ZKSQL: Verifiable and Efficient Query Evaluation with Zero-Knowledge Proofs

VLDB ’23

Xiling Li
Northwestern University
xiling.li@northwestern.edu

Chenkai Weng
Northwestern University
ckweng@u.northwestern.edu

Yongxin Xu
Northwestern University
yongxinxu2022@u.northwestern.edu

Xiao Wang
Northwestern University
wangxiao@northwestern.edu

Jennie Rogers
Northwestern University
jennie@northwestern.edu

2024. 07. 18

서울대 분산시스템연구실

석사과정 문보설

Motivation

- **<u>Motivation 1</u>**: Data providers could forge the data
  - Motivating example

    U.S. News Dropped Columbia's Ranking, but Its Own Methods Are Now Questioned

    After doubt about its data, the university dropped to No. 18 from No. 2. But now many are asking, can the rating system be that easily manipulated?

  - DoE cannot verify statistics provided by universities with conventional DBMS

Motivation

- **<u>Motivation 1</u>**: Data providers could forge the data
  - Motivating example

    U.S. News Dropped Columbia's Ranking, but Its Own Methods Are Now Questioned

    After doubt about its data, the university dropped to No. 18 from No. 2. But now many are asking, can the rating system be that easily manipulated?

  - DoE cannot verify statistics provided by universities with conventional DBMS

- **<u>Motivation 2</u>**: Data owners don't want to reveal private information

Motivation

- **Motivation 1**: Data providers could forge the data
  - Motivating example

    U.S. News Dropped Columbia's Ranking, but Its Own Methods Are Now Questioned

    After doubt about its data, the university dropped to No. 18 from No. 2. But now many are asking, can the rating system be that easily manipulated?

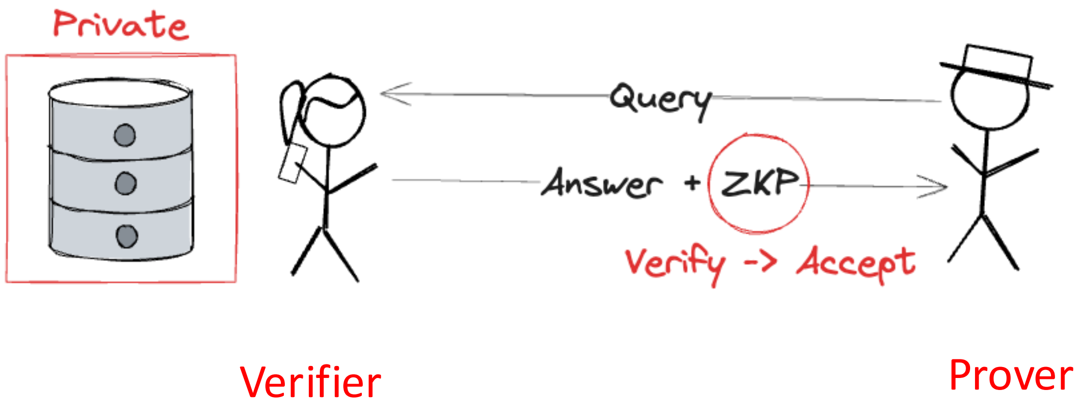
  - DoE cannot verify statistics provided by universities with conventional DBMS

- **Motivation 2**: Data owners don't want to reveal private information

> How can users access reliable statistics without compromising privacy?
>
> → Verifiable, Privacy-preserving Querying

ZKSQL: Zero-knowledge proofs over SQL

- **Goal**
  - Construct authenticated query answer without divulging private input data



ZKSQL: Zero-knowledge proofs over SQL

- <u>**Contributions**</u>
  - **First work** for ad-hoc SQL queries with ZKP
  - **Set-based protocols** for optimization (faster than the baseline*)
  - Experimental results on **TPC-H benchmark** demonstrate ZKSQL's speedup of up to **two orders of magnitude** over the baseline*

