SecureDB

A Secure Query Processing System in the Cloud

Group Member: Haibin LIN, Eric
Supervisor: Prof Benjamin Kao
Department of Computer Science, University of Hong Kong

Overview

1. **The Problem**
2. Related Work
3. Theoretical Background
4. System Architecture
5. Component Implementation
6. Experiment Result

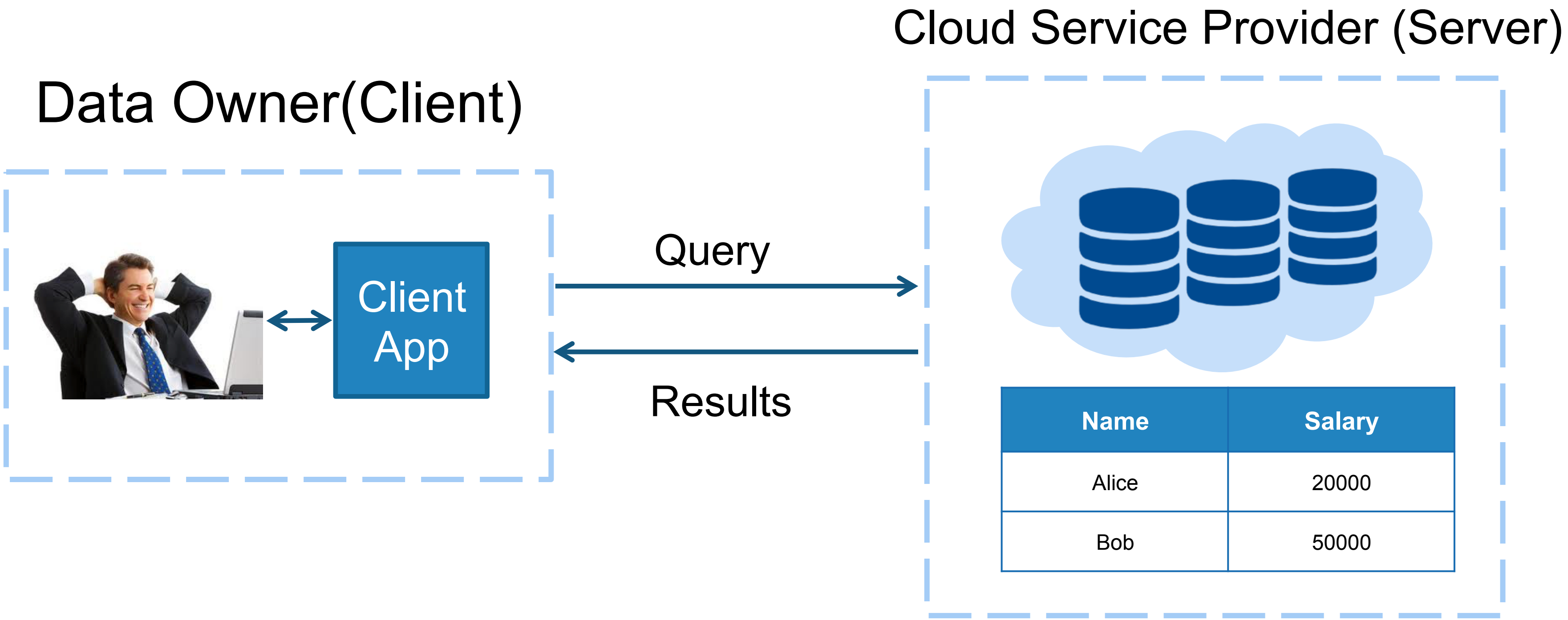
Background

Cloud Service Provider (Server)

