SMCQL: Privacy-Preserving Querying for Federated Databases

Johes Bater

With Greg Elliot, Craig Eggen, Satyender Goel, Abel Kho, and Jennie Rogers

Data Solutions… and Issues

- The rise of cheap computing and storage allows people to store and process enormous amounts of data
- This data however, is often fragmented among many different owners
- These users are hesitant to share information with each other, often due to privacy concerns

Motivating Problem

- Data owners want to combine their data with data from untrusted collaborators for analytics, without revealing their raw tuples
- Each party requires a solution that provides:
  - Privacy from each other
  - Efficient execution for queries
  - Usability for their clients

(Some) Existing Solutions

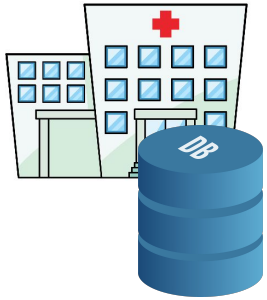
- *Single Database*
  - Store data from all users in one location
  - Problem: Cannot effectively restrict data access
- *Encrypted Query Processing*
  - Execute queries over encrypted data
  - Problem: Vulnerable to side-channel attacks, limited support for complex analytics
- *Differential Privacy*
  - Insert statistical noise into query results
  - Problem: No exact answers, privacy budget

Our Solution: *SMCQL*

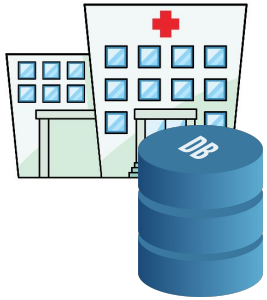
A privacy-preserving, federated database where:

- All computation is carried out *in-situ* using semi-honest secure multiparty computation (SMC)
- A public, unified schema specifies attribute-level privacy guarantees
- Users can submit SQL queries that are run on all participating database parties
- Heuristics-based optimizations automatically generate hybrid secure/plaintext execution plans

Running Example: Medical Data

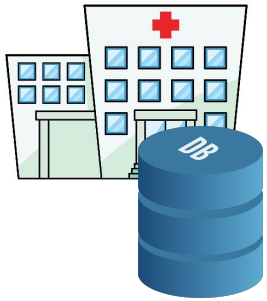


Running Example: Medical Data



patientID	sex	diag	…..
00001	M	blues	…..
00002	F	cdiff	…..
00003	M	X	…..

Running Example: Electronic Health Records



public	private	private	
patientID	**sex**	**diag**	**…..**
00001	M	blues	…..
00002	F	cdiff	…..
00003	M	X	…..

Research Consortium

A Clinical Data Research Network (CDRN) is a consortium of healthcare sites that agree to share their data for research.

For this project, we partnered with HealthLNK, a Chicago-based CDRN

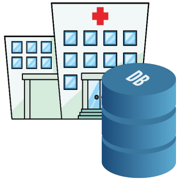
public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

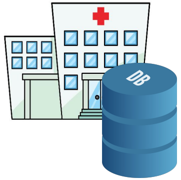
public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

Research Consortium (CDRN)

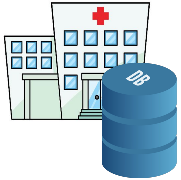
Sharing data among mutually distrustful parties



public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....



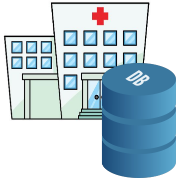
public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....



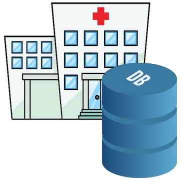
public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

Research Consortium (CDRN)

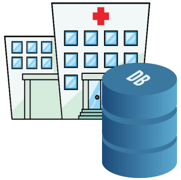
Sharing data among mutually distrustful parties



public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....



public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....



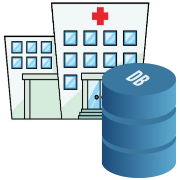
public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

Client

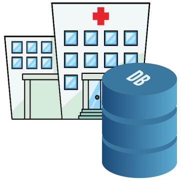
Research Consortium (CDRN)

"How many patients are there?"

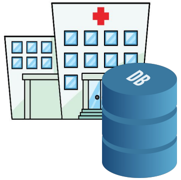
Client



public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....



public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....



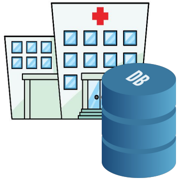
public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

Research Consortium (CDRN)

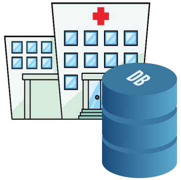
"How many patients are there?"

```
SELECT
COUNT(DISTINCT
PATIENT ID)
FROM diagnosis;
```

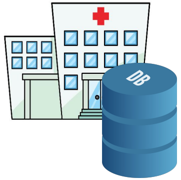
Client



public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....



public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....



public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

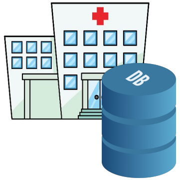
Research Consortium (CDRN)

“How many patients are there?”

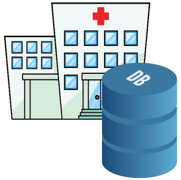
```
SELECT
COUNT(DISTINCT
PATIENT ID)
FROM diagnosis;
```

Client

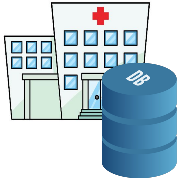
Honest Broker



public	private	private	
patientID	**sex**	**diag**	**.....**
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....



public	private	private	
patientID	**sex**	**diag**	**.....**
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....