*Baseline : Circuit-only implementation

Preliminaries

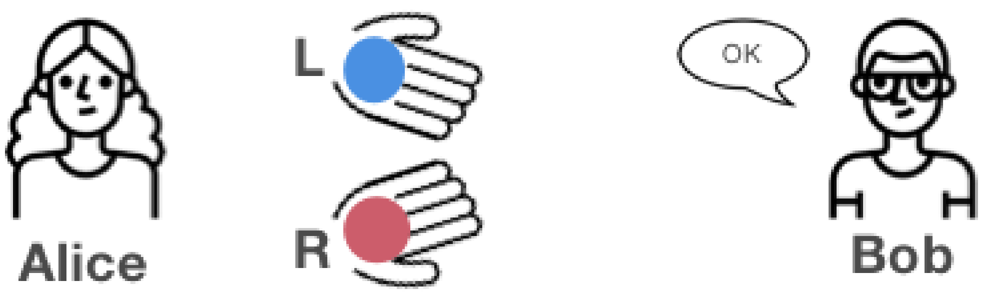
- **<u>Zero-knowledge proof</u>**
  - The prover P **convinces the verifier** V that the statement is true **without revealing any additional information**
  - The statement: Prover's result from some computation are correct and complete

Preliminaries

- **<u>Zero-knowledge proof</u>**
  - The prover P **convinces the verifier** V that the statement is true **without revealing any additional information**
  - The statement: Prover's result from some computation are correct and complete

Statement : Bob is not blinded

Secret : Color of the ball

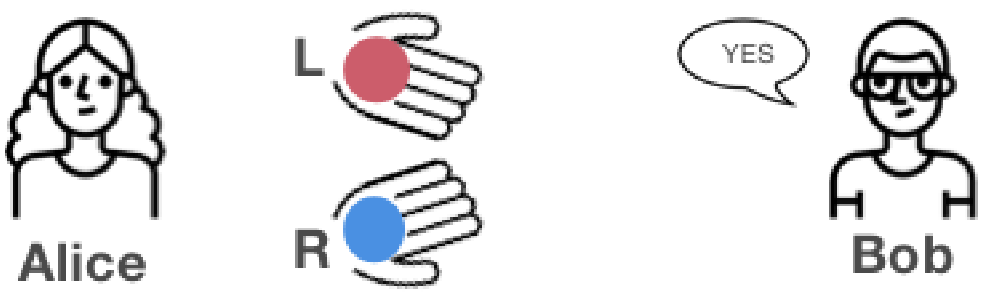


Verifier

Prover

Preliminaries

- **<u>Zero-knowledge proof</u>**
  - The prover P **convinces the verifier** V that the statement is true **without revealing any additional information**
  - The statement: Prover's result from some computation are correct and complete