RDS amazon web services

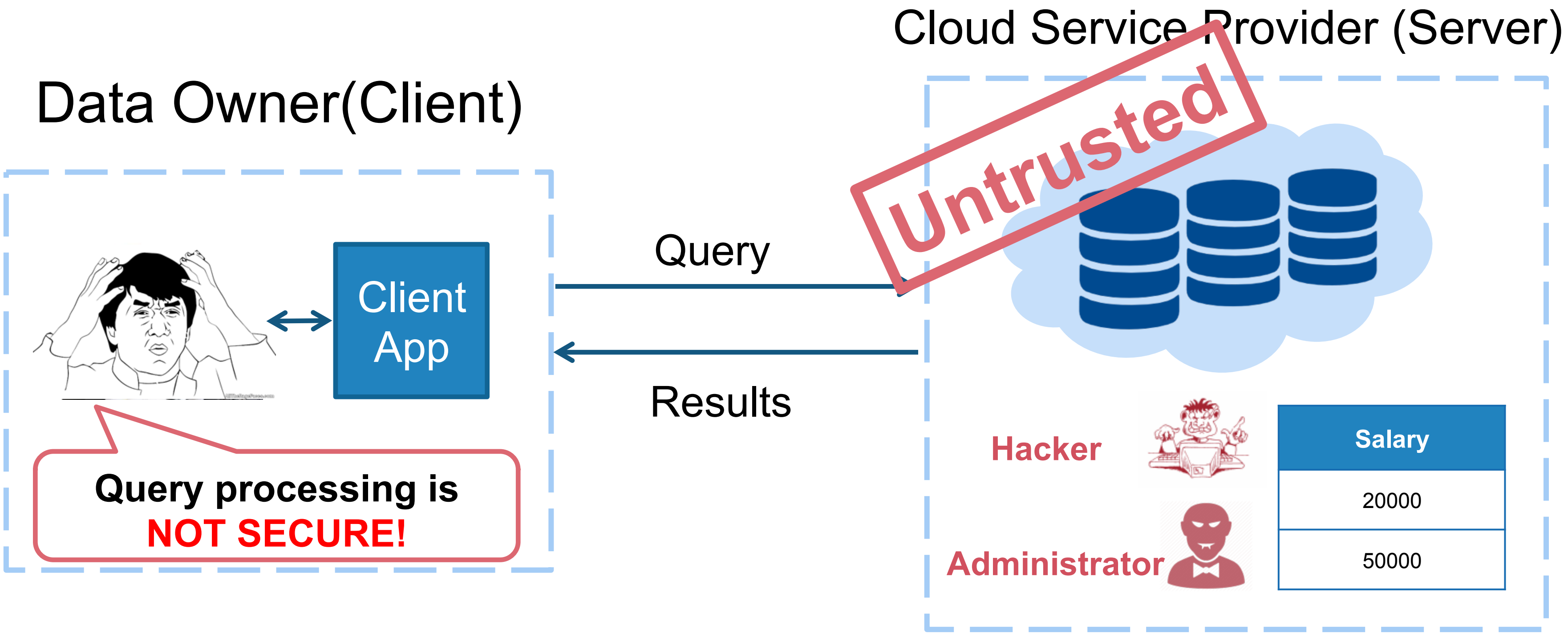
Microsoft SQL Azure

Background



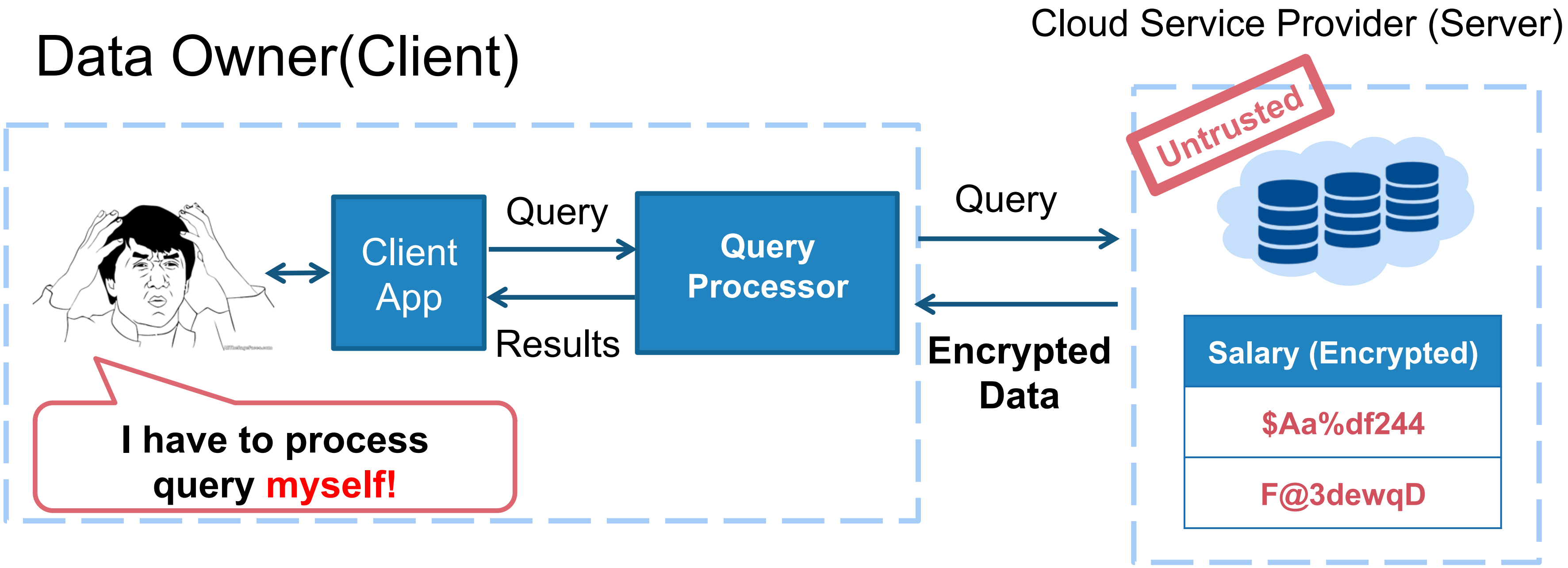
Name	Salary
Alice	20000
Bob	50000

The Problem



Cloud Service Provider (Server)
Data Owner(Client)
Untrusted
Client App
Query
Results
Hacker
Administrator
Query processing is
NOT SECURE!
Salary
20000
50000

Decrypt-Before-Query Approach



Overview

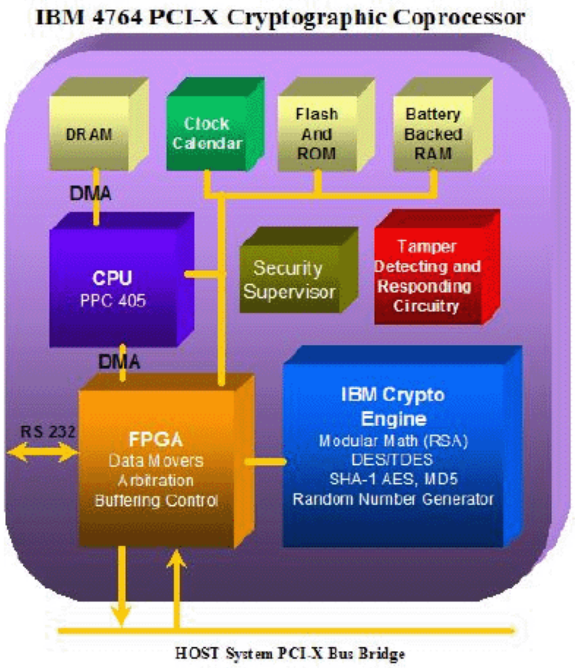
1. The Problem
2. **Related Work**
3. Theoretical Background
4. System Architecture
5. Component Implementation
6. Experiment Result

Related Work

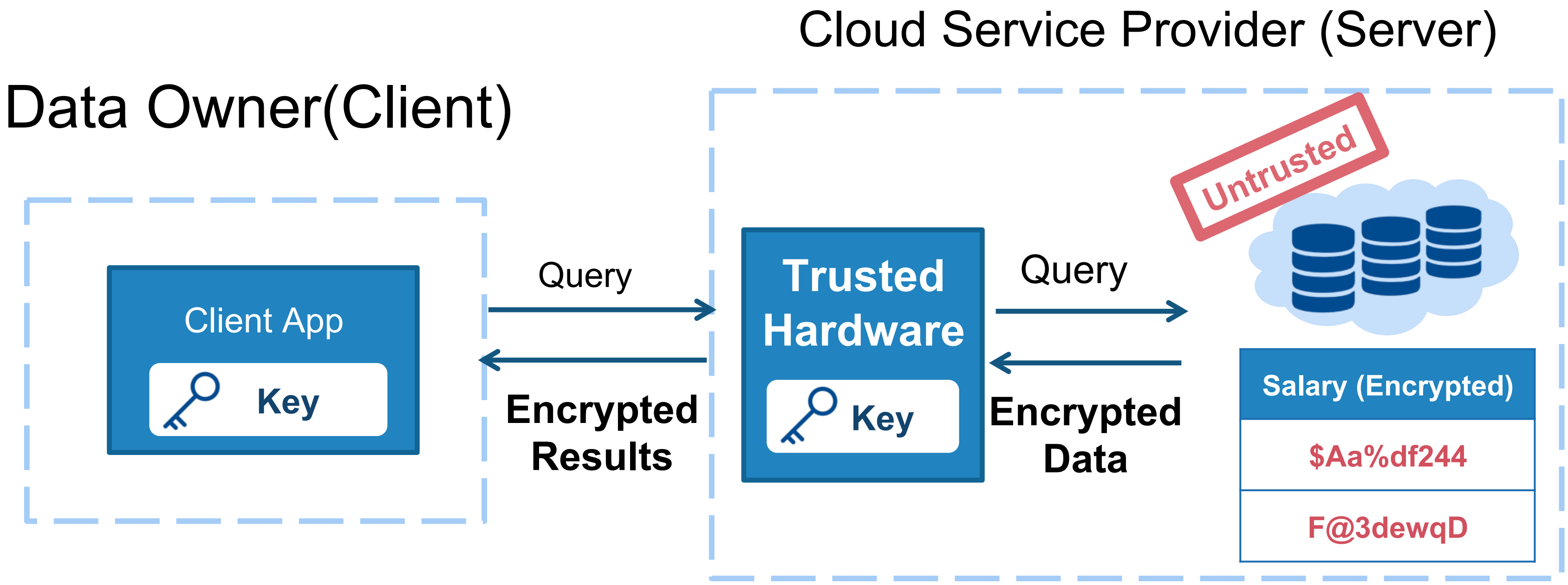
1. Hardware Approach

TrustedDB(2011)[1]

- Based on trusted **secure co-processor**
- Dedicated **hardware** for cryptographic operation



Related Work



Related Work

1. Hardware Approach

TrustedDB(2011)

Advantage	Disadvantage
Strong Security	Expensive Hardware $$$$$$$$
Accepts any kind of query	

Related Work

2. Software Approach

a. Fully Homomorphic Encryption

- Allows **arbitrary computation** on ciphertext without knowing the key, including +, -, *, /, >, =, √ …

- Limitation: **Computationally Expensive**
  e.g. 30 minutes per bit operation(2011)[2]

Related Work

2. Software Approach

b. CryptDB(2012)[3]

- Multiple layers of partially homomorphic encryptions

Encryption Layer	E1	E2	E3
Operations Supported	None	Equality check	Equality check Ordering comparison
Security Level	Strongest	Strong	Not secure against CPA

Related Work

2. Software Approach

b. CryptDB(2012)

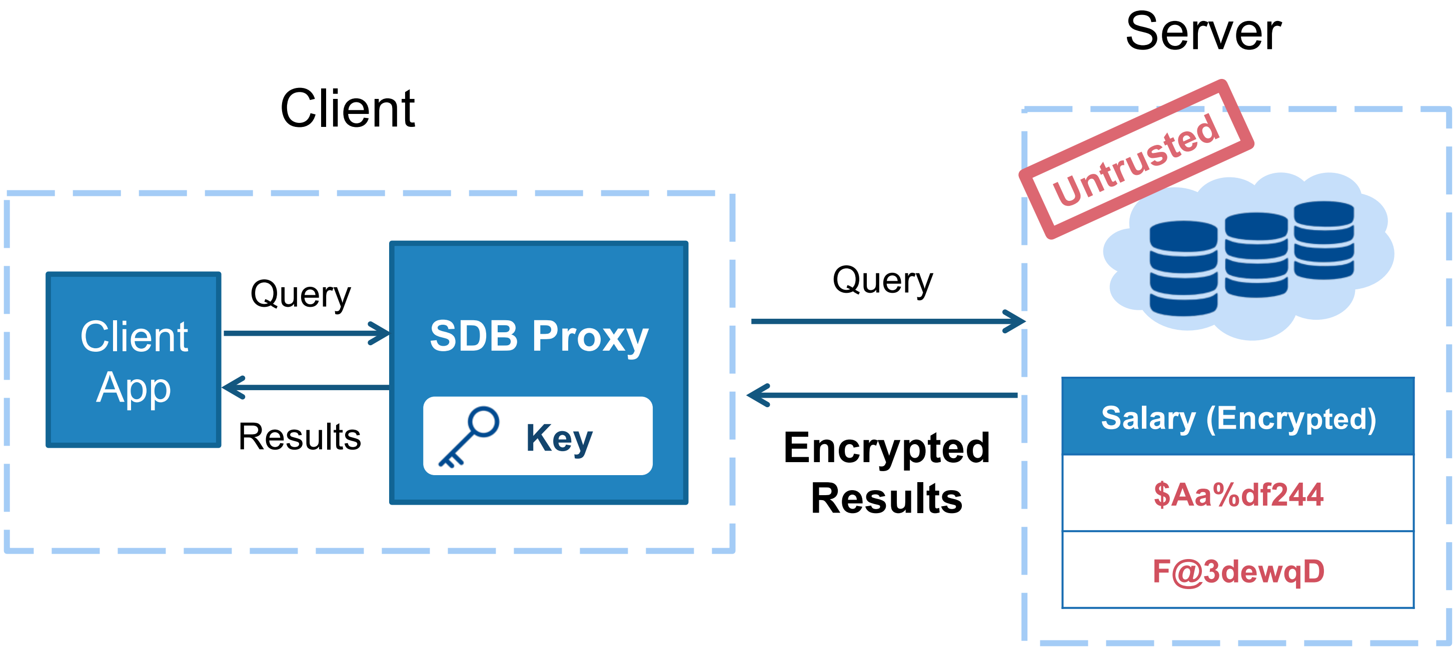
- Limitation: supports **limited types of queries**