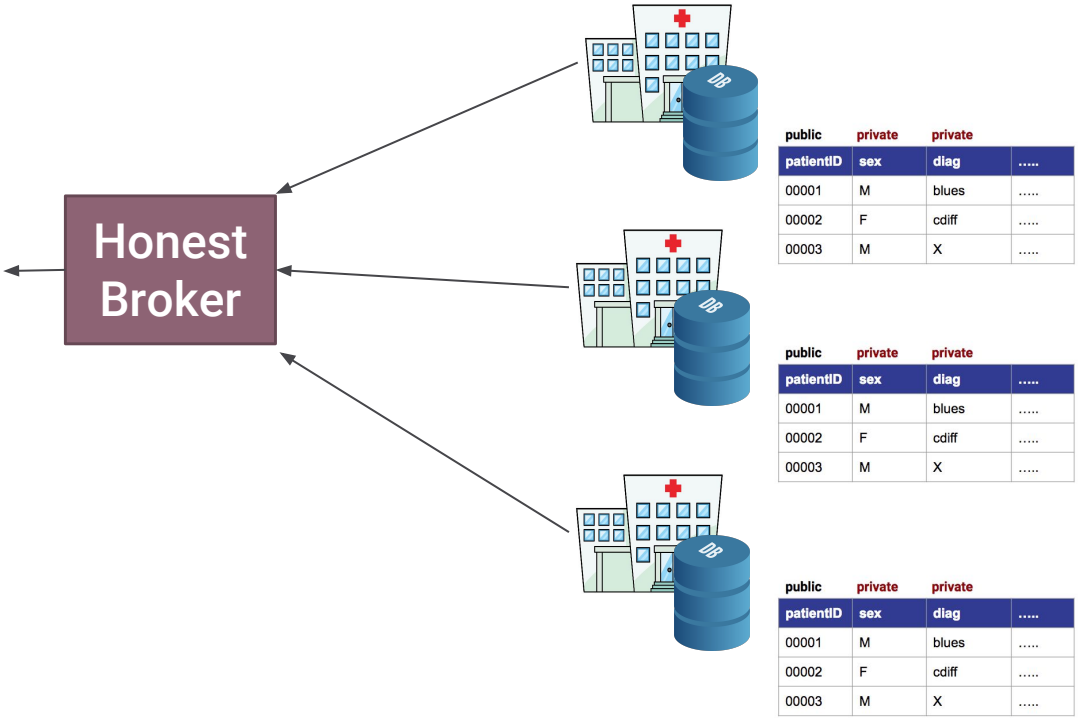
public	private	private	
patientID	**sex**	**diag**	**.....**
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

Research Consortium (CDRN)

"How many patients are there?"

```
SELECT
COUNT(DISTINCT
PATIENT ID)
FROM diagnosis;
```

Client



public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

Research Consortium (CDRN)

"How many patients have rare disease X?"

Client

Honest Broker

public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

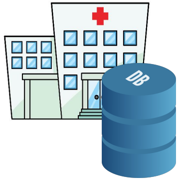
Research Consortium (CDRN)

"How many patients have rare disease X?"

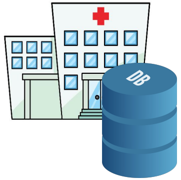
```
SELECT
COUNT(DISTINCT
patient id)
FROM diagnosis
WHERE diag=X;
```

Client

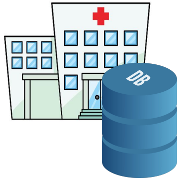
Honest Broker



public	private	private	
patientID	**sex**	**diag**	**.....**
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....



public	private	private	
patientID	**sex**	**diag**	**.....**
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....



public	private	private	
patientID	**sex**	**diag**	**.....**
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

Research Consortium (CDRN)

"How many patients have rare disease X?"

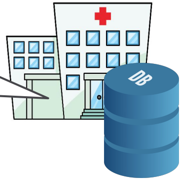
```
SELECT
COUNT(DISTINCT
patient id)
FROM diagnosis
WHERE diag=X;
```

Client

I'm not telling anyone private data!

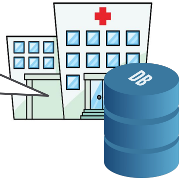
public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

I'm not telling anyone private data!



public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

I'm not telling anyone private data!



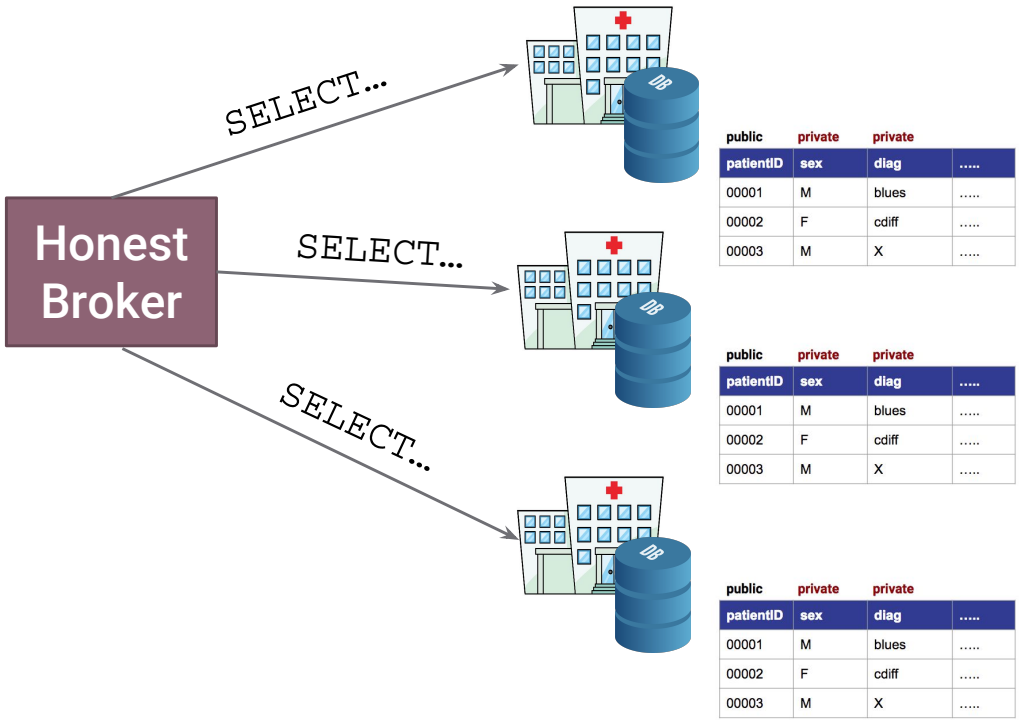
public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

