
http://www.tpc.org/tpch

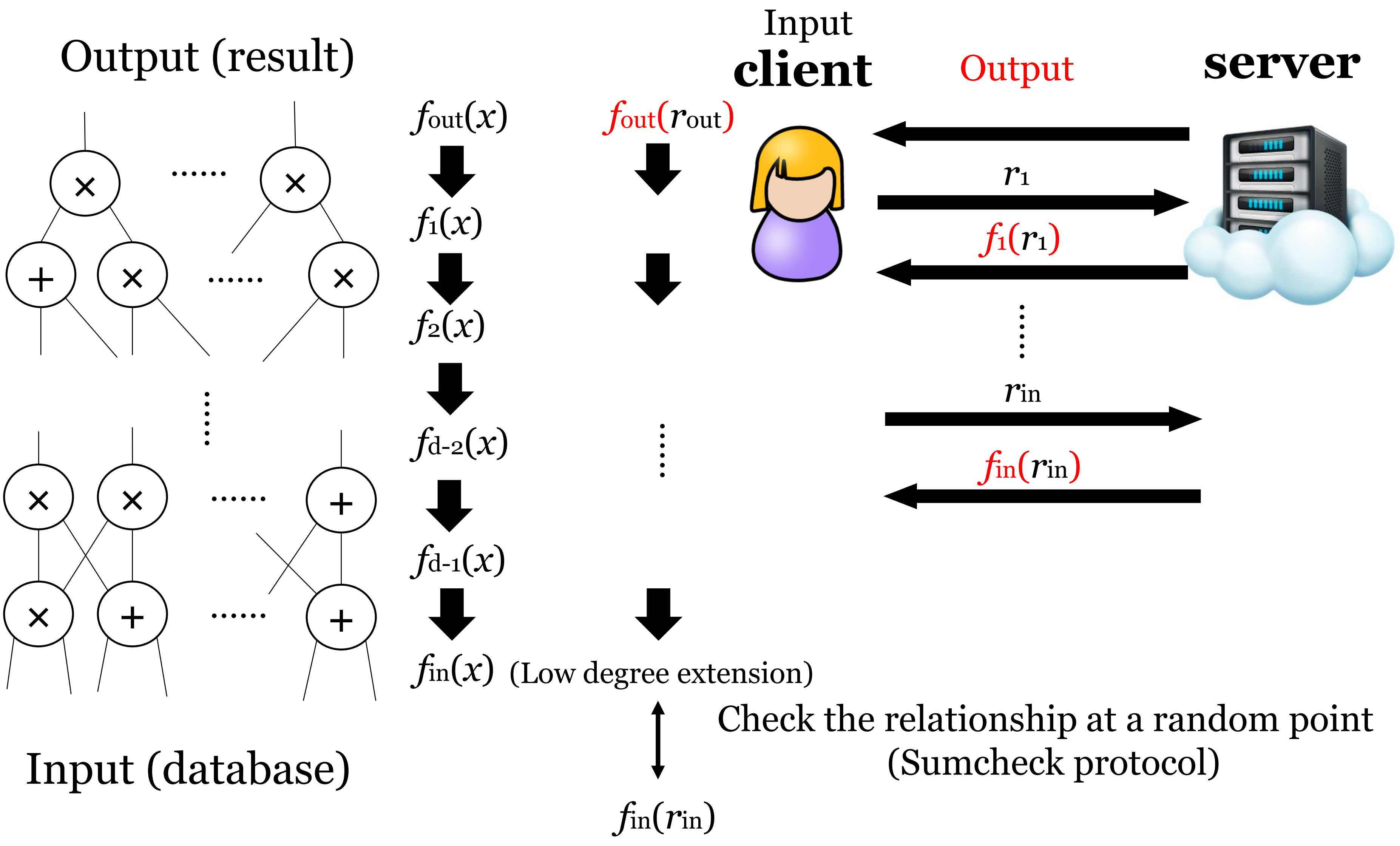
Our Construction

Interactive Proof (IP)[GKR08, CMT12, …]

Example

```
SELECT  SUM (l_extendedprice * (1 - l_discount))  AS  revenue  FROM  lineitem, part
WHERE
( p_partkey = l_partkey
AND p_brand = ‘Brand#41’
AND p_container IN (‘SM CASE’, ‘SM BOX’, ‘SM PACK’, ‘SM PKG’)
AND l_quantity >= 7 AND l_quantity <= 7 + 10
AND p_size BETWEEN 1 AND 5
AND l_shipmode IN (‘AIR’, ‘AIR REG’)
AND l_shipinstruct = ‘DELIVER IN PERSON’ )
OR
( p_partkey = l_partkey
AND p_brand = ‘Brand#14’
AND p_container IN (‘MED BAG’, ‘MED BOX’,‘MED PKG’, ‘MED PACK’)
AND l_quantity >= 14 AND l_quantity <= 14 + 10
AND p_size BETWEEN 1 AND 10
AND l_shipmode IN (‘AIR’, ‘AIR REG’)
AND l_shipinstruct = ‘DELIVER IN PERSON’ )
OR
( p_partkey = l_partkey
AND p_brand = ‘Brand#23’
AND p_container IN (‘LG CASE’, ‘LG BOX’, ‘LG PACK’, ‘LG PKG’)
AND l_quantity >= 25 AND l_quantity <= 25 + 10
AND p_size BETWEEN 1 AND 15
AND l_shipmode IN (‘AIR’, ‘AIR REG’)
AND l_shipinstruct = ‘DELIVER IN PERSON’ );
```

Interactive Proof (IP)[GKR08, CMT12, ...]