Query Type	Example	Supported?
Computation	SELECT **a * b** FROM T	
Comparison	SELECT a, b FROM T WHERE **a > b**	
Computation & Comparison	SELECT a, b FROM T WHERE **a * b > c**	

What is SecureDB?

- SDB is a secure query processing system based on **secret sharing**
- Motivation
  1. Runs on commodity hardware
  2. Accepts a wide range of queries
  3. Both efficient and secure!
  4. Less effort for the client

What is SecureDB?



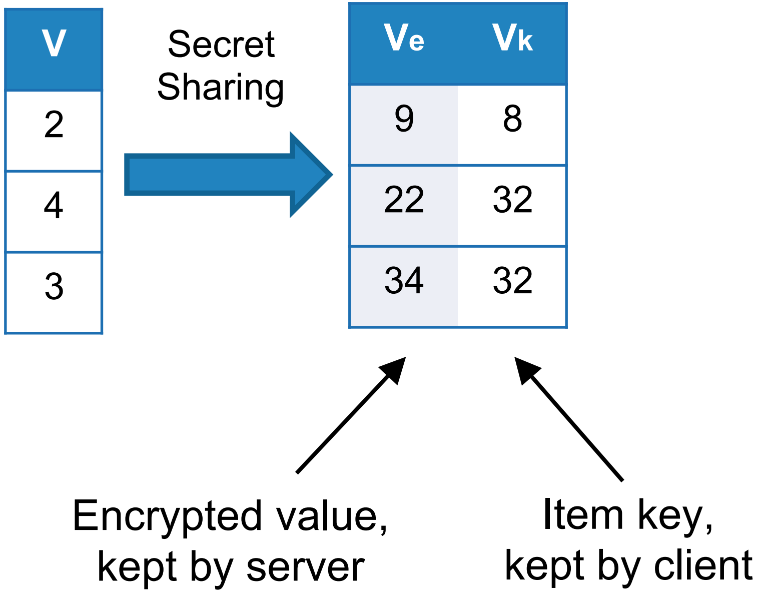
Overview

1. The Problem
2. Related Work
3. **Theoretical Background**
4. System Architecture
5. Component Implementation
6. Experiment Result

Secret Sharing

- Secret Sharing Scheme
  - For a sensitive value V, we split it into two shares: the **encrypted value** V_e and the **item key** V_k
  - One needs **both** V_e and V_k to recover the value of V

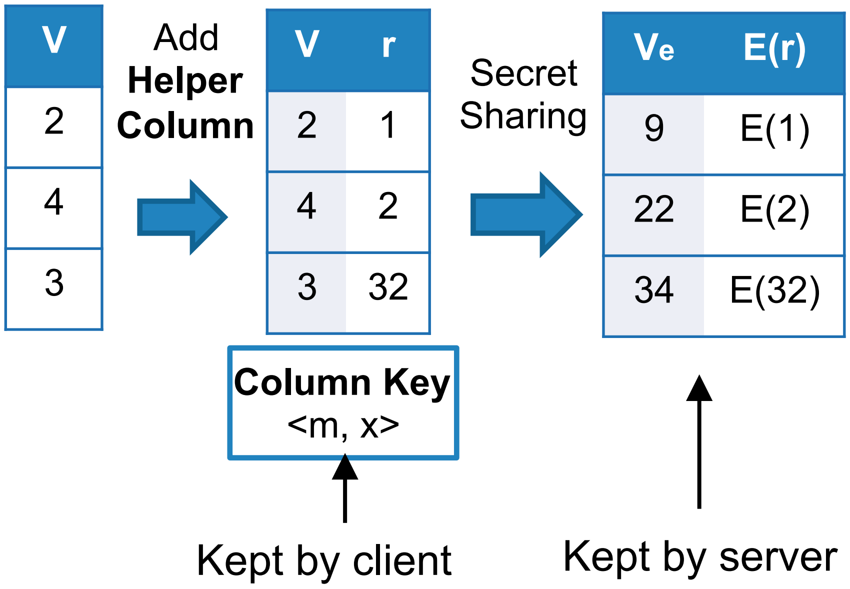
    $V = \mathrm{Decrypt}(V_e, V_k)$



Secret Sharing

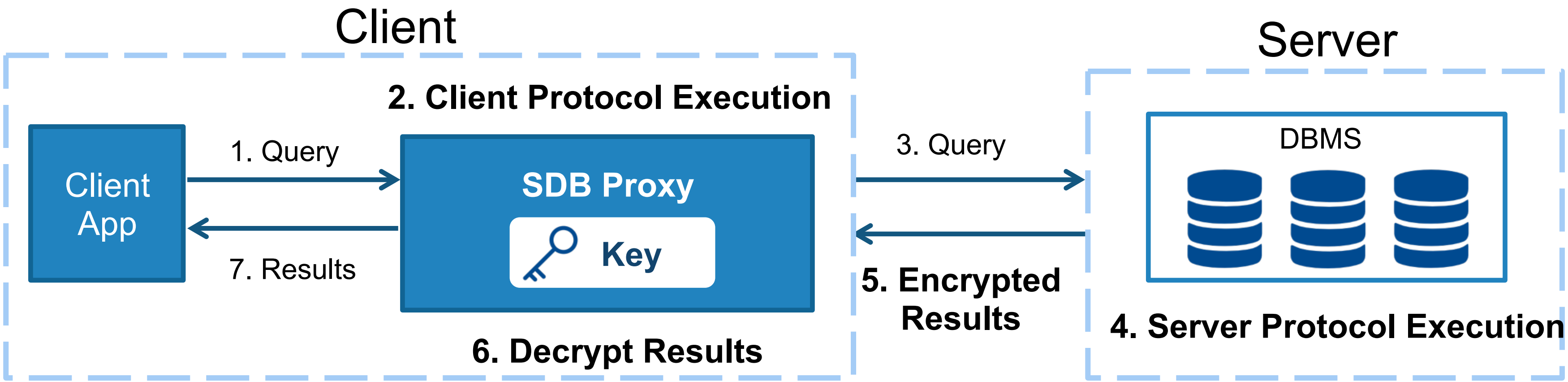
- Secret Sharing in SDB
  - Encrypt sensitive values on a column basis
  - Add helper column r so that client can compute item keys **on the fly**