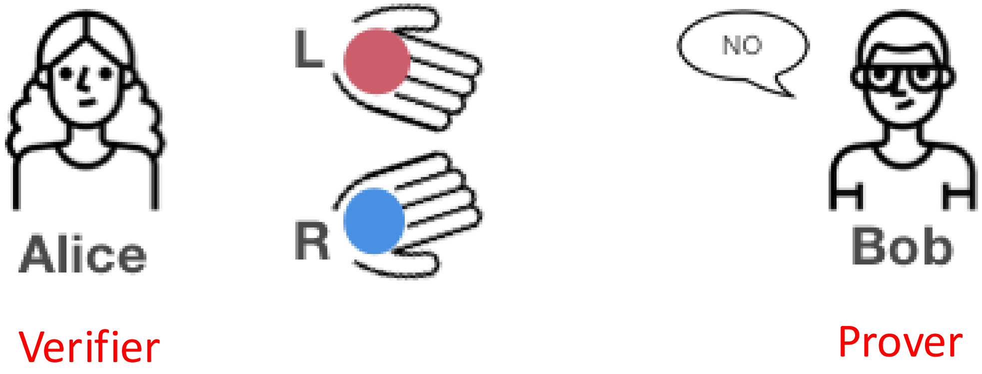


Verifier

Prover

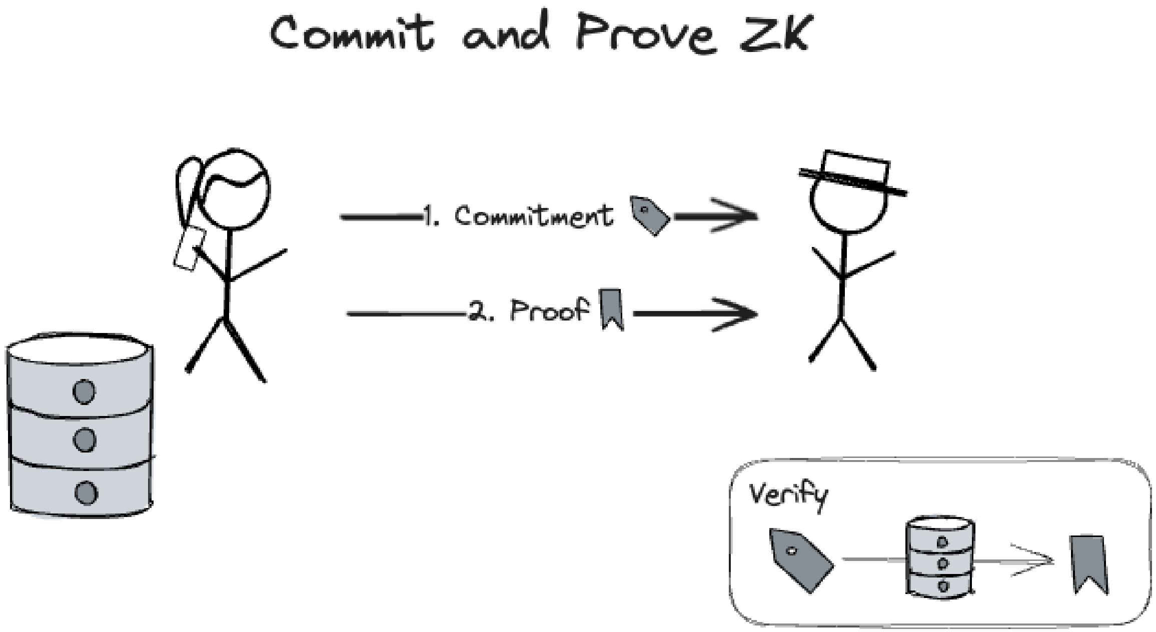
Preliminaries

- **Zero-knowledge proof**
  - The prover P **convinces the verifier** V that the statement is true **without revealing any additional information**
  - The statement: Prover's result from some computation are correct and complete



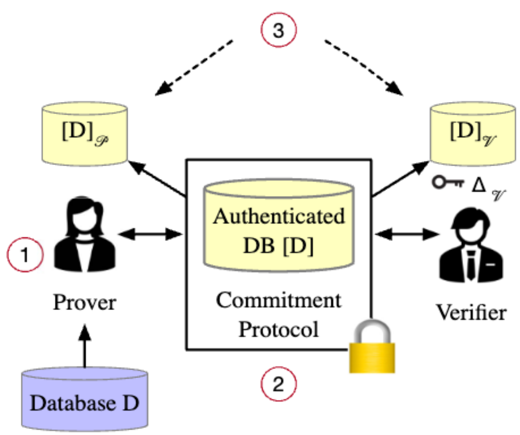
Preliminaries

- **<u>Commit and Prove ZK</u>**
  - Commitment : hide and bind the original data
  - Verifier can confirm that the answer is calculated over the data committed before