Research Consortium (CDRN)

"How many patients have rare disease X?"

```
SELECT
COUNT(DISTINCT
patient id)
FROM diagnosis
WHERE diag=X;
```

Client



public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

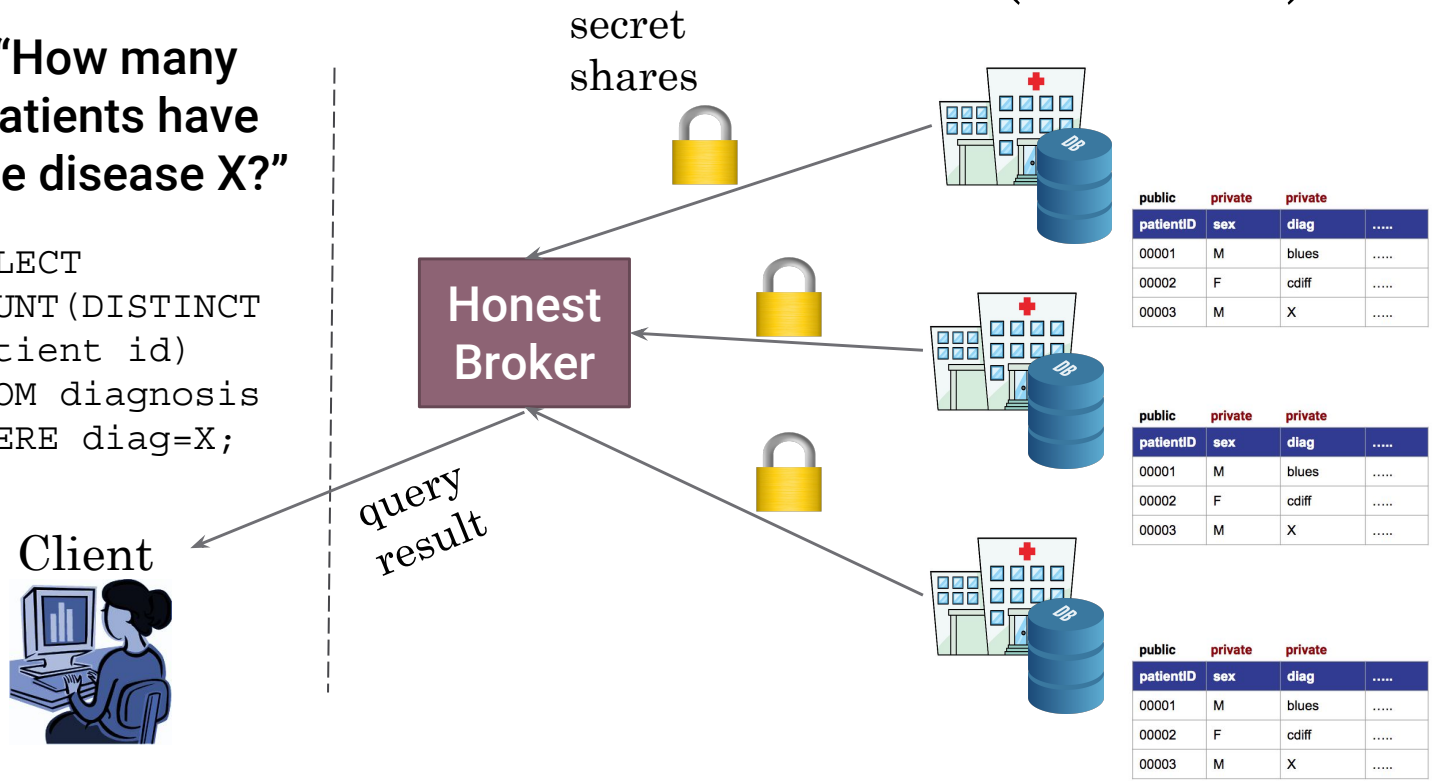
public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

Securely compute query

Research Consortium (CDRN)

“How many patients have rare disease X?”

```
SELECT
COUNT(DISTINCT
patient id)
FROM diagnosis
WHERE diag=X;
```



public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

public	private	private	
patientID	sex	diag	.....
00001	M	blues	.....
00002	F	cdiff	.....
00003	M	X	.....

Distrustful Data Federation (DDF)

- *Privacy-preserving*
  - Must not leak data to other participating nodes
  - Protects from outside attackers
- *Usable*
  - Accepts SQL queries
  - Automatic translation into SMC
- *Efficient*
  - Runs in a reasonable amount of time
  - Highly optimized to minimize SMC

SMC Building Blocks

- *Garbled circuits*
  - Cryptographic protocol used to securely compute a function across two parties
  - Protects a query's program traces from snooping
- *Oblivious RAM* (ORAM)
  - Data structure that shuffles all data on any read/write
  - Protects memory traces from leaking information
  - Up to $O(\log^2 n)$ cost per I/O
- *ObliVM*
  - Code generator that translates C-style code into garbled circuits and ORAM
  - Translates DB operators into garbled circuits