$V_k = genItemKey(r, <m,x>)$



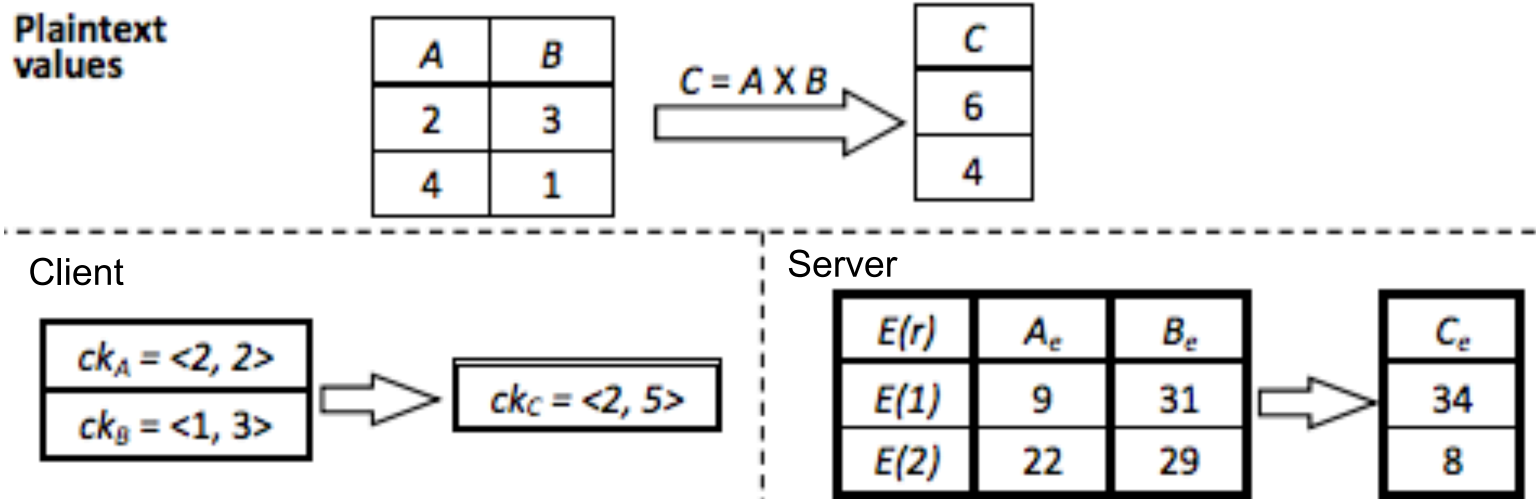
Computation Protocol

- Secure Computation Protocol
  - For any operation on V (+, -, *, <, >, =), the server can complete the operation **without knowing column keys**
  - Includes **client protocol** and **server protocol**



Computation Protocol

- Example: Secure protocol for multiplication
  1. Client computes a new column key. $Ckc = \langle m_A * m_B,\ x_A + x_B \rangle$
  2. Server computes on the bulk encrypted data. $C_e = A_e * B_e \mod n$
  3. Finally, client decrypts the encrypted result with Ckc



Challenge

- Every basic operator(e.g. *, +, >) has a unique protocol
- How to automate the execution process?
    1. Build a new DBMS from scratch? Or
    2. Incorporate these protocols with a existing database system?

Overview

1. The Problem
2. Related Work
3. Theoretical Background
4. **System Architecture**
5. Component Implementation
6. Experiment Result

System Architecture

- SparkSQL: a cluster computing engine that supports SQL
- User Defined Function(UDF) & Query Rewrite

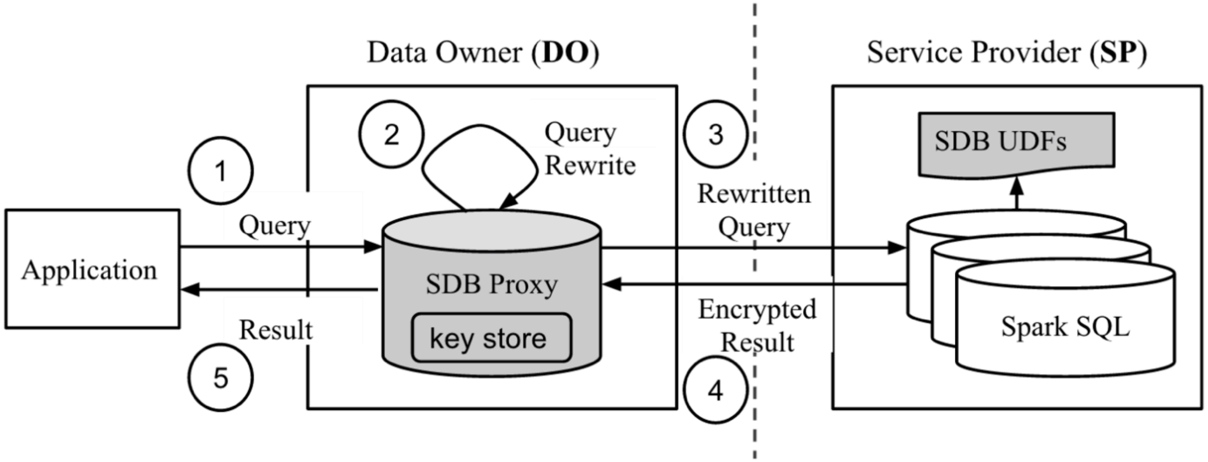


Figure 5: Architecture of SDB

1 select A * B from T

3 select **sdb_mul(A,B, …)**, **row_id** from T

Why Query Rewrite & UDF?

1. Performance wise

- User Defined Function executed in the same address space of SparkSQL

  => Little memory copy, little network transfer and no IPC

2. Engineering wise

   - Normal operators provided by SparkSQL
   - Server side queries optimized by SparkSQL
   - Machine failures, disk-based processing and parallelism handled by SparkSQL

Overview

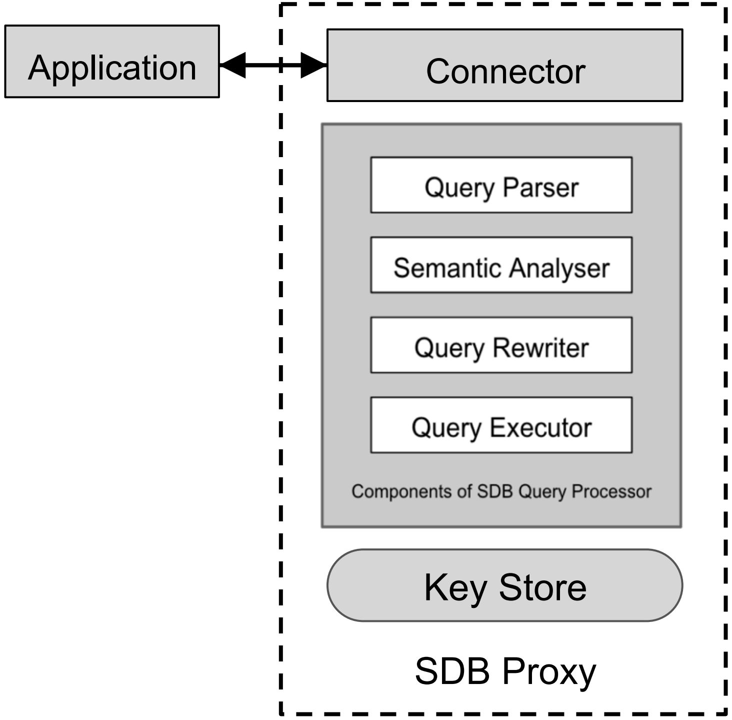
1. The Problem
2. Related Work
3. Theoretical Background
4. System Architecture
5. **Component Implementation**
6. Experiment Result

SDB Proxy Components

Components of SDB Proxy

- Connector
- Key Store
- Query Processor

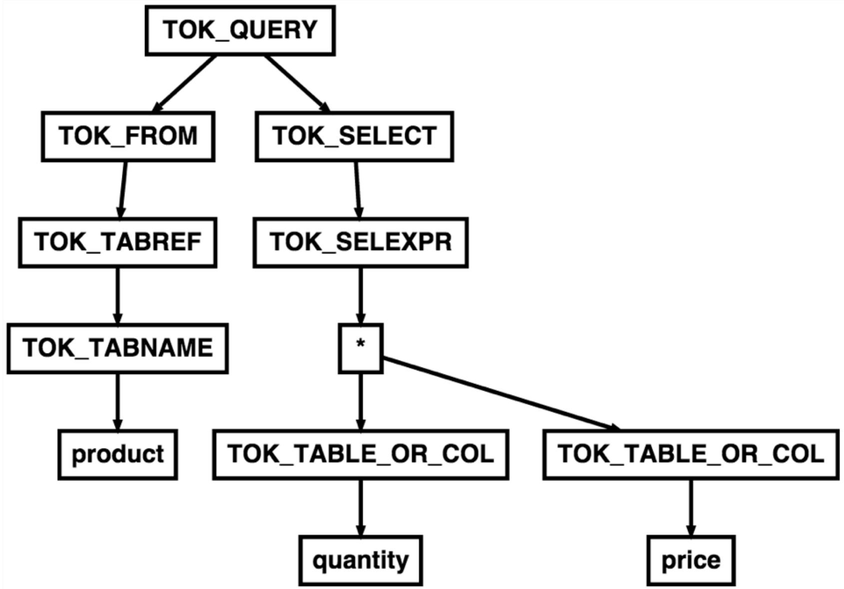
Currently supports +, -, *, >, =, <, count().
~18000 lines of Java code