Output (result)

$f_{out}(x)$

$f_1(x)$

$f_2(x)$

$f_{d-2}(x)$

$f_{d-1}(x)$

$f_{in}(x)$ (Low degree extension)

Input (database)

$f_{out}(r_{out})$

Input

client

Output

server

r_1

$f_1(r_1)$

r_{in}

$f_{in}(r_{in})$

Check the relationship at a random point
(Sumcheck protocol)

$f_{in}(r_{in})$

Using IP for Verifiable Databases

✓ No setup time

✓ Fast prover time (no crypto operations)

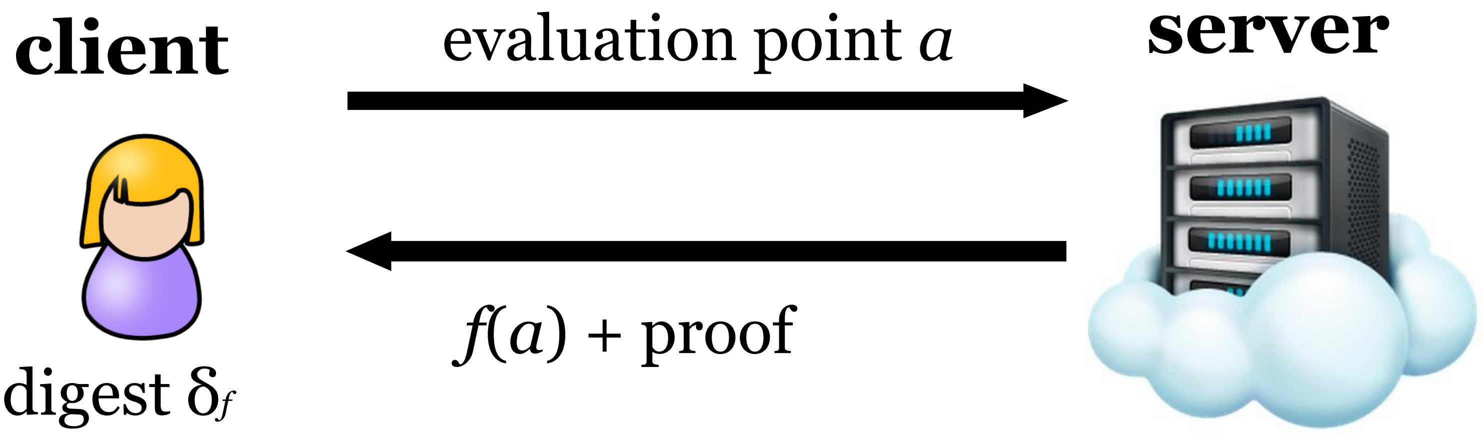
× Storage of the database locally

(Last step: evaluate a polynomial defined by the input at a random point)

Delegating Database to the Server

- Our solution: Verifiable Polynomial Delegation (VPD)

[KZG10, PST13]