ObliVM Code Generator

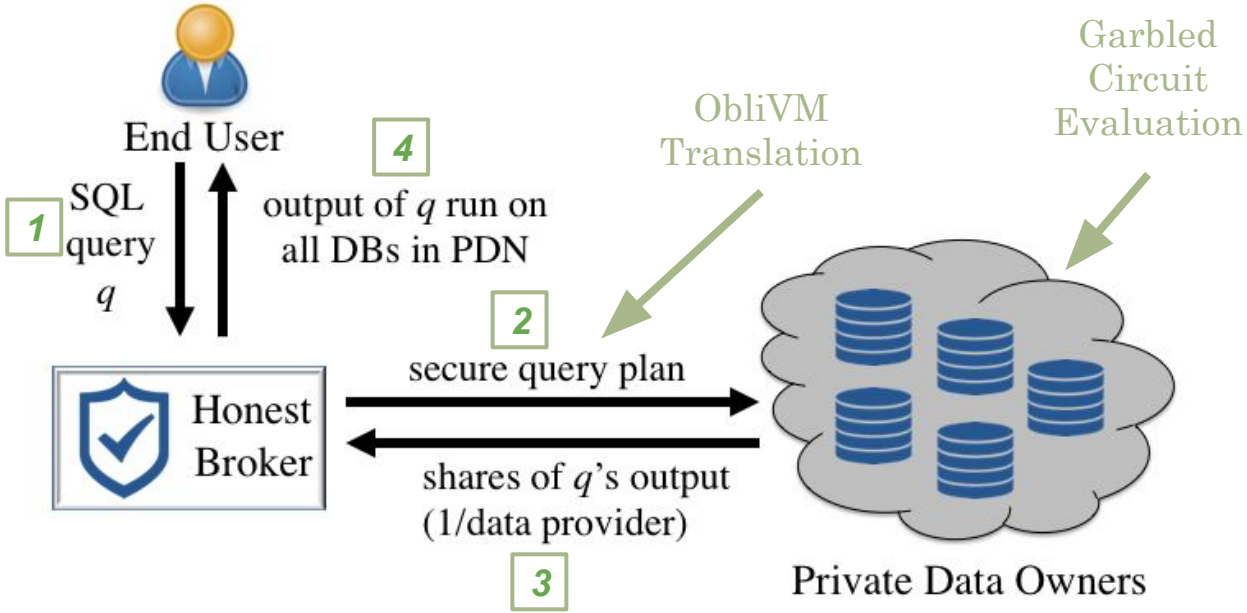
- Convert a plaintext function into an oblivious version
- ObliVM library
- C-style syntax
- We use templates for generating secure operators

```
int$dSize[m*n] join(int$lSize[m] lhs, int$rSize[n] rhs) {
      int$dSize[m*n] dst;
      int dstIdx = 0;

      for(int i = 0; i < m; i=i+1) {
          int$lSize l = lhs[i];
          for(int j = 0; j < n; j=j+1) {
              int$rSize r = rhs[j];
              if($filter(l, r) == 1) {
                dst[dstIdx] = $project;
                dstIdx = dstIdx + 1;
              }
          }
       }
    return dst;
}
```

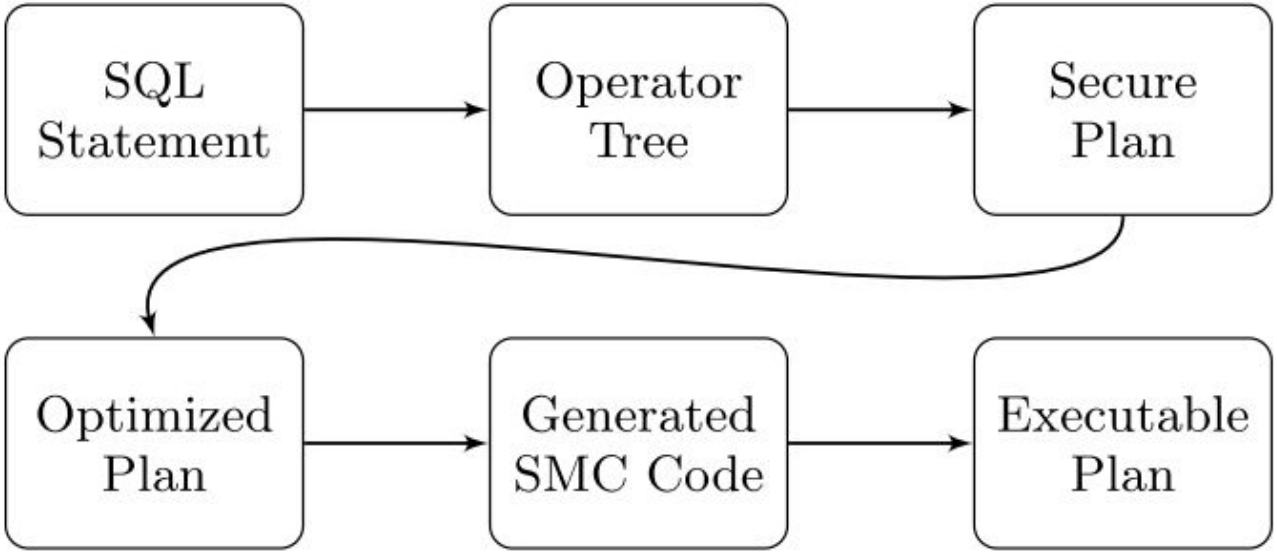
`blue` = constant inserted at compile time `red` = function generated at compile time `green` = constant populated at runtime

DDF Architecture



This work is a 2-party prototype

Honest Broker: Query Planning



HealthLNK Reference Queries

COMORBIDITY

```
SELECT diag, COUNT(*) cnt
FROM diagnoses
WHERE patient_id IN cdiff_cohort
GROUP BY diag
ORDER BY cnt
LIMIT 10;
```

ASPIRIN COUNT

```
SELECT COUNT(DISTINCT pid)
FROM diagnosis d
  JOIN medication m ON d.pid = m.pid
WHERE d.diag = hd AND m.med = aspirin
  AND d.time <= m.time;
```

RECURRENT C. DIFF

```
WITH rcd AS (
  SELECT pid, time, row_no() OVER
    (PARTITION BY pid ORDER BY time)
  FROM diagnosis
  WHERE diag=cdiff)

SELECT DISTINCT pid
FROM rcd r1 JOIN rcd r2 ON r1.pid =
r2.pid
WHERE r2.time - r1.time >= 15 DAYS
  AND r2.time - r1.time <= 56 DAYS
  AND r2.row_no = r1.row_no + 1;
```

Hand-Coded SMC Performance

Test (50 tuples)	Plaintext (ms)	Secure (ms)	Slowdown
Comorbidity	148	253,894	1,609X
Recurrent C. Diff	165	159,145	967X
Aspirin Count	193	8,195,317	43,337X

Pure SMC: TOO SLOW!