Query Parser

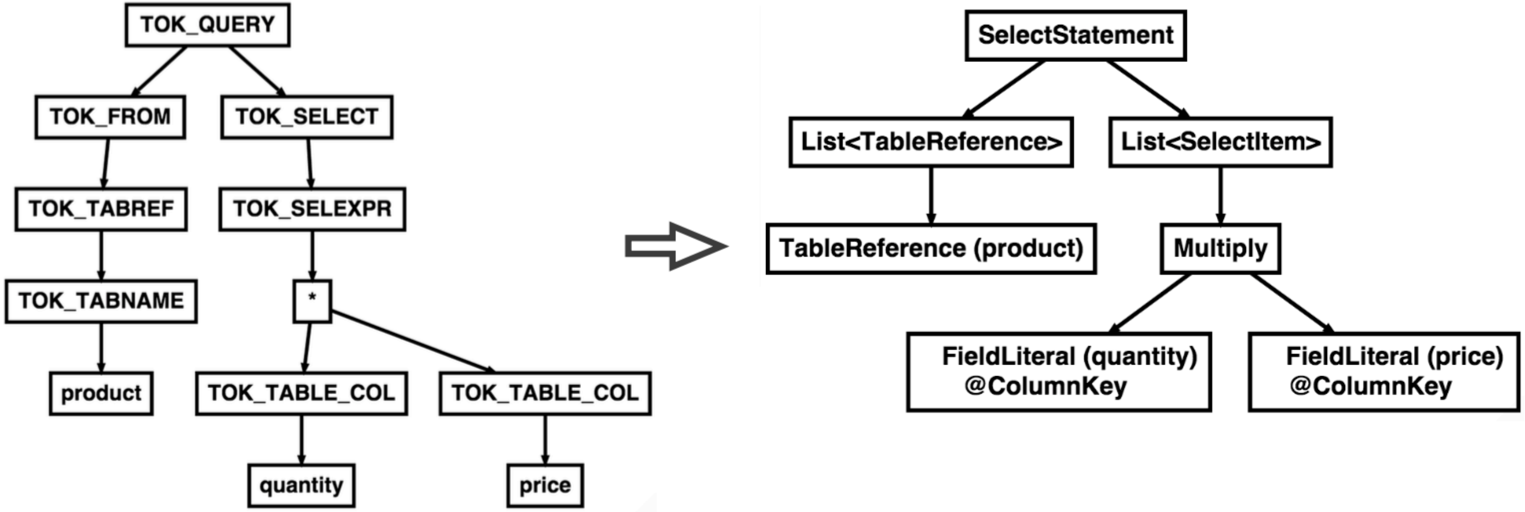
- Parse query **strings** into **abstract syntax trees**

SELECT quantity * price
FROM product



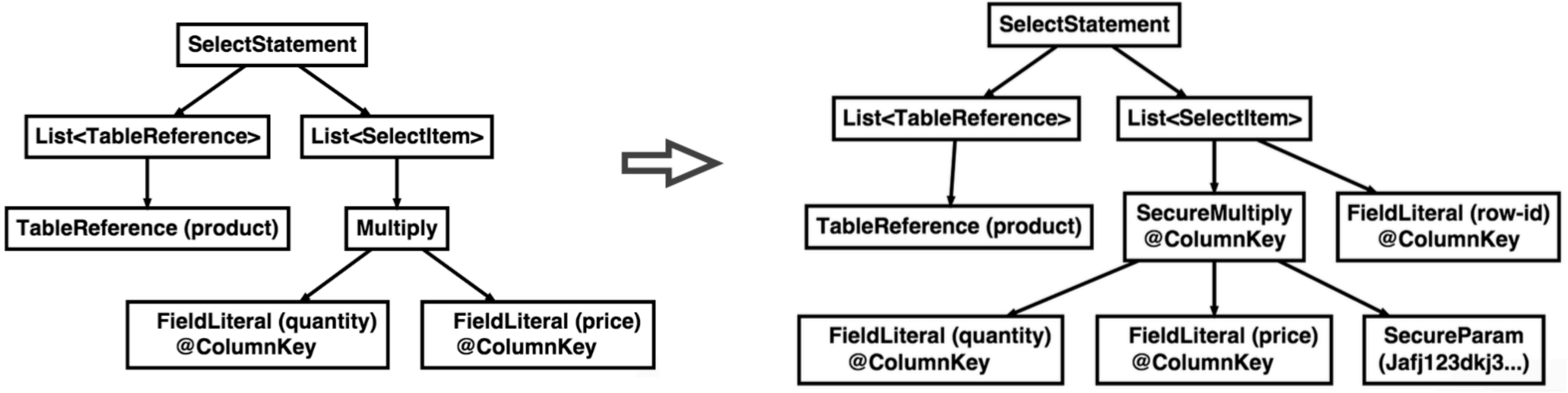
Semantic Analyser

- Transform abstract syntax trees into logical plan trees, access key store to
  1. **Verify if column is valid / sensitive**
  2. **Annotate sensitive columns with column keys**



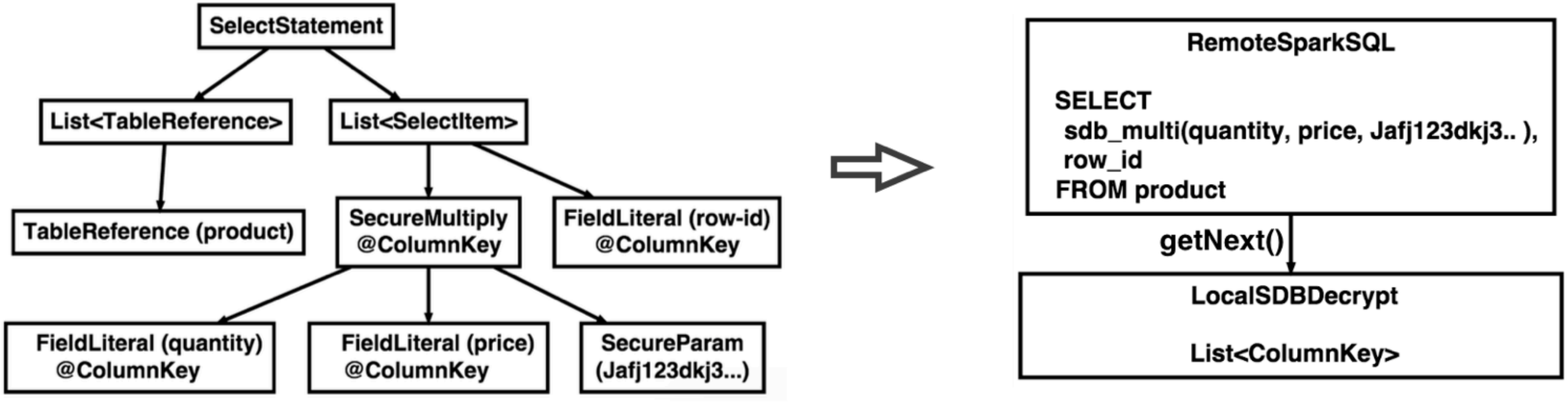
Query Rewriter

1. Identify and rewrite secure operators



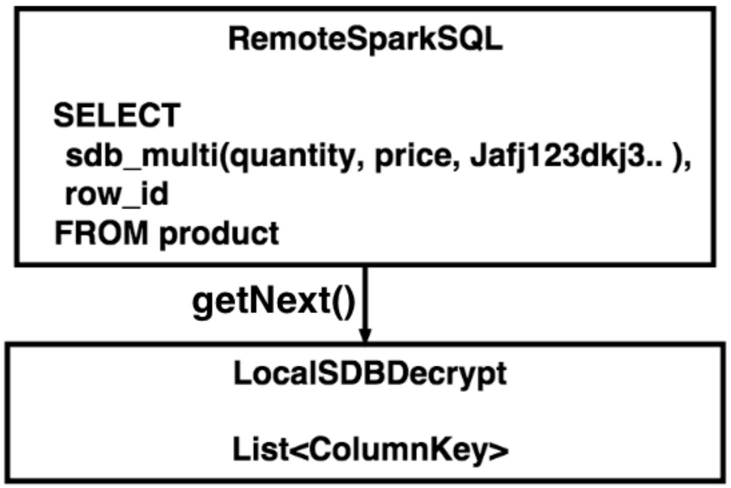
Query Rewriter

2. Transform logical plan trees into physical plan trees



Query Executor

1. Submit rewritten queries to SparkSQL
2. Decrypt encrypted results
3. Return plaintext results via connector



Overview

1. The Problem
2. Related Work
3. Theoretical Background
4. System Architecture
5. Component Implementation
6. **Experiment Result**