client

evaluation point a

server

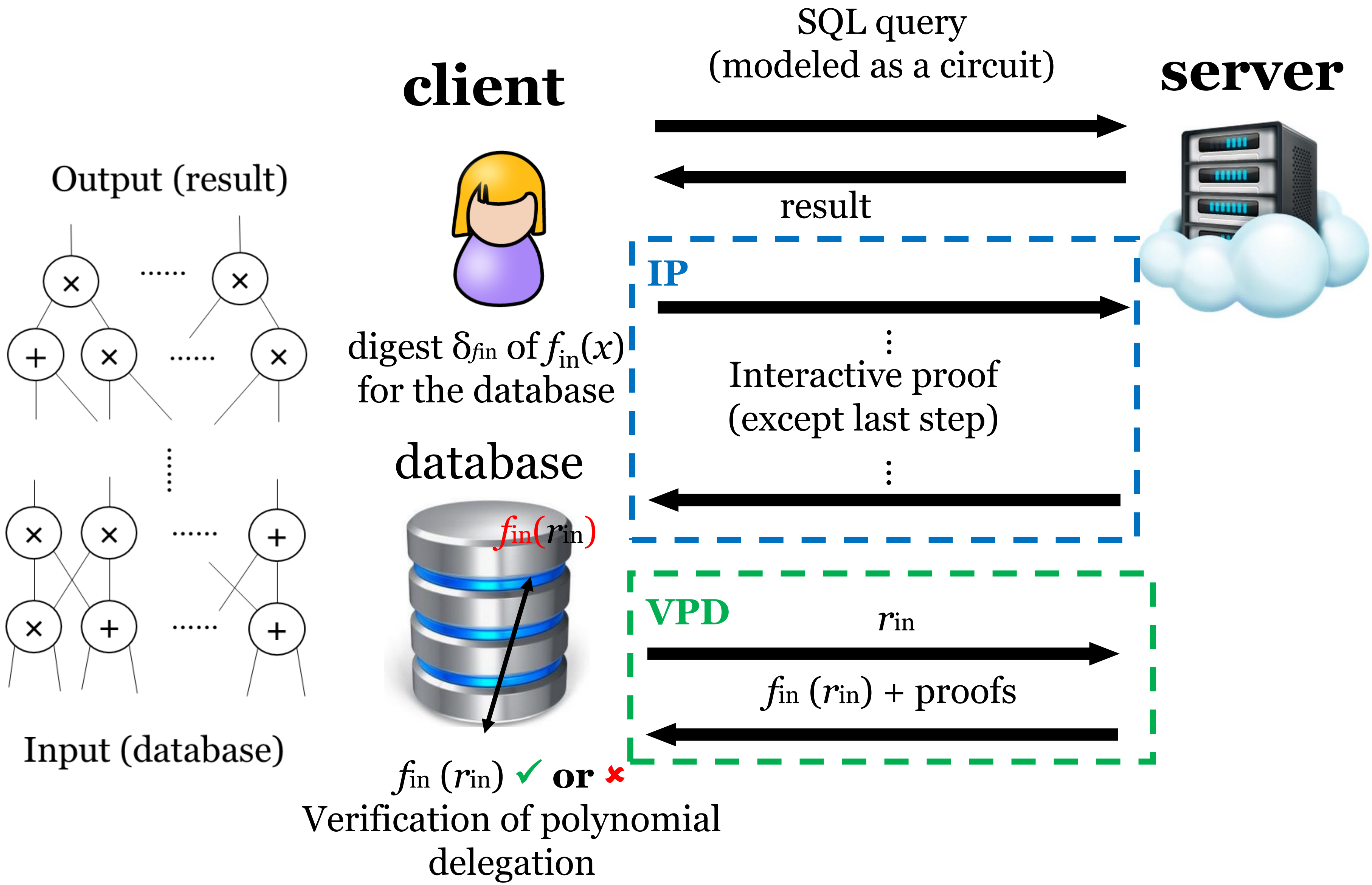
$f(a)$ + proof

digest δ_f (32Bytes)

Verification: ✓ **or** ✘

$f(x)$

vSQL protocol



Output (result)

Input (database)

client

digest $\delta_{f_{in}}$ of $f_{in}(x)$ for the database

database

$f_{in}(r_{in})$

SQL query (modeled as a circuit)

server

result

IP

Interactive proof (except last step)

VPD

r_{in}

$f_{in}(r_{in})$ + proofs

$f_{in}(r_{in})$ ✓ **or** ✗

Verification of polynomial delegation

Using IP for Verifiable Databases

✓ No setup time

✓ Fast prover time (no crypto operations)

