# Processing Over Encrypted Query Data In Internet of Things (IoTs) : CryptDBs, MONOMI and SDB

G. Ambika[1],
PG & Research Scholar[1], Department of Computer Science,
Marudupandiyar College(Affiliated to Bharathidasan University), Thanjavur - 613 403, Tamilnadu,India.

Dr. P. Srivaramangai[2]
Associate Professor[2], Department of Computer Science,
Marudupandiyar College(Affiliated to Bharathidasan University), Thanjavur - 613 403, Tamilnadu,India.

***Abstract:*** Internet of Things (IoT) is the developing technologies that would be the biggest agents to modify the current world. Machine-to-machine communications perform with virtual, mobile and instantaneous connections. In IoT system, it consists of data-gathering sensors various other household devices. Intended for protecting IoT system, the end-to-end secure communication is a necessary measure to protect against unauthorized entities (e.g., modification attacks and eavesdropping,) and the data unprotected on the Cloud. The most important concern hereby is how to preserve the insightful information and to provide the privacy of user data. In IoT, the encrypted data computing is based on techniques appear to be promising approaches. In this paper, we discuss about the recent secure database systems, which are capable to execute SQL queries over encrypted data.

*****

## I . INTRODUCTION

IoT is the most important part for improving communication, because in order to interconnect different devices and it could be able to communicate with each other. Security and privacy is proven one of the most challenging areas in IoT. Even though, the aspects of oT cryptography and security are not completely defined. Different and Alternative and definitions have been recommended by both research sectors and industrial parts. Size and Heterogeneity are two major factors that describe the IoT technologies. All other properties, like maneuvering, sensing, storing, being able to capture, and process data are unnecessary manner; if it is not working with your device particularly needs one of these properties. However, the ability to communicate is essential when labeling a device as an IoT device. Security is necessary to implement efficiently by using protocols and algorithmic schemes, in a great number of applications, and in different devices. In other case, it is able to adopt IoT services not possible at large scales.

The usability of IoT technologies should be kept as high as possible. In this way, applied methodologies have to be improved, in order to help scalability and heterogeneity. The methodology is developed to take care of personal data protection and to protect user's anonymity. IoT supports the challenge of security that can also implicate a great principle for trust by the usability of both services and applications. Cyber attacks are recorded almost day-to-day life; mainly happen due to the low security of the services, applications and devices [1]. At last, we discuss end points are proven also in terms of privacy, weak designed.

### Encrypted Query Processing

A DBMS is to update or retrieve the exact data to/from the physically stored medium. The desired information is determined in a reliable manner that is the scientific art of Query Processing from a database system. Database system should be able to respond to requests for information from the user i.e. process queries [2]. Database security is always obtained the information to the user securely when requires user queries. Database security has been provided by operating system security and physical security. These methods provide a secure support for processing and storing the sensitive data. Cryptographic support is a vital part of database security [3]. Database is getting down while several organizations cannot work properly, so that they require its protection. The confidential data are securely stored and protected in a repository database. Likewise, the information should not fail into the unauthorized hands of those who misuse it [4]. Efficient algorithms are necessary part as protected that provides the ability to allow encryption/decryption of data and query over encrypted database query.

A plaintext SQL query is sent to an encrypted query where the cloud cannot learn about the values in the query by performing an encrypted query processing system. After the query process is executed over encrypted data and thereafter the encrypted result is transmit to the user side. A medical data system considers as an example. If the heart rate is larger than 200 after that it transmitted into an encrypted query, where 200 is replaced with its order-preserving encryption and then the heart rate with the encrypted column name:

SELECT * FROM tab WHERE heart rate > 100

Encrypted query,

SELECT * FROM tab WHERE column-043 > 0x19

In fact, the encrypted data is operated by the encrypted database (EDB) systems. It has the database queries based on mathematical operations and then has motivated researchers to create the encrypted database (EDB) systems. Internal structure of DBMS is not modified because of Crypt-DB proposes a database proxy layer to encrypt and decrypt data, namely Crypt-DB utilizes native DBMS [5]. After that it proposes three main key ideas. They are,

- Initial to execute SQL query on encrypted data
- Second one to adopt adjustable encryption strategies for different queries
- Finally to chain encrypted keys to user passwords.