Security Analysis

Security threats

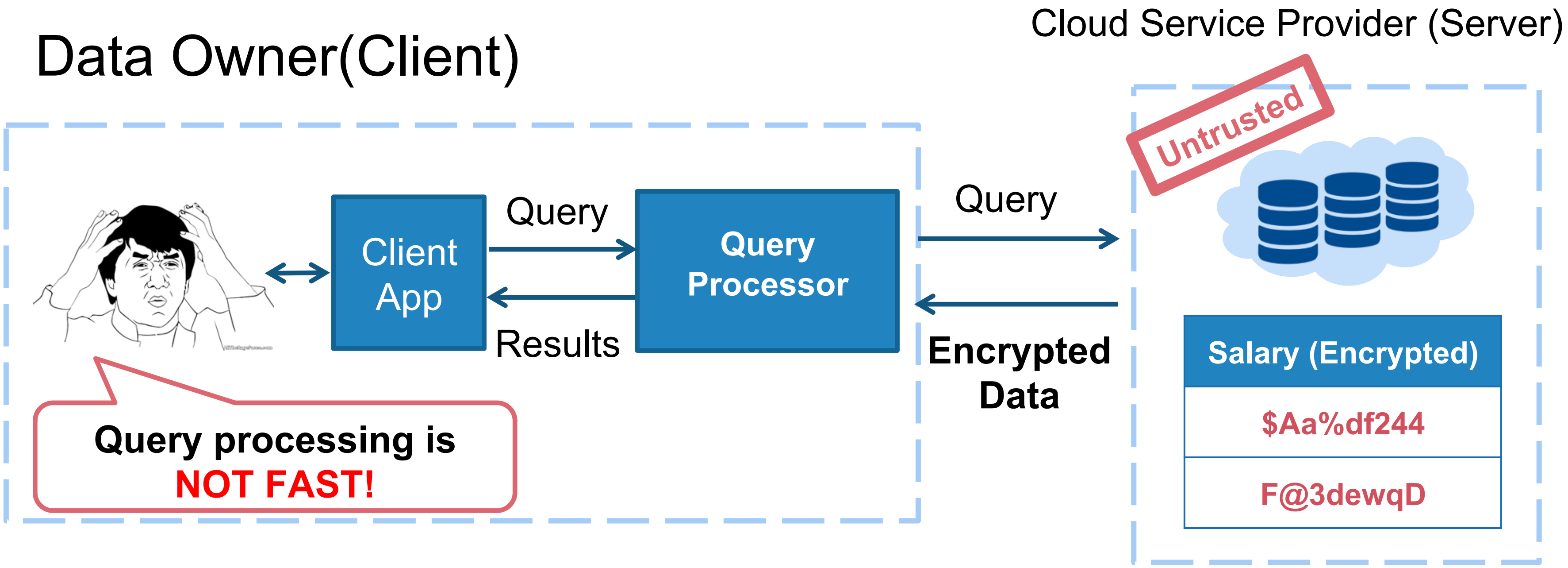
- **Database (DB) Knowledge** – See encrypted values stored on servers’ disks
- **Chosen Plaintext Attack (CPA) Knowledge** – Select plaintext values and observe encrypted values
- **Query Result (QR) Knowledge** – See queries submitted and the encrypted results

Security Analysis

Security Level in SDB

- SDB generates 2048-bit column keys similar to RSA
- SDB is secure against DB + CPA threat and DB + QR threat
- Limitation: secret sharing doesn’t support floating point numbers

Decrypt-Before-Query Approach



Importance of Secret Sharing

- Compare with Decrypt-before-query(DBQ)
- Experiment Environment
  - Client: 1 CPU
  - Server: 8 CPU X 10 Machines
- Result
  a. Total Cost: SDB < DBQ
  b. Client Cost: SDB << DBQ

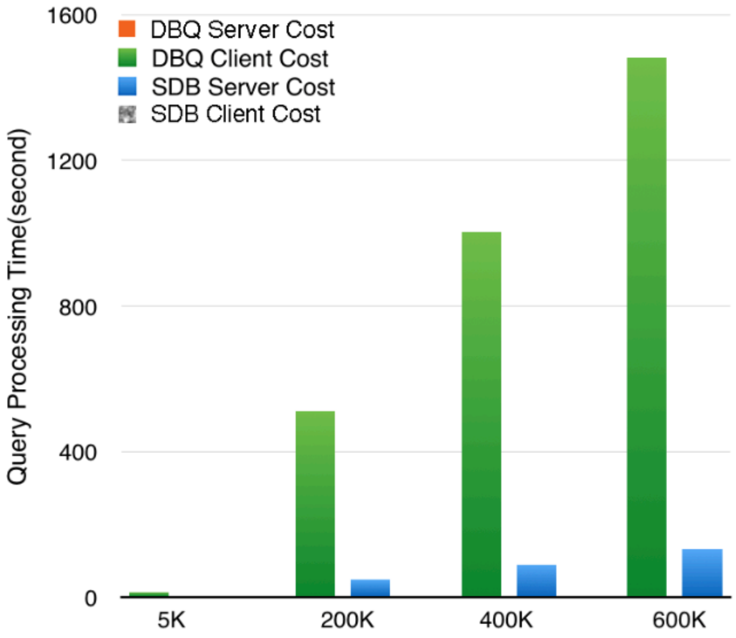


Figure 17: DBQ vs. SDB

SELECT A, B FROM T WHERE A < p, 1% selectivity

Query Cost Breakdown

- Server cost >> client cost
- Decrypt cost >> other client cost

- Future work: Encryption/Decryption optimization

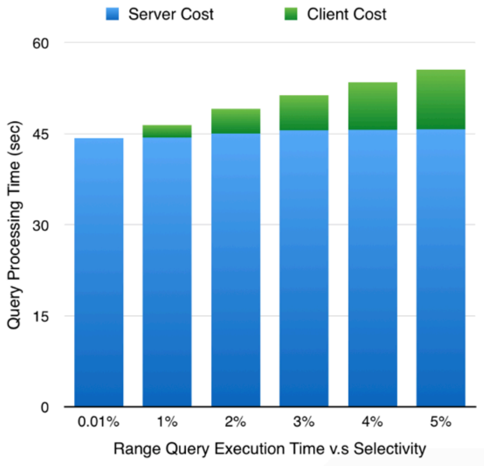


Figure 18: Client/Server Cost v.s. Selectivity

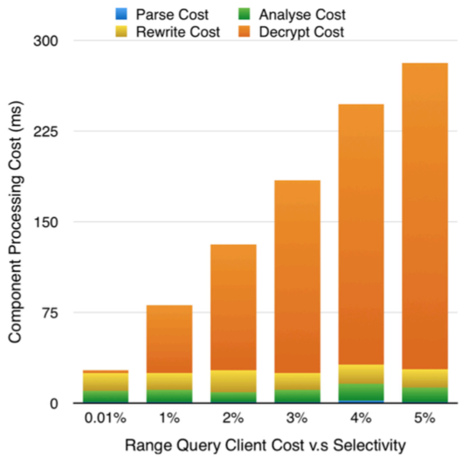


Figure 19: Client Costs v.s. Selectivity

SELECT A, B from T WHERE A < q

Overhead of Secure Operators

- Compare with SparkSQL
  - Execute on plaintext, bypassing all secure operators
  - Three types of queries
    - EC Range: SELECT A, B FROM T WHERE A < 100
    - EE Range: SELECT A, B FROM T WHERE A < B
    - Count: SELECT count(A) FROM T WHERE A < 100

