vSQL: Verifying Arbitrary SQL Queries over Dynamic Outsourced Databases

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

client

digest $\delta$

Verification: ✓ or ✗

database

SQL database query

result + proof

server
Efficiency Measures of Verifiable Databases

- **client**
  - digest $\delta$
  - Verification: ✔️ or ✗
  - verification time

- **server**
  - SQL database query
  - result + proof
  - proof size
  - prover time

- **setup time**
- **server time**
Prior Work in Verifiable Databases

1. Customized Approach (E.g., ADS [Tamassia03])
   • Range [LHKRo6, MNTo6, ...], multi-range [PPT14, ...], join[PJRT05, ...]
     ✓ Efficient
     × Only support limited operations
   • IntegriDB [ZKP15]

Expressiveness
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

Expressiveness

Expressiveness vs. Efficiency

- SNARK
- IntegriDB
- multi-range
- join
- range
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

1. **SELECT SUM (l_extendedprice * (1 - l_discount)) AS revenue FROM lineitem, part WHERE**
2. (p_partkey = l_partkey
3. AND p_brand = ‘Brand#41’
5. AND l_quantity >= 7 AND l_quantity <= 7 + 10
6. AND p_size BETWEEN 1 AND 5
7. AND l_shipmode IN (‘AIR’, ‘AIR REG’)
8. AND l_shipinstruct = ‘DELIVER IN PERSON’ )
9. OR
10. (p_partkey = l_partkey
11. AND p_brand = ‘Brand#14’
12. AND p_container IN (‘MED BAG’, ‘MED BOX’, ‘MED PKG’, ‘MED PACK’)
13. AND l_quantity >= 14 AND l_quantity <= 14 + 10
14. AND p_size BETWEEN 1 AND 10
15. AND l_shipmode IN (‘AIR’, ‘AIR REG’)
16. AND l_shipinstruct = ‘DELIVER IN PERSON’ )
17. OR
18. (p_partkey = l_partkey
19. AND p_brand = ‘Brand#23’
20. AND p_container IN (‘LG CASE’, ‘LG BOX’, ‘LG PACK’, ‘LG PKG’)
21. AND l_quantity >= 25 AND l_quantity <= 25 + 10
22. AND p_size BETWEEN 1 AND 15
23. AND l_shipmode IN (‘AIR’, ‘AIR REG’)
24. 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

1. SELECT SUM (l_extendedprice * (1 - l_discount)) AS revenue FROM lineitem, part
   WHERE
   2. ( p_partkey = l_partkey
   3. AND p_brand = 'Brand#41'
   4. AND p_container IN ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
   5. AND l_quantity >= 7 AND l_quantity <= 7 + 10
   6. AND p_size BETWEEN 1 AND 5
   7. AND l_shipmode IN ('AIR', 'AIR REG')
   8. AND l_shipinstruct = 'DELIVER IN PERSON' )
   9. OR
   10. ( p_partkey = l_partkey
   11. AND p_brand = 'Brand#14'
   12. AND p_container IN ('MED BAG', 'MED BOX','MED PKG', 'MED PACK')
   13. AND l_quantity >= 14 AND l_quantity <= 14 + 10
   14. AND p_size BETWEEN 1 AND 10
   15. AND l_shipmode IN ('AIR', 'AIR REG')
   16. AND l_shipinstruct = 'DELIVER IN PERSON' )
   17. OR
   18. ( p_partkey = l_partkey
   19. AND p_brand = 'Brand#23'
   20. AND p_container IN ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
   21. AND l_quantity >= 25 AND l_quantity <= 25 + 10
   22. AND p_size BETWEEN 1 AND 15
   23. AND l_shipmode IN ('AIR', 'AIR REG')
   24. AND l_shipinstruct = 'DELIVER IN PERSON' );
Interactive Proof (IP) \cite{GKR08, CMT12, ...}

\begin{itemize}
  \item Input (database)
  \item Output (result)
  \item Client
  \item Server
\end{itemize}

\begin{align*}
  f_{\text{out}}(x) & \quad \downarrow \\
  f_1(x) & \quad \downarrow \\
  f_2(x) & \quad \downarrow \\
  \vdots & \quad \downarrow \\
  f_{d-1}(x) & \quad \downarrow \\
  f_{\text{in}}(x) & \quad \downarrow \\
  \text{(Low degree extension)} & \quad \downarrow \\
  f_{\text{in}}(r_{\text{in}}) & \quad \uparrow \\
\end{align*}

Check the relationship at a random point (Sumcheck protocol)

\begin{align*}
  f_{\text{out}}(r_{\text{out}}) & \quad \downarrow \\
  f_{\text{in}}(r_{\text{in}}) & \quad \uparrow \\
  r_1 & \quad \downarrow \\
  \vdots & \quad \downarrow \\
  r_{\text{in}} & \quad \uparrow \\
\end{align*}
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]

- Evaluation point $a$
- $f(a) + proof$
- Verification: ✓ or ✗
vSQL protocol

Output (result)

Input (database)

SQL query (modeled as a circuit)

Interactive proof (except last step)

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)

**Evaluation point:** $f(a)$ + proof

**Commitment of the auxiliary inputs with extractability:** digest $\delta_f$

**Verification:** ✓ or ✗ $f(a)$ + proof

**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 $\times$ 13 columns (2.8GB) in the largest table.

<table>
<thead>
<tr>
<th>Query #19</th>
<th>Setup</th>
<th>IntegriDB</th>
<th>SNARK</th>
<th>vSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7 hours</td>
<td>100 hours*</td>
<td>0.4 hour</td>
<td></td>
</tr>
<tr>
<td>Prover</td>
<td>1.8 hours</td>
<td>54 hours*</td>
<td>1.3 hours</td>
<td></td>
</tr>
<tr>
<td>Verification</td>
<td>232 ms</td>
<td>6 ms</td>
<td>148 ms</td>
<td></td>
</tr>
<tr>
<td>Communication</td>
<td>184 KB</td>
<td>0.3 KB</td>
<td>28 KB</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Prover</th>
<th>Verification</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5 hour</td>
<td>85ms</td>
<td>85.7KB</td>
</tr>
</tbody>
</table>
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]

By time:
- $state_1$
- $state_2$
- $state_3$
- ... 
- $state_T$

E.g., Add $r_1$, $r_2$, $r_3$

By memory:
- $state'_1$
- $state'_2$
- $state'_3$
- ... 
- $state'_T$

Memory consistency
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

By Instruction:

- $\text{state}''_1$
- $\text{state}''_2$
- $\text{state}''_3$
- ... (more states)

By time:

- $\text{state}_1$
- $\text{state}_2$
- $\text{state}_3$
- ... (more states)

By Memory:

- $\text{state}'_1$
- $\text{state}'_2$
- $\text{state}'_3$
- ... (more states)

Diagram:

- Add
- Load
- Sorting Network
- # of Add
- # of Load
Our New RAM to Circuit Reduction

By Instruction:

- Add
  - state''_1
  - state''_2
  - state''_3

Permutation protocol

By time:

- state_1
- state_2
- state_3

By Memory:

- state'_1
- state'_2
- state'_3

By time:

- state_T

By Memory:

- state'_T
Our New Verifiable RAM

- $8 \times$ faster prover time
- $120 \times$ 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