*i*HOM: How to Enrich SQL Queries in CryptDB

Hassan Mahdikhani (PhD Student) and Rongxing Lu*

Contact Email: rlu1@unb.ca

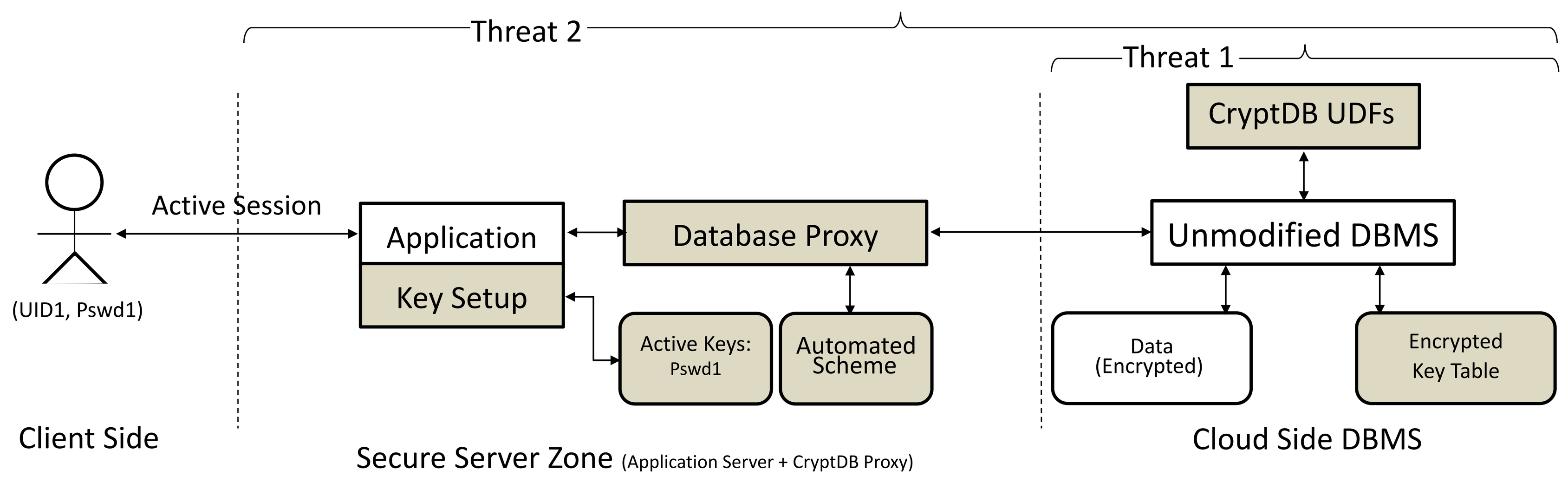
Canadian Institute for Cybersecurity (CIC), University of New Brunswick (UNB)

ABSTRACT

Significant research studies have been conducted on implementing end-to-end database encryption which tries to convince users that privacy and security concerns are addressed and resolved. To satisfy this requirement, homomorphic encryption allows computations to be done on ciphertexts without exposing and decrypting them and aims to protect the security, confidentiality and privacy of computing components. In this paper, we propose a new *iHOM* scheme that supports the commonly used arithmetic operations in database queries, i.e. multiplication and addition. The proposed scheme will be applied on CryptDB, the most arguably well-known encrypted database system that can process SQL queries over encrypted data to provide practical and provable confidentiality. Since current CryptDB has a *HOM* scheme based on Paillier Homomorphic encryption, the multiplication queries can not be performed at all. Thus, the new improved scheme, *iHOM*, will enhance existing capabilities of CryptDB to run queries that contain both additive (+) and multiplicative (*) operations in a homomorphic way.