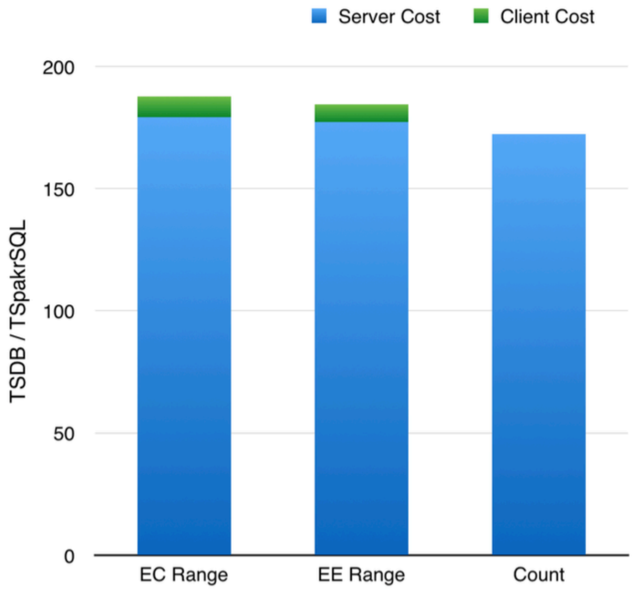
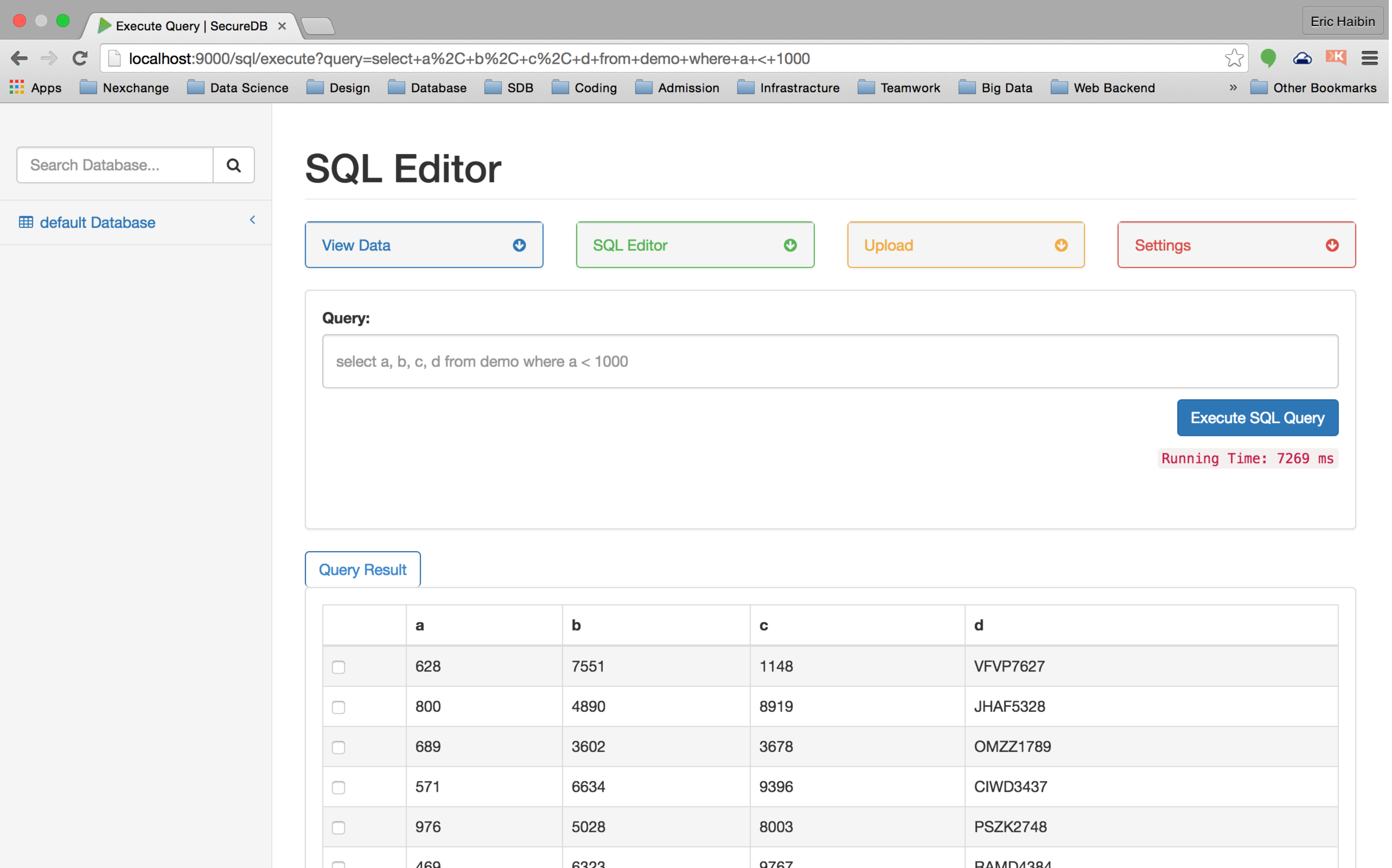


Figure 20: TSDB / TSpark

- Result
  - ~180 times slower

$$b^r \bmod n$$

  - Computation cost of modular exponential is high
  - Future work: UDF optimization


Execute Query | SecureDB
Eric Haibin
localhost:9000/sql/execute?query=select+a%2C+b%2C+c%2C+d+from+demo+where+a+<+1000
Apps
Nexchange
Data Science
Design
Database
SDB
Coding
Admission
Infrastructure
Teamwork
Big Data
Web Backend
Other Bookmarks
Search Database...
default Database
SQL Editor
View Data
SQL Editor
Upload
Settings
Query:
select a, b, c, d from demo where a < 1000
Execute SQL Query
Running Time: 7269 ms
Query Result
a b c d
628 7551 1148 VFVP7627
800 4890 8919 JHAF5328
689 3602 3678 OMZZ1789
571 6634 9396 CIWD3437
976 5028 8003 PSZK2748

Execute Query | SecureDB

Eric Haibin

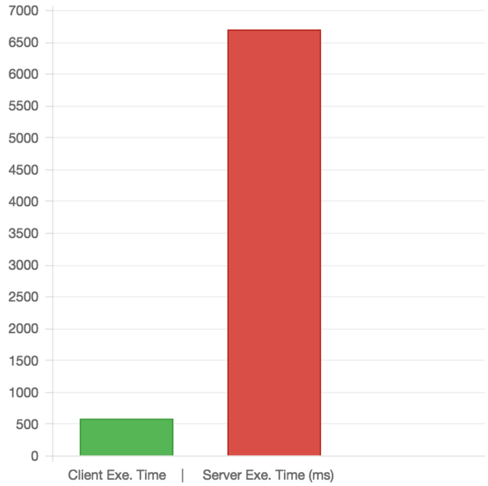
localhost:9000/sql/execute?query=select+a%2C+b%2C+c%2C+d+from+demo+where+a+<+1000

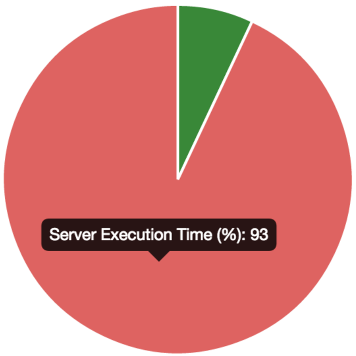
Query Execution Analysis

SQL Executed on remote cluster server:

```
select demo.a, demo.b, demo.c, demo.d, demo.row_id from demo where
sdb_compare(sdb_keyup(sdb_mul(demo.r,sdb_add(demo.a,demo.s,demo.s,"hh1lccfur8j8bzursih3yyk80ktcvem58dzl05pj3rxit8wa8tn7snsr6v
"gm2ulme39lzl0uap8q22fwqhbolhdry79ge2ofv6dorvwnjetceppfib16s5wg9iill0nvsp1zyjfczgj0471n9gy9wh2tex33ufaedtv3v017wkqj6o6zw5ziif
< 0
```

Total number of results: 486





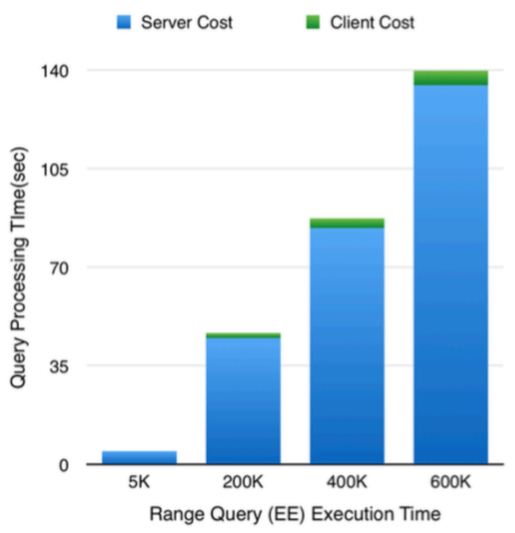
Client Parse Time: 5 ms Client Analyse Time: 21 ms
Client Rewrite Time: 34 ms Client Execute Time: 517 ms
Server Execute Time: 6692 ms