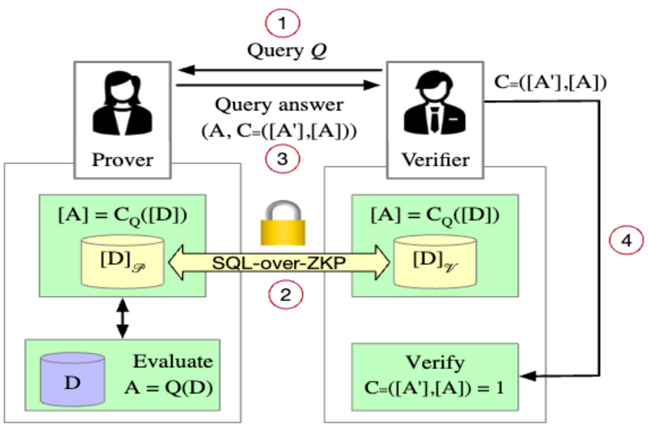


ZKP in Query Evaluation

- **Workflow**
  - **1) Setup** : The engine sets up the commitments over which we will evaluate our zero-knowledge proof, [D] - (fig (a))
  - **2) Proof generation & Verification :** P and V interactively verify the answer to one or more SQL queries with respect to [D] - (fig (b))



(a) Private database commitment.

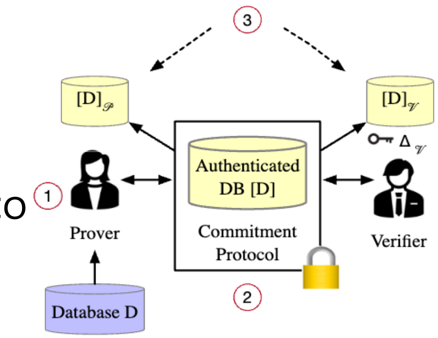


(b) Authenticated querying over ZK proofs.

Setup

- **Setup**
  - Generate **commitment [D]** for Database D
  - V can confirm multiple query answers refer to the same dataset D

- **Workflow**
  1) **Step 1:** P input its private database D to commitment protoco
  2) **Step 2:** P and V generate the tags one for each bit in D
  3) **Step 3:** P holds $[D]_P$, V holds $[D]_V$ and authentication key Δ_V

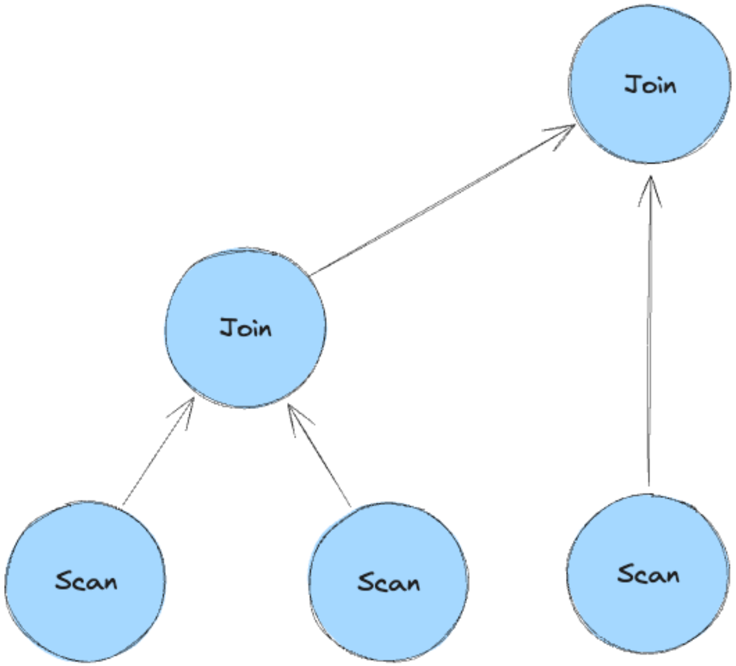


(a) Private database commitment.