Architectural Layering Model of CryptDB

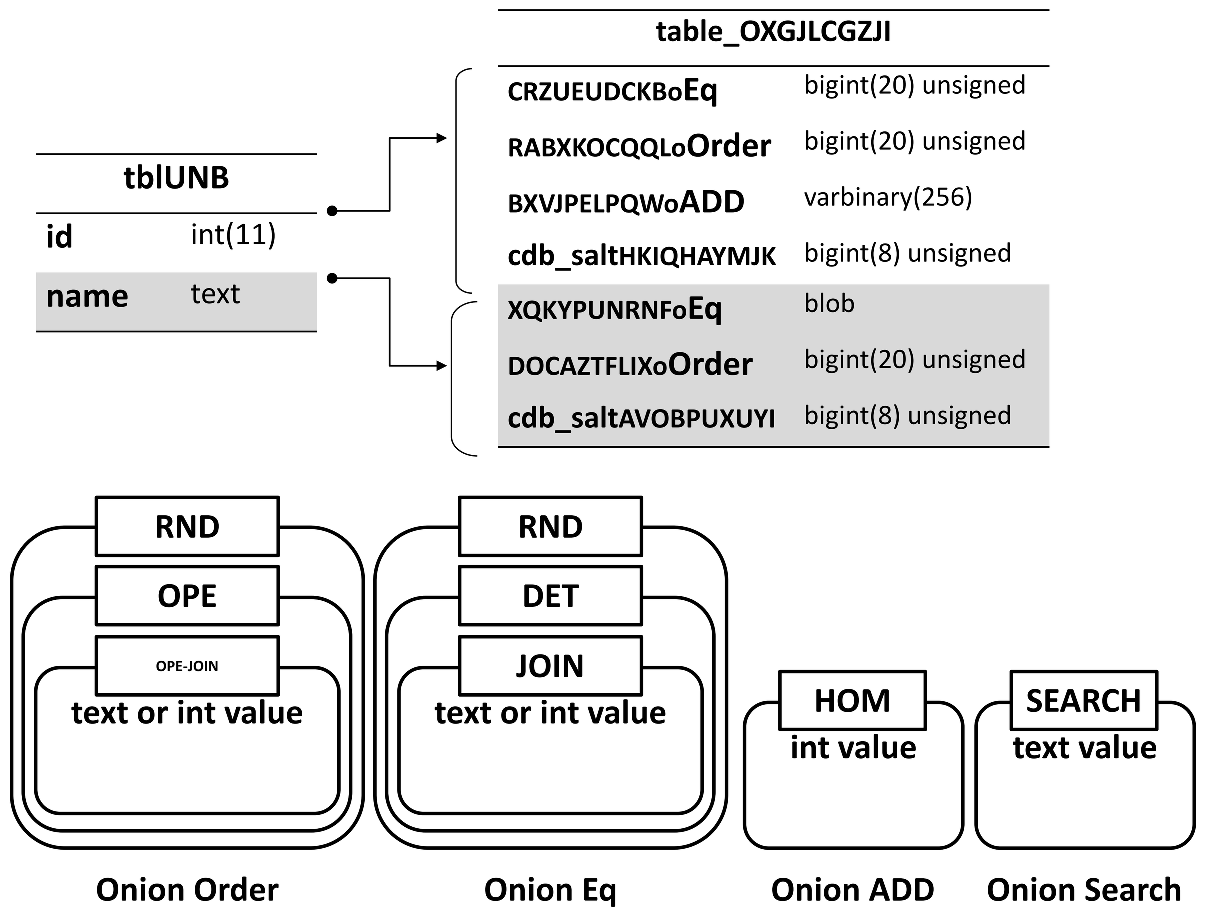
- **User-side interface** (**Client Side**), where the desktop or web application would be able to interact with encrypted database transparently.
- **CryptDB proxy** (**Secure Server Zone**) stores master keys,generates user database schema and maintains the current status of encryption layer for each table columns.
- **Unmodified DBMS** provides conventional data management services over encrypted data no architectural modifications are applied and DBMS would operate like a typical one.



Threat Management Model

- **Threat Management 1**: By executing SQL commands over encrypted data and SQL-Aware Encryption strategy not only direct access to the physical memory of running systems or virtual machines but also access to cloud side DBMS by honest-but-curious database administrators will not result in major concerns.
- **Threat Management 2**: As CryptDB makes use of different keys for different users and data items, it is difficult for any type of attackers to achieve significant data breach,

Onions of Encryptions in CryptDB



SELECT * FROM tblUNB WHERE id=12;

→ Proxy Parser →

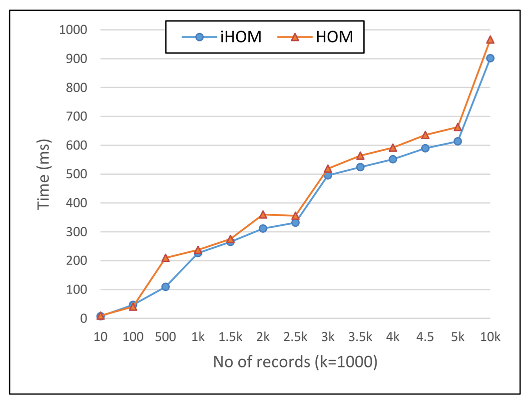
SELECT * FROM table_OXGJLCGZJI WHERE CRZUEUDCKBo Eq = x27c3e

SHE Symmetric Homomorphic Encryption

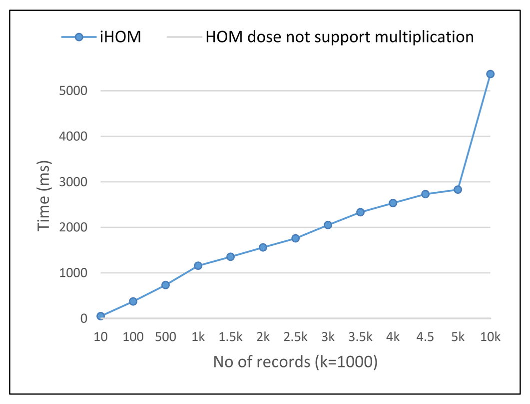
- **Key Generation:** KeyGen(λ) is a probabilistic function that generates secret key SK = (s, q) and public parameter p, i.e., (s, q, p) ← KeyGen(λ)