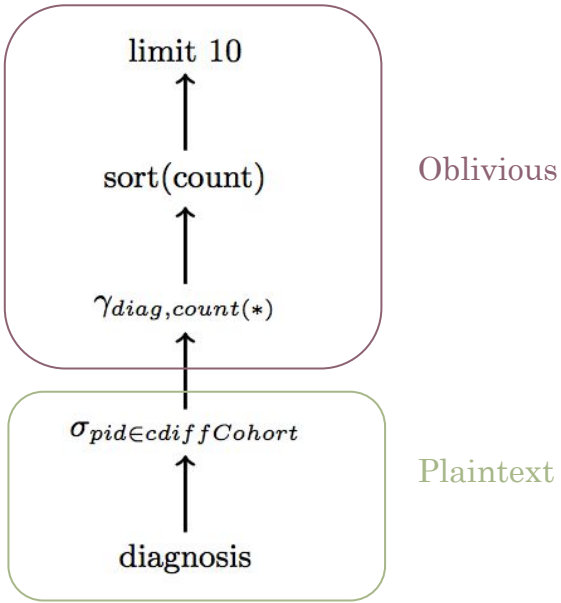
Attribute-Level Security Model

Unified schema with security annotations:

- *Public attributes*
  - Readable by all parties
  - E.g., Lab results, anonymized IDs
- *Protected attributes*
  - Conditionally available to other parties (k-anonymous)
  - E.g., Age, gender, diagnosis codes
- *Private attributes*
  - Only available to originating party
  - E.g., Timestamps, zip codes

Reducing Use of Secure Computation

- Trace the flow of sensitive attributes through the operator tree
- Identify minimal subtree that must be computed securely to uphold security policy

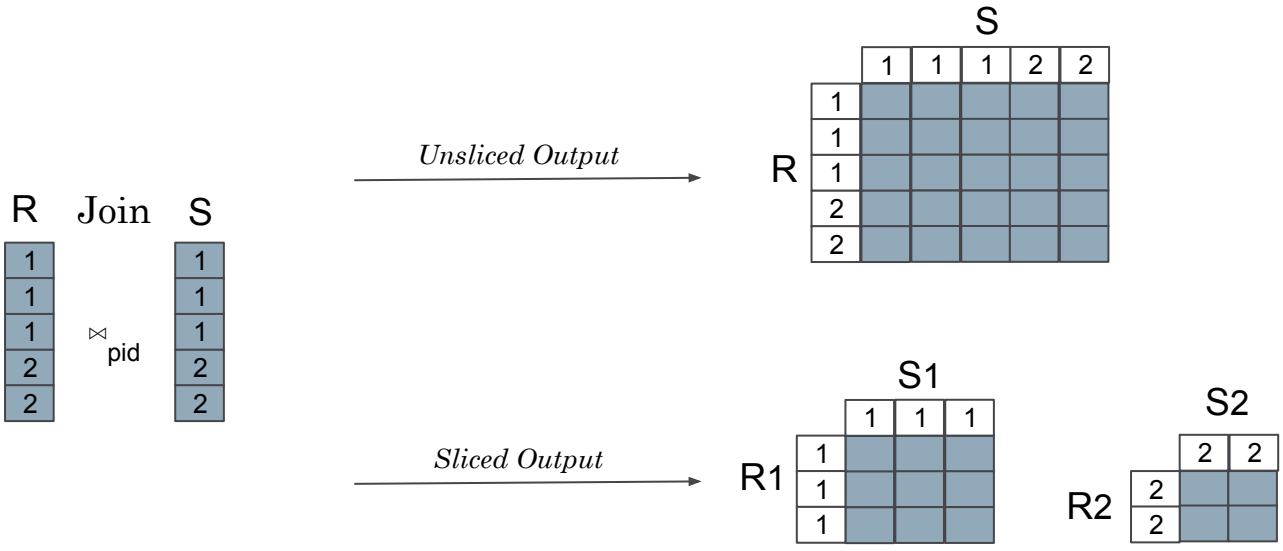


COMORBIDITY

```
SELECT diag, COUNT(*) cnt
FROM diagnoses
WHERE patient_id IN cdiff_cohort
GROUP BY diag
ORDER BY cnt
LIMIT 10;
```