× ~~Storage of the database locally~~

~~(Last step: evaluate a polynomial defined by the input at a random point)~~

Verifying Computations in NP

- Some functions are hard to compute using arithmetic circuits

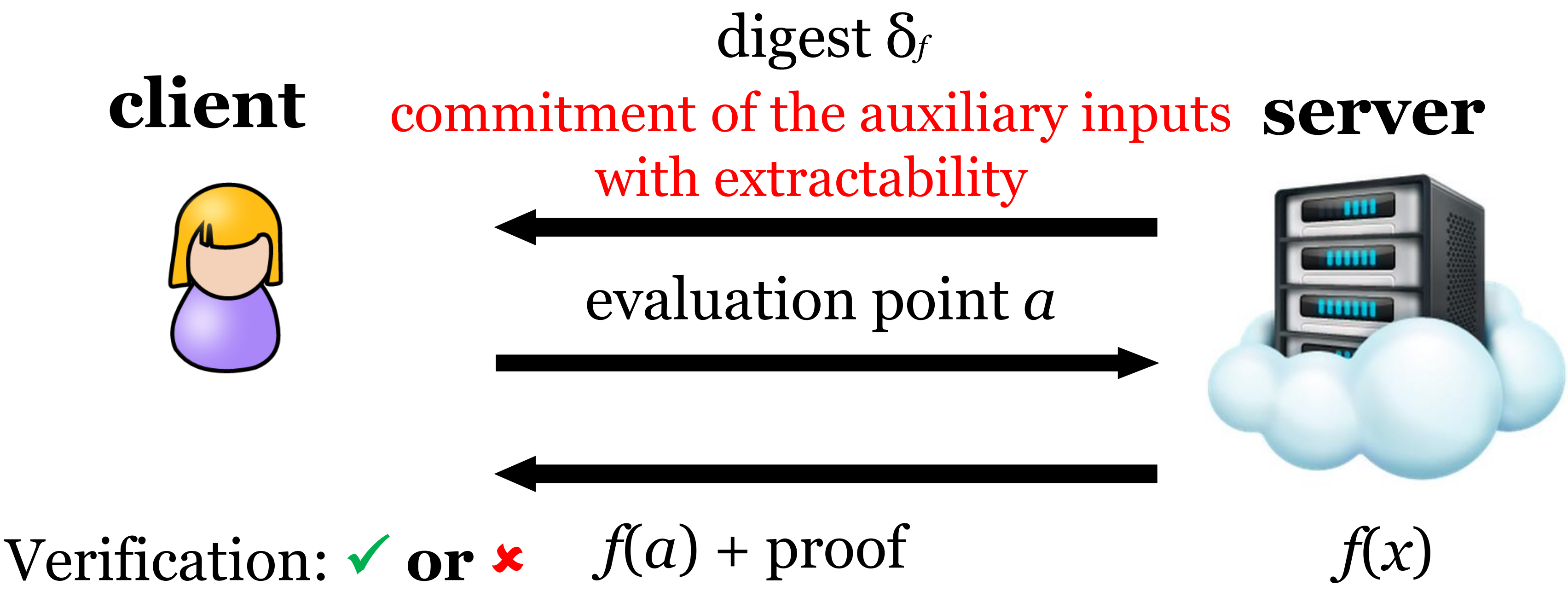
  E.g., Integer division $a \div b$

- They are easy to verify with inputs from the server: $a = q \times b + r$

- Interactive Proof does not support auxiliary input

Verifying Computations in NP

- Our solution: Extractable Verifiable Polynomial Delegation (VPD)



client

server

digest δ_f

commitment of the auxiliary inputs with extractability

evaluation point a

$f(a)$ + proof

$f(x)$

Verification: ✓ **or** ✘

Result: extending IP (GKR, CMT etc.) to NP computations without using FHE [CKLR11, ...]

vSQL

✓ Setup only for the database, not for queries

✓ Faster prover time

(crypto operations is only linear to the database size, does not depend on the circuit size)

✓ Supports auxiliary inputs

✓ Expressive SQL updates (details in the paper)

Experimental Results

Comparison with Prior Work

Queries and database: TPC-H benchmark
Database size: 6 million rows × 13 columns (2.8GB) in the largest table.

Query #19		**IntegriDB**	**SNARK**	**vSQL**
	Setup	7 hours	100 hours*	0.4 hour
	Prover	1.8 hours	54 hours*	1.3 hours
	Verification	232 ms	6 ms	148 ms
	Communication	184 KB	0.3 KB	28 KB

Follow-up:
4× faster!

Update

Query #15: create a new table on the fly by range and sum

Old table: 2.8GB new table: 1.7MB

Prover	**Verification**	**Communication**
0.5 hour	85ms	85.7KB

Summary of vSQL

- vSQL: Verifiable Polynomial Delegation + Interactive Proof
  - ➢Comparable efficiency, better expressiveness compared to customized VC
  - ➢Up to 2 orders of magnitude faster compared to SNARKs
  - ➢Setup only for database, no query dependent setup