  Where p, q are two prime numbers, p>>q and |p|=λ, the length of q also relies on some security parameter, and s is a number randomly picked from Z^*_p.
- **Encryption:** $c = Enc(SK, m, d, r) = s^d(rq + m) \bmod p$. / **Decryption:** $m = Dec(SK, c, d) = \left(c \times s^{-d} \bmod p\right) \bmod q$.
- **Homomorphic properties:**
  - **Addition:** Given two ciphertext $c_1 = Enc(m_1) = s^{d_1}(r_1 q + m_1) \bmod p$ and $c_2 = Enc(m_2) = s^{d_2}(r_2 q + m_2) \bmod p$, when $d_1 = d_2 = \alpha$, $(r_1+r_2)q < p$ and $m_1 + m_2 < q$, we have have $E(m_1; r_1) \cdot c_1 + c_2 = Enc(m_1 + m_2)$. or simplicity, we omit the random items, and we have $E(m_1) \cdot E(m_2) = E(m_1 + m_2)$.
  - **Multiplication:** Given $c_1 = Enc(m_1) \bmod p$ and $c_2 = Enc(m_2) \bmod p$, we have $c_1 \times c_2 \bmod p = Enc(m_1 \times m_2)$.
  - **Scalar Multiplication:** For a ciphertext $c_1 = Enc(m_1) = s^{d_1}(r_1 q + m_1) \bmod p$ and a message $m_2 \in Z_q$ we have $c_1 \times m_2 \bmod p = Enc(m_1 \times m_2)$.

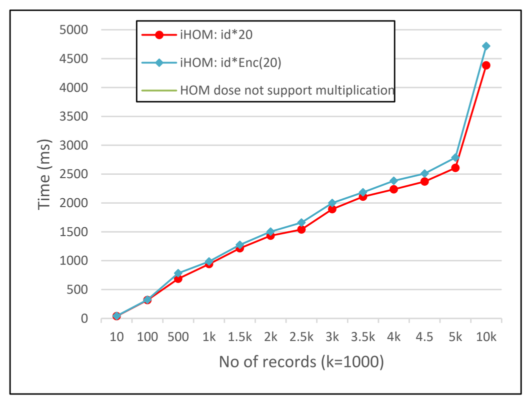
Performance Evaluation



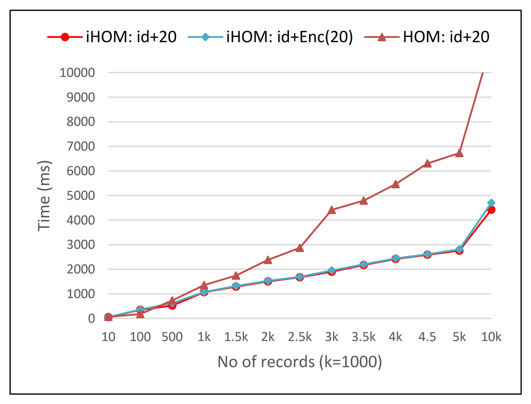
SUM(id)
Aggregation



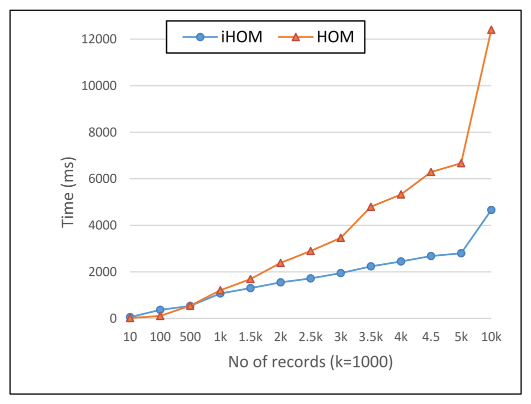
id × id
HOM dose not support multiplication.



id × m(plaintext)



id + m (plaintext)



Id + id

Proposed *i*HOM vs HOM

Query	HOM	*i*HOM	Description
id1 + id1	✓	✓	id1 is a table column.
id1 + id2	×	✓	Distinct columns in a same table.
id1 + m	✓	✓	id1 is a table column & m is scalar.
id1 + Enc(m)	×	✓	Complex expressions (e.g. id+20×id).
id1 × id1	×	✓	
id1 × id2	×	✓	
id1 × m	×	✓	
id1 × Enc(m)	×	✓	
SUM(id1)	✓	✓	Aggregate function.

Conclusion

This study investigate the inability of *HOM*, to perform multiplications and mixed computational SQL tasks, afterwards an efficient symmetric homomorphic encryption (SHE) scheme that covered the deficiency of CryptDB to execute multiplication queries has been utilized to implement proposed *iHOM* to enhance the the abilities of CryptDB. Extensive experiments have been made to measure how much iHOM has improved the ability and performance of CryptDB.

Future work

The research on CryptDB should continue and improvements to resolve the deficiency issues of CryptDB could be achieved in future studies such as executing complex computational expressions which consist of addition, multiplication and subtraction in mixtures of table columns and plain values.