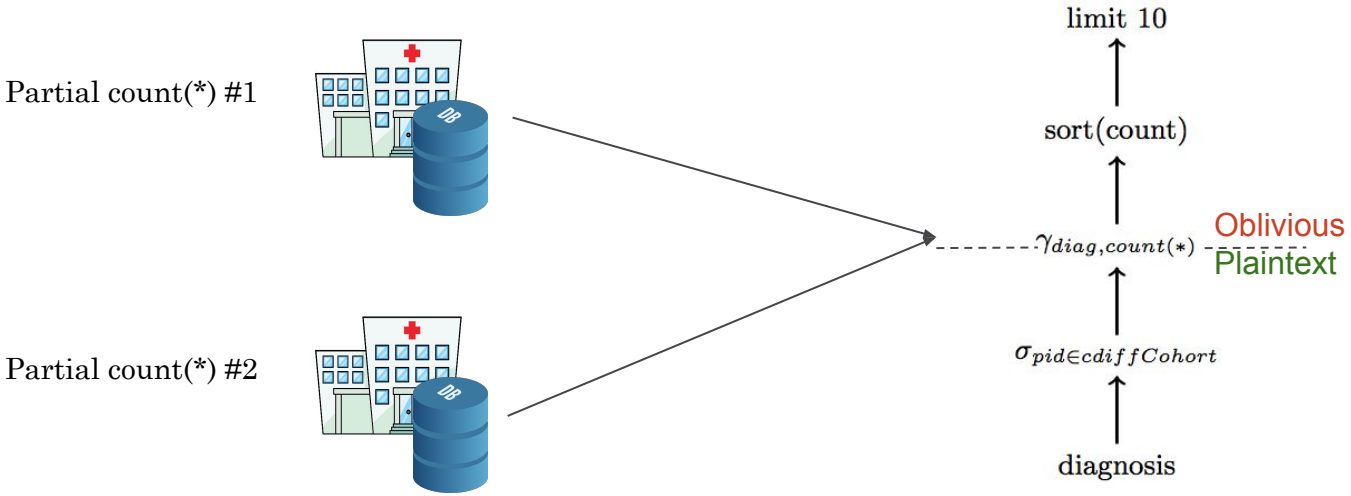
Optimizing Queries: Sliced Evaluation

Horizontally partition data on public attributes for oblivious evaluation



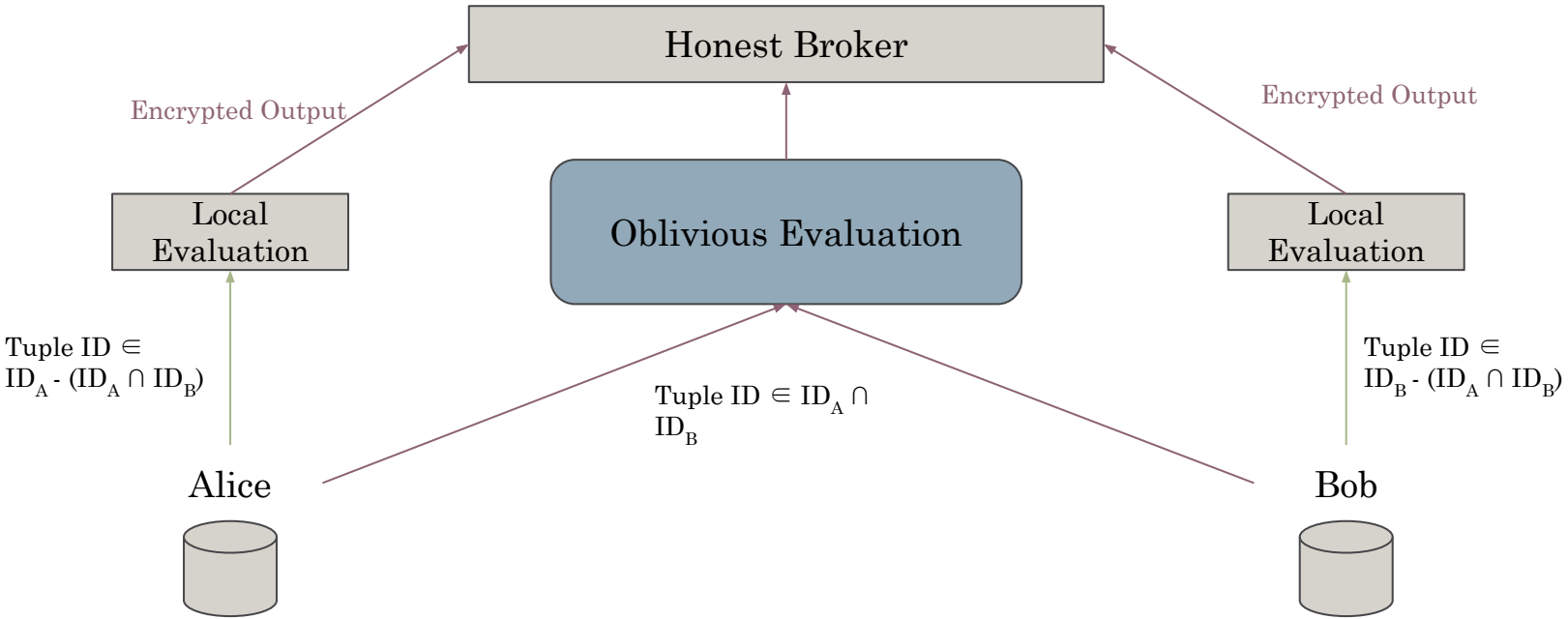
Optimizing Queries: Split Operators

Precompute part of the operator locally

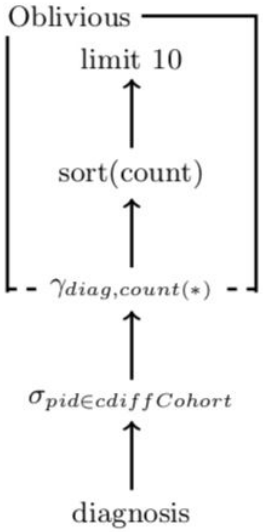


Optimizing Queries: Semi-Join

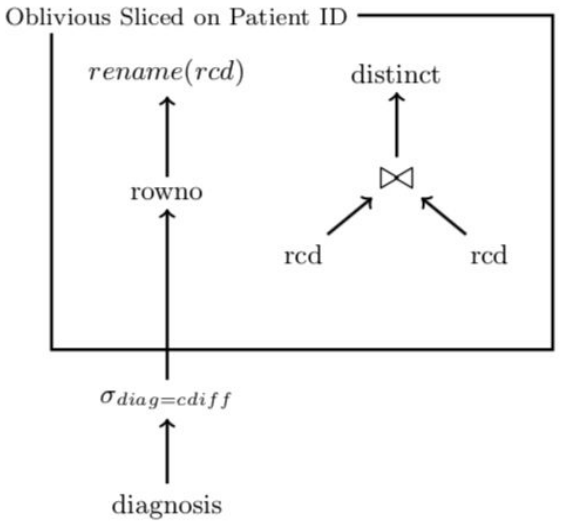
Find single-party slices to eliminate unnecessary SMC