- The relationship of $[D]_P$ and $[D]_V$: $[\boldsymbol{D}]_{\boldsymbol{P}} = [\boldsymbol{D}]_{\boldsymbol{V}} + \boldsymbol{D} \cdot \Delta_{\boldsymbol{V}}$

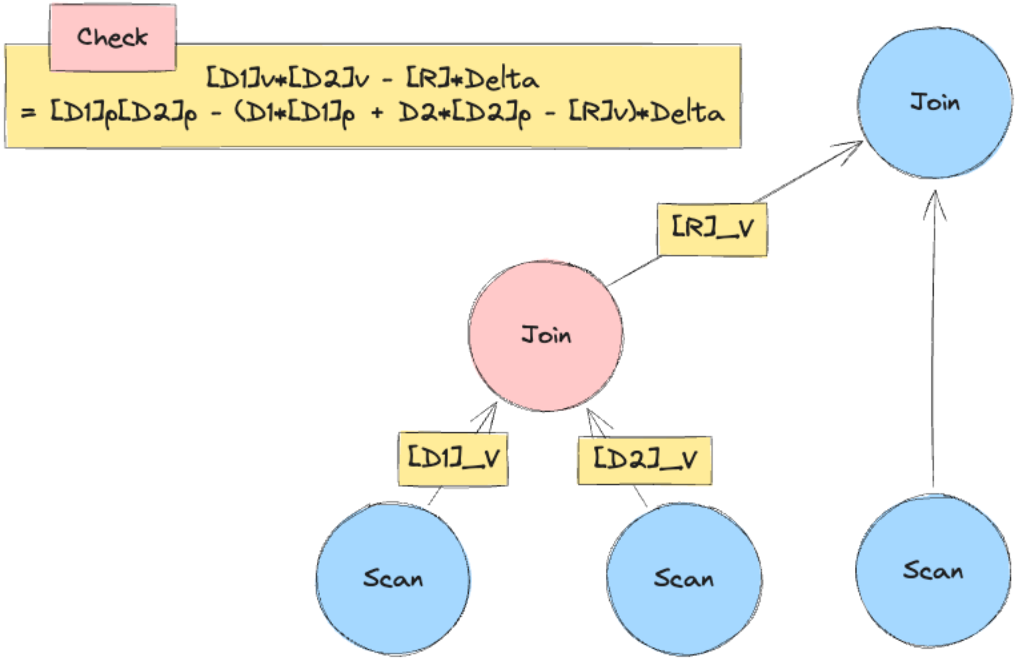
Query Evaluation

- **Workflow**
  1) V sends a SQL statement over the D
  2) ZKSQL parses the SQL into DAG of ZKSQL operators
  3) P and V make commitment of the output
  4) Verify the correctness of input and evaluation with each operator in DAG



Query Evaluation

- **Workflow**
  1) V sends a SQL statement over the D
  2) ZKSQL parses the SQL into DAG of ZKSQL operators
  3) P and V make commitment of the output
  4) Verify the correctness of input and evaluation with each operator in DAG



Set-based optimization

- **Idea**: Decoupling operator evaluation from proving
  - Circuit proves the correctness of evaluation process (Expansive)

    → We need correctness of the result

```
pragma circom 2.0.0;

template Sort4() {
    signal input values[4];
    signal output sorted[4];

    component selectors[4][4];
    component min[4];
    component isUsed[4];

    // Initialize isUsed to 0
    for (var i = 0; i < 4; i++) {
        isUsed[i] <-- 0;
    }

    for (var i = 0; i < 4; i++) {
        for (var j = 0; j < 4; j++) {
            selectors[i][j] = IsZero();
            selectors[i][j].in[0] <== isUsed[j];
            selectors[i][j].in[1] <== values[j] - min[i].out;
        }

        min[i] = Min(4);
        for (var j = 0; j < 4; j++) {
            min[i].in[j] <== selectors[i][j].out * values[j] + (1 - selectors[i][j].out) * 99999;
        }

        for (var j = 0; j < 4; j++) {
            isUsed[j] <== isUsed[j] + selectors[i][j].out;
            require(isUsed[j] <= 1);
        }
        sorted[i] <== min[i].out;
    }
}

component main {public [input]} = Sort4();
```