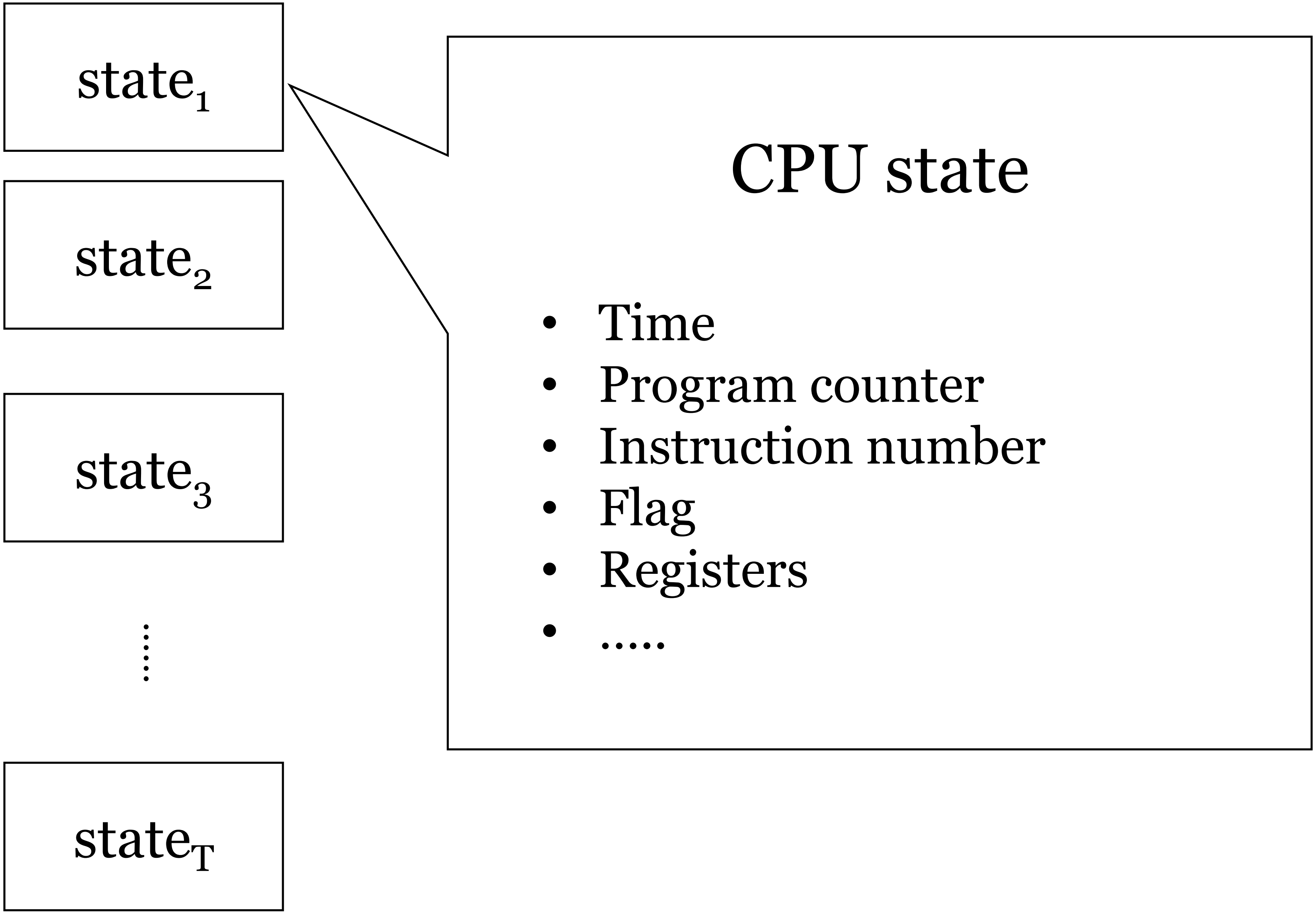
One Preprocessing to Rule Them All: Verifiable Computation with Circuit-Independent Preprocessing and Applications to Verifiable RAM Programs

- Interactive argument for NP, with function independent preprocessing
- Apply to verifiable RAM computations
- Theorem: Prover time linear in #of CPU steps T

  vs. quasi-linear using SNARKs [BCTV14]
- 8× faster prover time, 120× smaller memory consumption, up to 2 million CPU steps

RAM to Circuit Reduction [BCTV14]

By time:



state_1

state_2

state_3

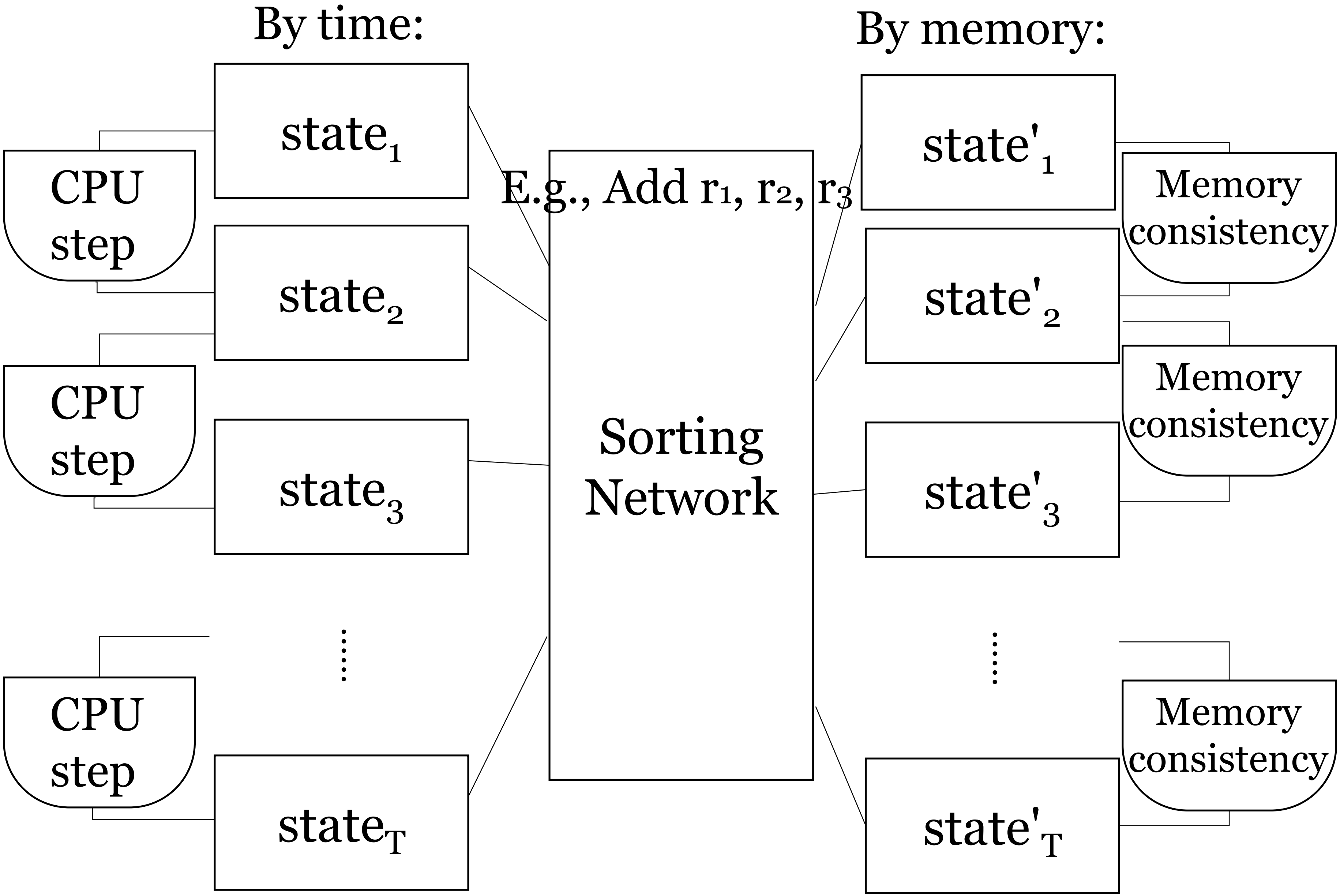
⋮

state_T

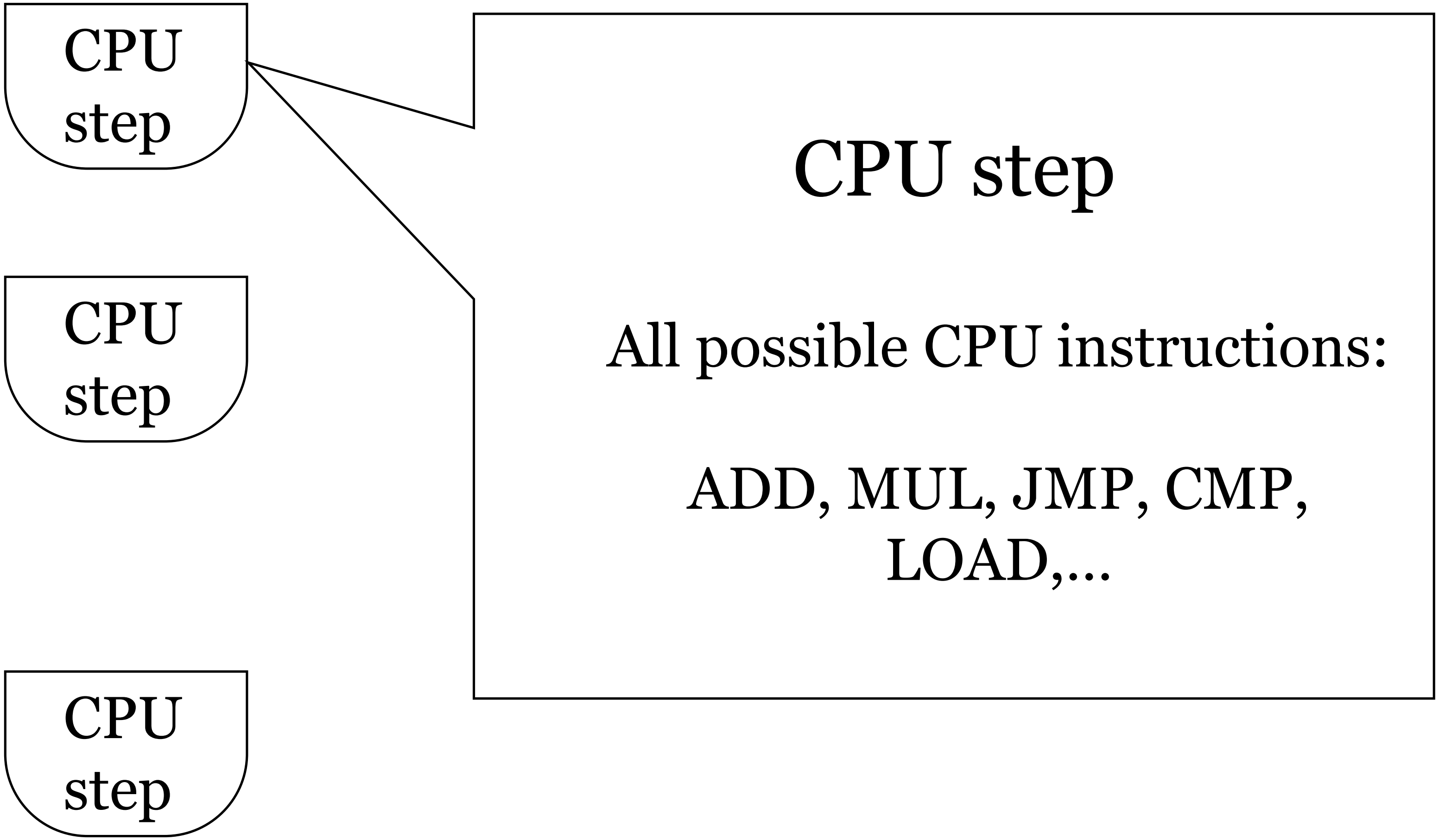
CPU state

- Time
- Program counter
- Instruction number
- Flag
- Registers
- …..

RAM to Circuit Reduction [BCTV14]