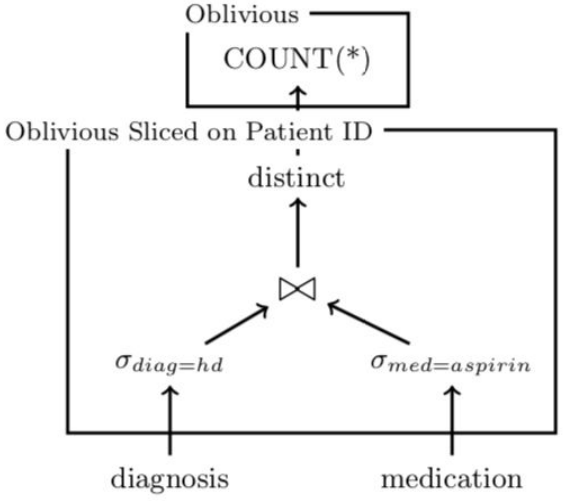
Optimized Plans



(a) Comorbidity



(b) Recurrent *C.Diff*

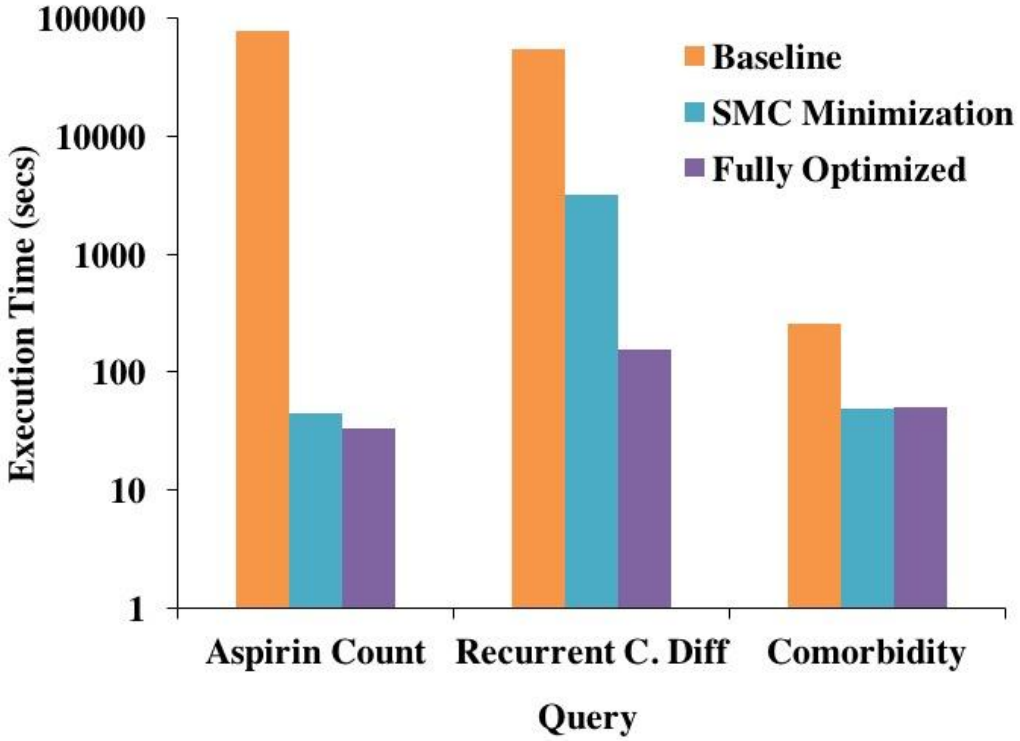


(c) Aspirin Count

Experimental Setup

- HealthLNK data repository
  - One year of data
  - 500,000 patients
  - 42 million diagnoses
  - 23 million medication records
  - 15 GB
- Experimental Setup
  - 2-party prototype
  - 3 pairs of servers