Future Work

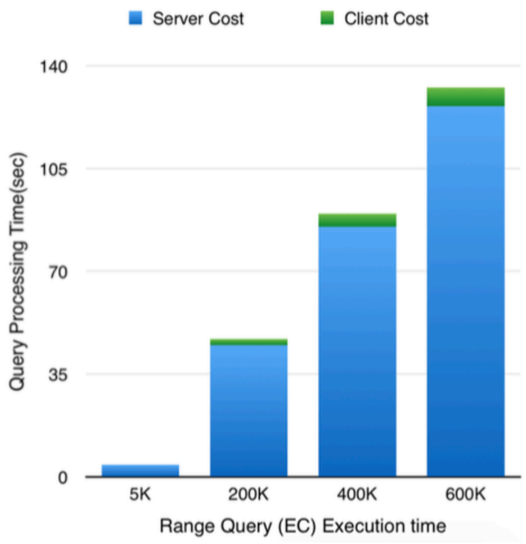
- Query expressiveness extension
  - Join, Cartesian product, SUM(), AVG()
  - GroupBy, Having Clause
- Crypto optimization
  - Encryption/Decryption optimization
  - UDF optimization

Q&A

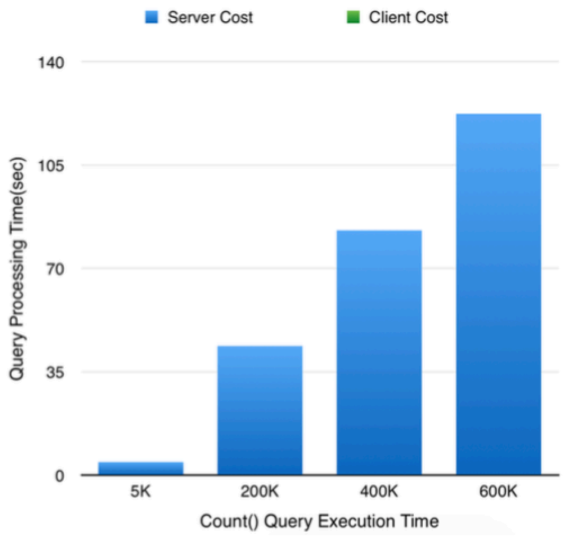
Query Cost vs. Data Size



SELECT A, B from T WHERE A < q



SELECT A, B from T WHERE A < B

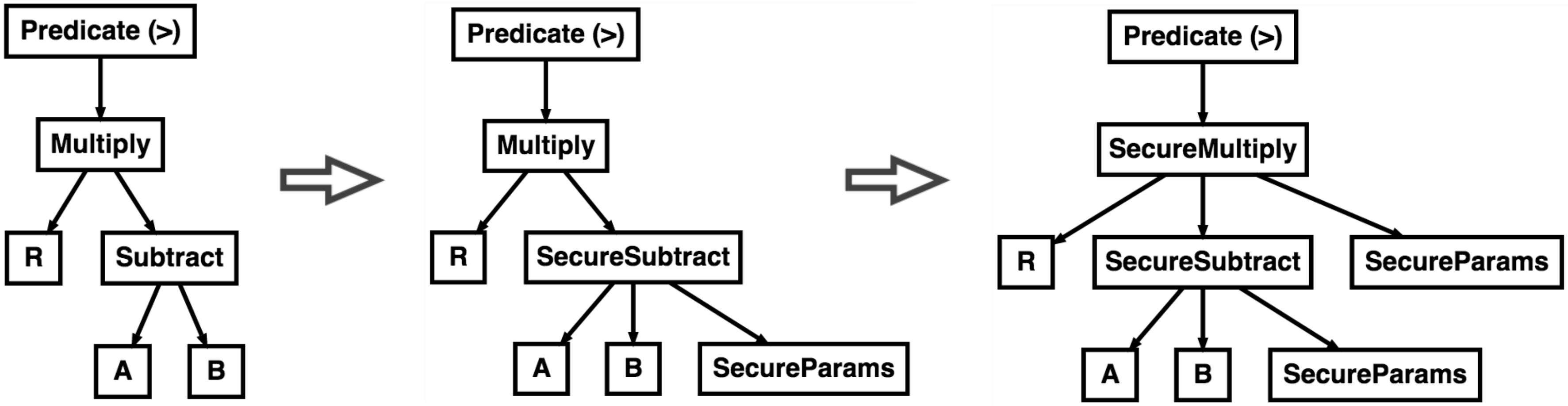


SELECT COUNT(A) from T WHERE A < q

More on Query Rewrite

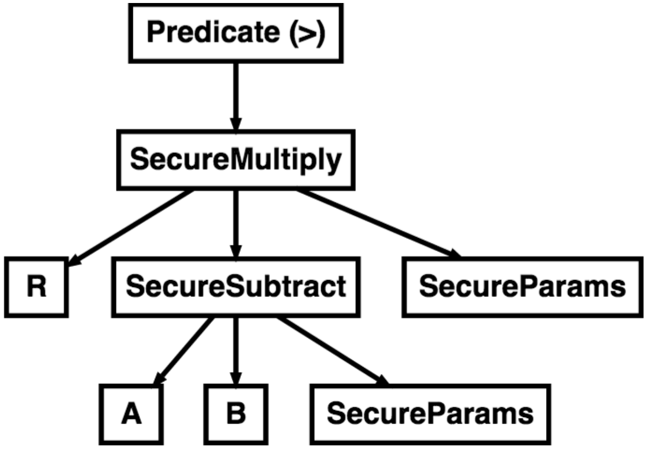
- What if multiple secure operators are involved?

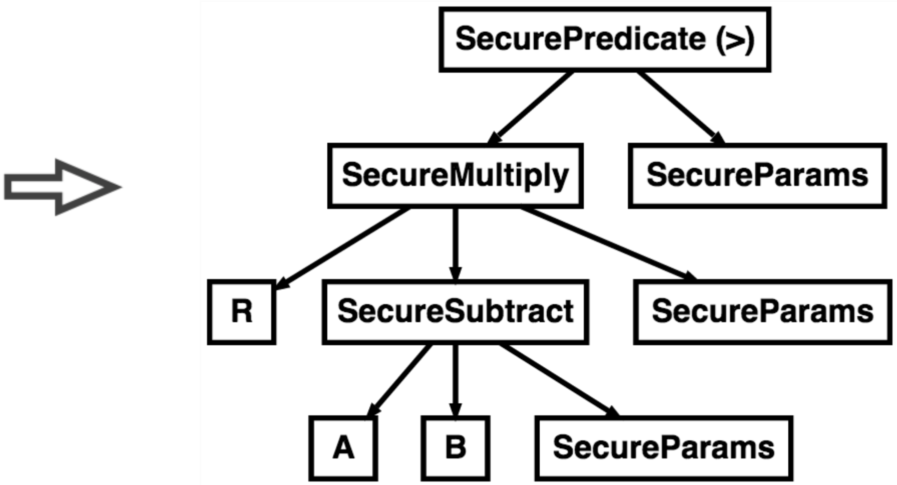
R * (A - B) > 0



More on Query Rewrite

- What if multiple secure operators are involved?





sdb_compare(sdb_keyup(sdb_mul(r, sdb_add(a,b, ..), ..), ..), ..)

Demo Video

Reference

[1] Bajaj, S., & Sion, R. (2014). TrustedDB: A Trusted Hardware-Based Database with Privacy and Data Confidentiality. Knowledge and Data Engineering, IEEE Transactions on, 26(3), 752-765. Chicago

[2] Gentry, C., & Halevi, S. (2011). Implementing Gentry's fully-homomorphic encryption scheme. In Advances in Cryptology–EUROCRYPT 2011 (pp. 129-148). Springer Berlin Heidelberg.

[3] Popa, R. A., Redfield, C., Zeldovich, N., & Balakrishnan, H. (2012). CryptDB: Processing queries on an encrypted database. Communications of the ACM, 55(9), 103-111.