Set-based operator

- **Example: Sort**
  - P computes intermediate result by local computation(sort) on plaintext, T
  - **Circuit** checks if the adjacent ones in T satisfy the sort definition (Ti < Ti+1) **O(n)**
  - **Set equality operation** checks if [T] contains **exactly same rows** as input of sort [R] **O(n)**

Set-based operator

- **<u>Equality</u>**
  - **Strawman**
  - n: # of tuples in table R and S

1) Circuit sort R and S
2) Compare each tuples in R and S

O(nlogn)
Computation and
Communication

Set-based operator

- **<u>Equality</u>**
  - **Polynomial Identity Testing**
  - n: # of tuples in table R and S

1) V samples a uniform $\alpha \leftarrow F_{2^{128}}$ and send it to P
2) P and V compute and open the value (r_i: tuple from R, s_i: tuple from S)

$$\prod_{i=1}^{n}([r_i] - a) - \prod_{i=1}^{n}([s_i] - a) = 0$$

- If R != S, probability to pass test $p \leq n/2^{128}$

O(n)
Computation and
Communication

Set-based operator

- **Example: Join**
  - Circuit checks confirms that tuples in result T satisfies the join criteria
  - **Set difference** proves that no spurious tuples are added to T
  - **Disjoint** proves that no rows are omitted from T

Set-based operator

- **Set difference**
  - 1) P and V commit $[R - T] = [\Delta_R]$
  - 2) Set difference checks $(Equality, [\Delta_R]||[T], [R])$
  - 3) Compute same proof on $[S], [\Delta_S]$
- **Disjoint**
  - 1) P and V evaluate $[K_R] = (Project, [\Delta_R], R)$ and $[K_S]$
  - 2) Disjoint checks $(Equality, [K_R]||[K_S]||[T], [R]||[S])$

Experimental results

- **<u>Experiment Setting</u>**
  - ZK circuit: EMP-toolkit
  - Backend Database: PostgreSQL
  - Database size: 60k, 120k, 240k rows
  - 16vCPU, 128GiB
- **<u>Testset</u>**
  - Subset of TPC-H benchmark
  - Q1, Q3, Q5, Q8, Q9, Q18