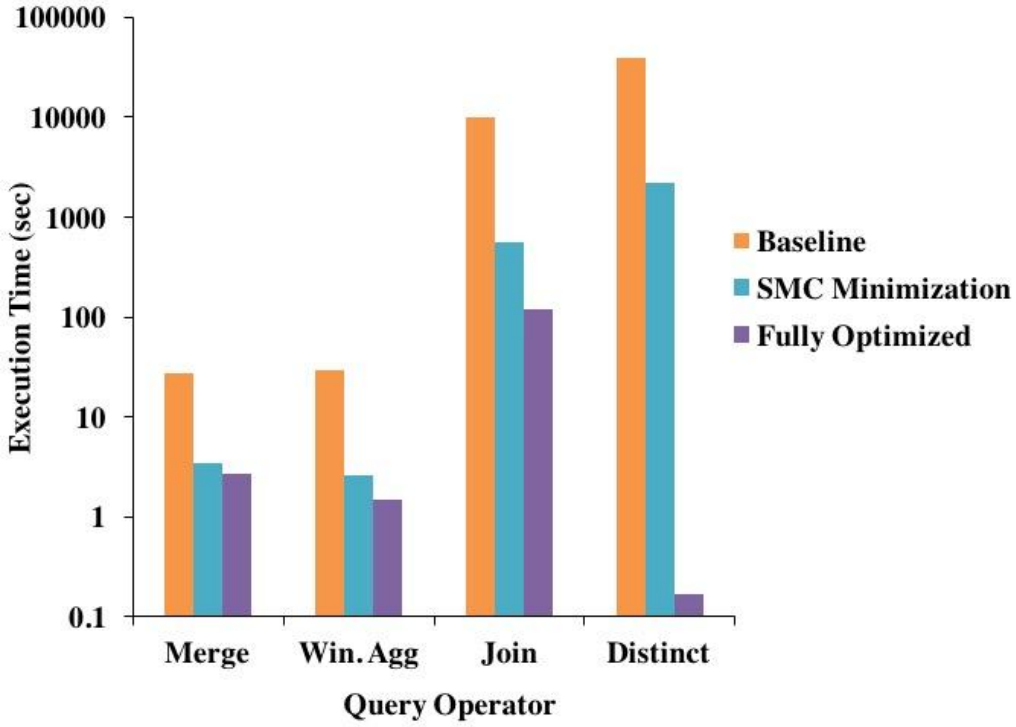
Performance on Reference Queries



Minimizing SMC use by reducing secure subtrees greatly improves performance

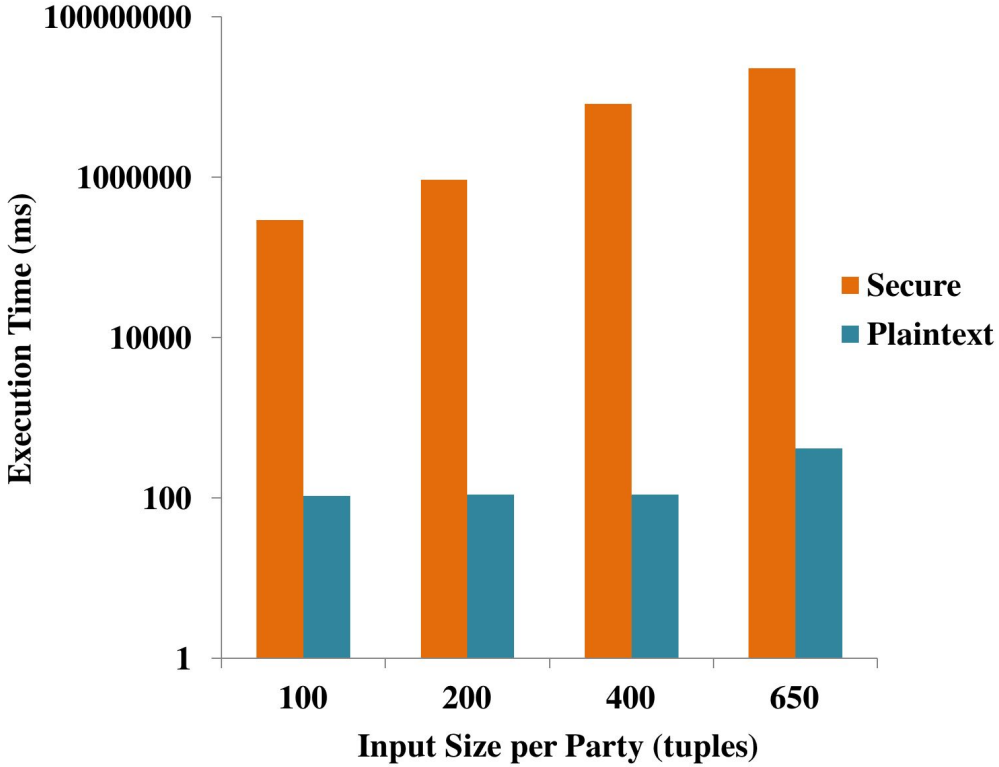
Full Optimization using slicing often provides further benefits

Operator Level Performance for Recurrent C. Diff



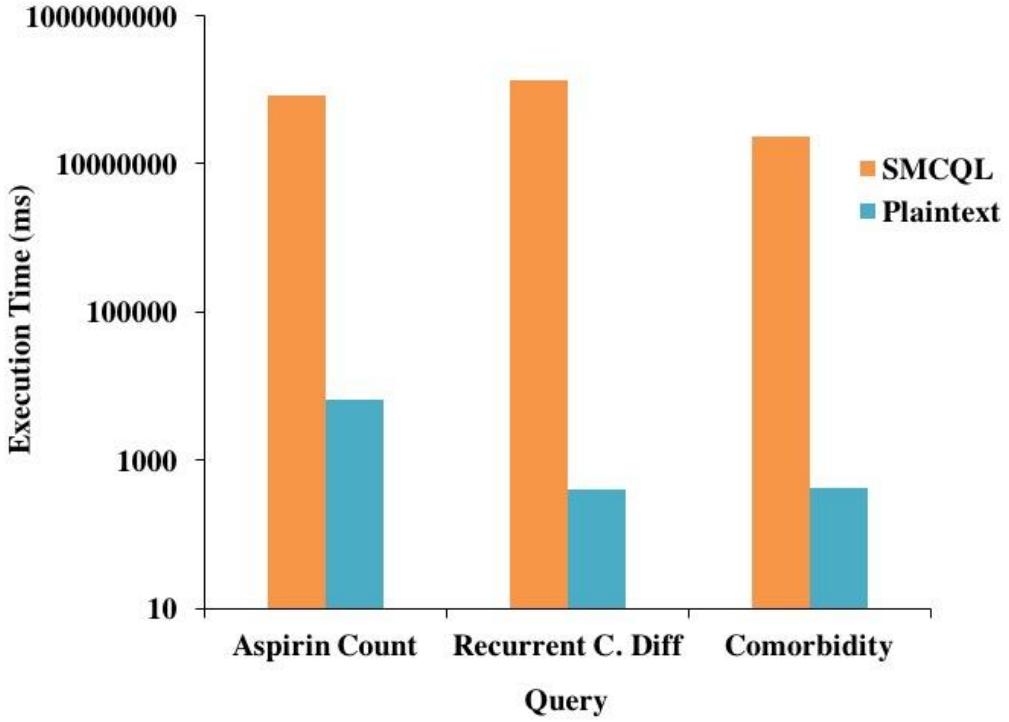
Optimization is dependent on both the query and the input data

System Scale Up



Minimizing the secure subtree enables us to scale to larger inputs.

SMCQL vs Plaintext



Secure computation has substantial overhead, but there is fertile ground for optimization in future work.

Contributions

- Generalization of federated DBMSs for mutually distrustful parties in a semi-honest setting
- SQL-to-SMC translator
- Heuristics-based optimizer for managing use of SMC that leverages fine-grained privacy annotations for schemas