Inefficiency: Preprocessing



CPU step

CPU step

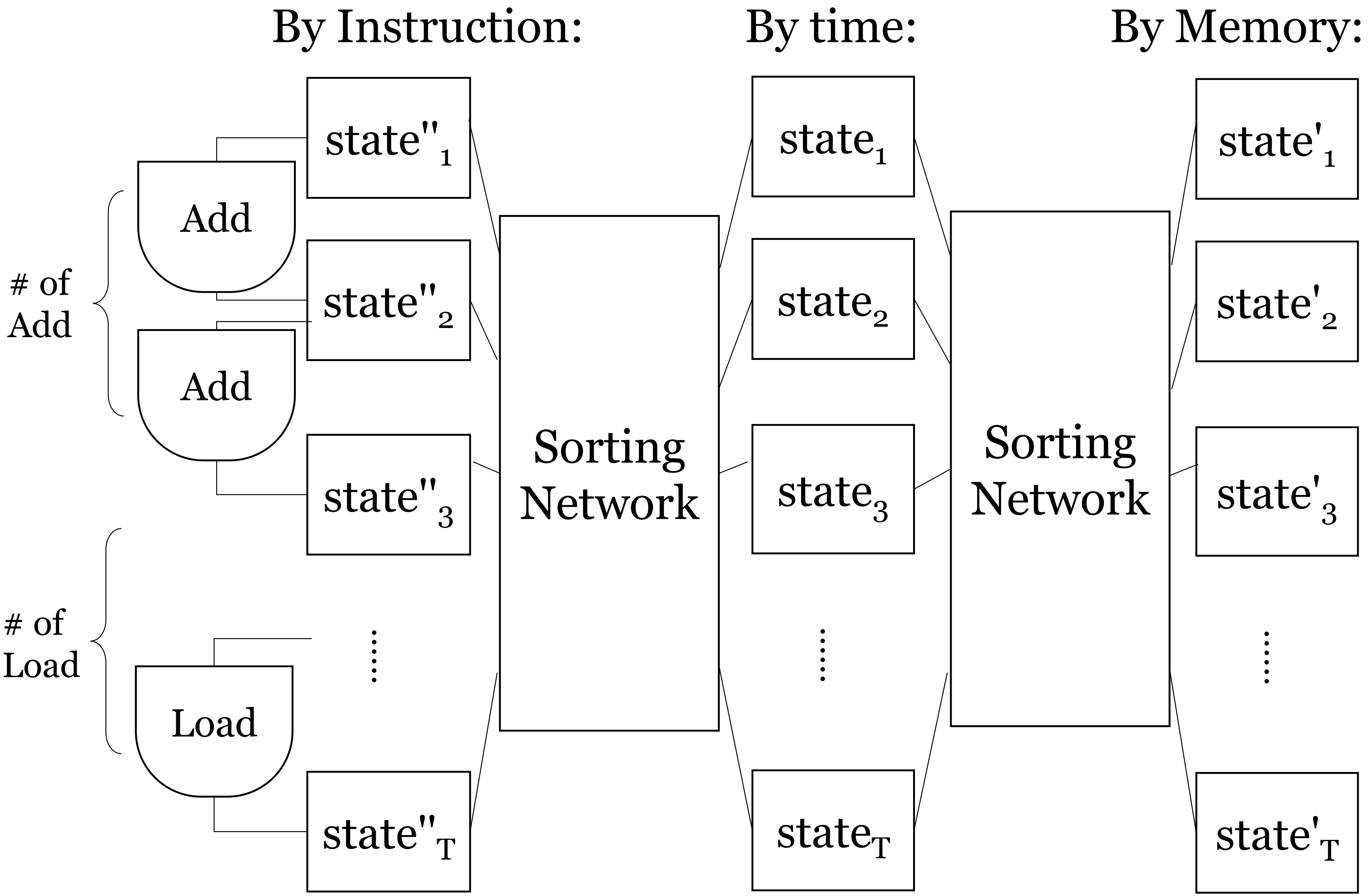
CPU step

CPU step

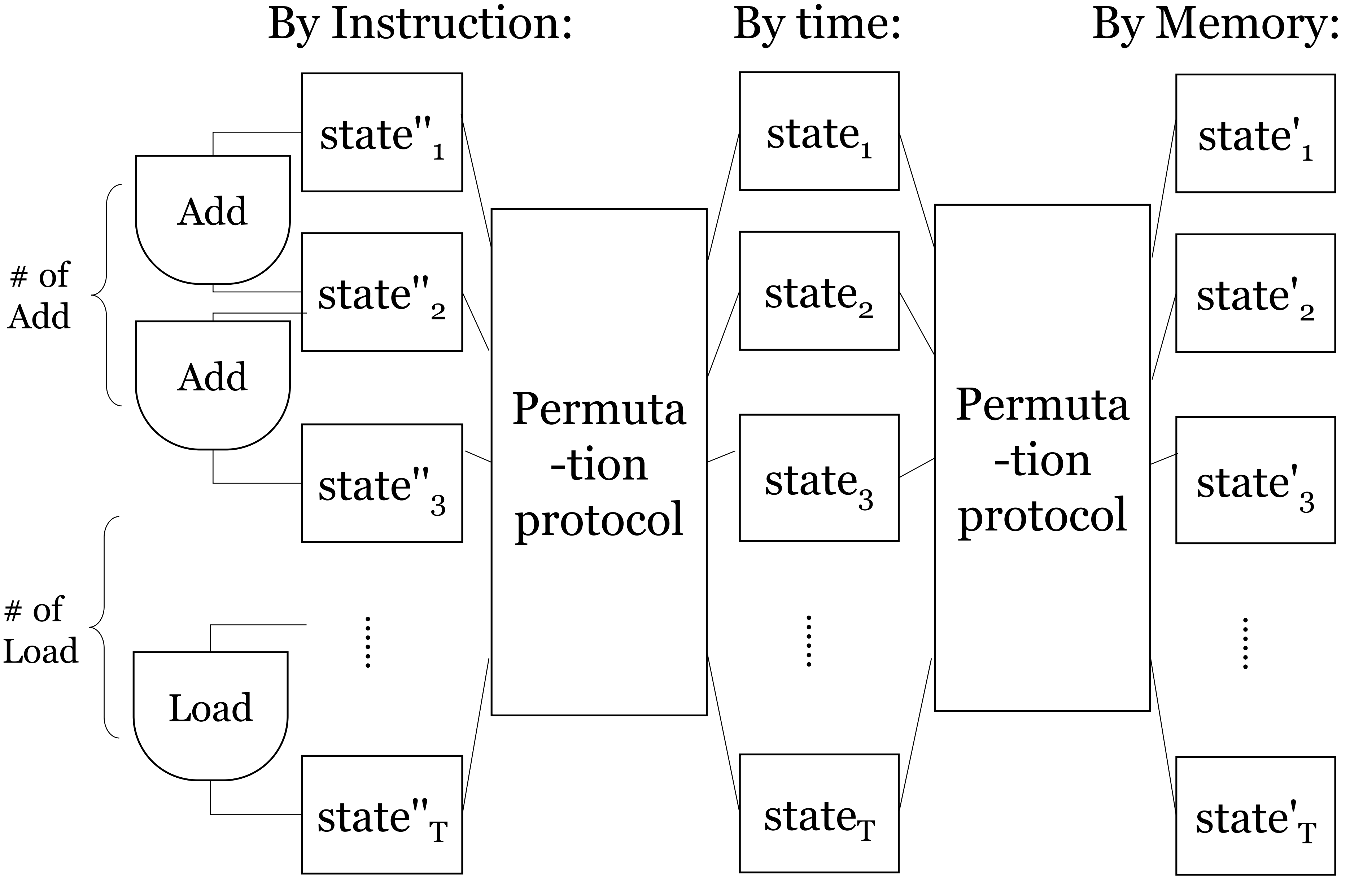
All possible CPU instructions:

ADD, MUL, JMP, CMP, LOAD,...

Our New RAM to Circuit Reduction



Our New RAM to Circuit Reduction



Our New Verifiable RAM

- 8× faster prover time
- 120× smaller memory consumption

  (up to 2 million CPU steps)
- Prover time linear in #of CPU steps T
- One preprocessing for both RAM and circuit

Summary

Verifiable Polynomial Delegation + Interactive Proof

- vSQL, verifiable databases
- Verifiable RAM

Ongoing work:

- Verifiable RAM with states
- Zero-knowledge with applications to crypto-currencies

Thank you!!!
Q&A