Experimental results

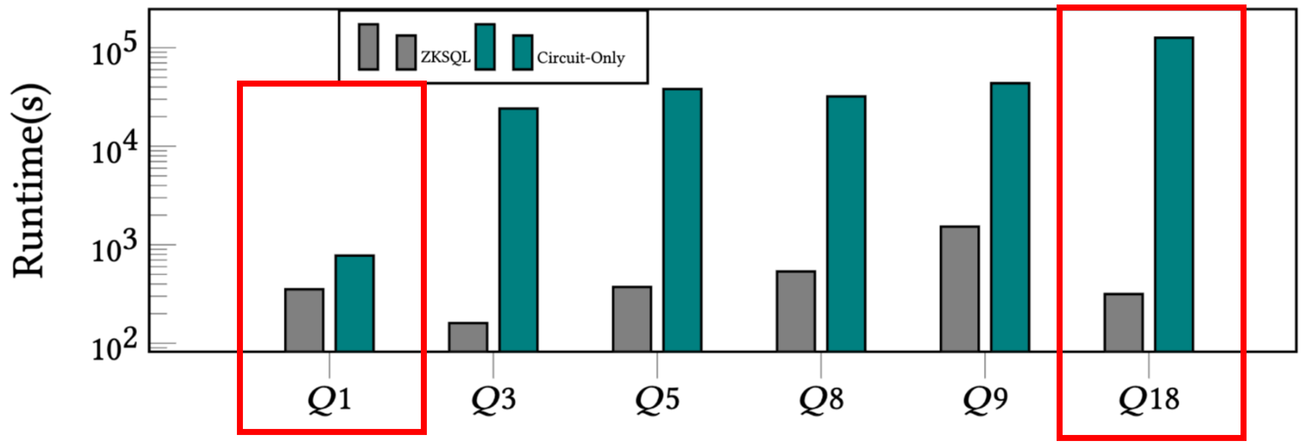


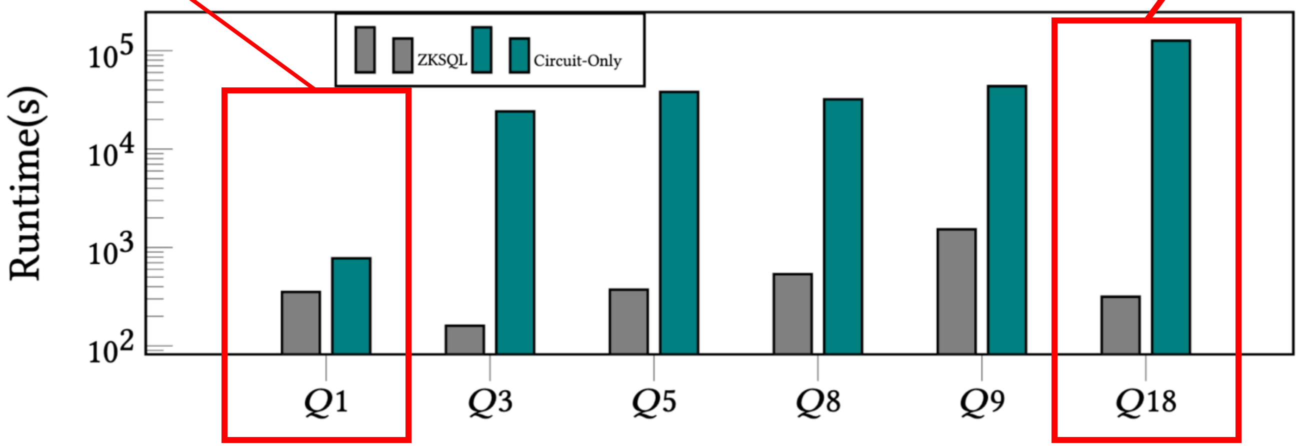
Figure 9: Runtime of ZKSQL vs Circuit-Only baseline.

- Authenticated query answer generation time over 60k rows
- Average **x100 improvement**

Experimental results

Few set-based operations

Many set-based operations
No filter & projection



→ Proving set operation is more efficient than circuit evaluation

Experimental results

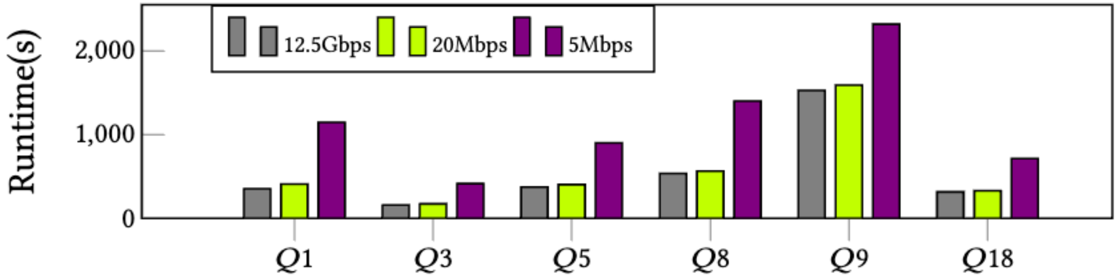


Figure 11: Runtime with decreasing network speeds.

- No bottleneck until 5Mbps
- Reducing the bandwidth to 5Mbps – only X2 slower