Crypt-DB hardly supports queries including analytical query and computation process [6]. For that reason, Crypt-DB which can process the large data set and complex analytical query because of that MONOMI is established based on the design of Crypt-DB. The analytical load is more difficult to execute the encrypted data on the server; MONOMI proposes an executing technique to split between the client and server. Different encrypting algorithms are utilized to different queries by using different from the Crypt-DB and MONOMI. Group of security operators (e.g.,$\times$ , $\pm$, $\pi$, $\oplus$, $\bowtie_S$ ) is realized a secure query processing system(SDB) referred in [8] with data interoperation which can competently maintain a quantity of difficult SQL query involving whole TPC-H benchmark queries on the server. The encrypted query system that hardware encryption is better than software encryption [7].

## II. ENCRYPTION SCHEME FOR EQP

The following encryption schemes can be utilized for the encrypted query processing.

### a. Random (RND)

The RND scheme provides powerful security assurances: It is probabilistic, that means the same plaintext can be encrypted to a different cipher-text. On the other hand, it does not allow the feasible computation process in a reasonable amount of time. For highly confidential data seems like medical diagnosis, private messages or credit card numbers that cannot required to be compared with other entries for equality. Random implementation exploits Advanced Encryption Standard (AES) to encrypt strings and the Blowfish to encrypt integers. Both implementations make use of the Cipher Block Chaining (CBC) mode with a random Initialization Vector (IV) and ability under chosen-plaintext attack (IND-CPA) secure by the previous author’s research view [9]. Size of blocks has the two ciphers. Blowfish has a block size of 64 bit, whereas AES is used with a block size of 128 bit [10]. After using Blowfish needs to store integers and AES only needs half of that space.

### b. Deterministic (DET)

Deterministic encryption permits for equality checks that it can perform chooses with, equality joins, equality predicates, COUNT, GROUP BY, DISTINCT, etc. The plaintext *m* results are encrypted into the same cipher-text *c*. AES in electronic codebook (ECB) mode is a block-cipher encryption with such a property. Because of this deterministic property, it is in general advised ECB not only use for encryption of bulky packets. An attacker has to modify the order of the blocks or replace a block in an indistinguishable manner (i.e., substitution attack), or learn information about the plaintext with a histogram of repeated blocks. AES-CBC is applied twice on the input, where AES-CMC is a tweaked combination of AES-CBC with a zero initialization vector. The second CBC round is initiated in the reverse order, i.e., from the last block to the first block. By this way, the first blocks become deterministically random and do not leak equality within a data item. As a result, AES-ECB should be only used for plaintexts smaller than or equal to 16 bytes but maximum security in DET. AES in CMC mode can be utilized for large plaintexts [16].

### c. Homomorphic Encryption (HE)

Homomorphic encryption (HE) HE performs arbitrary arithmetic operations over cipher-texts without decryption [13] to provide semantic security. As an example, with an additive HE scheme, for two encryptions E(x) and E(y), there exists a function f like f(E(x),E(y)) = E(x + y). Totally homomorphic encryption (FHE) is prohibitively slow process and needs computing power that it cannot be utilized nowadays. El Gamal's [11] and Paillier’s [12] are examples of PHE schemes. For example, with Paillier's PHE, the product of two encryptions encrypts the sum of the encrypted values, i.e., E(x) X E(y) = E(x+y). Partially homomorphic encryption (PHE) is efficient for specific operations and may be utilized in practice. PHE performs either multiplication or addition over cipher-texts and guarantees semantic security.

### d. Join (JOIN, OPE-JOIN)

JOIN and OPE-JOIN schemes are signified both “sub schemes” of DET respective of OPE. The JOIN and OPE-JOIN schemes are determined the computational abilities of their “parent schemes” (i.e. to check whether a plaintext a is equal to plaintext b, respective identifying the order of the column entries). This type works represented over multiple columns and allows to check whether a plaintext in column a is equal to a plaintext in column b for JOIN, if a plaintext in column a is smaller or bigger than a plaintext in column b for OPE-JOIN. Both operators work with multiple columns allowing for constructs like: SELECT * FROM test_table WHERE name1=name2 AND name2=name3. All three name columns consequence that the same plaintext will be encrypted in the same cipher-text across all three columns. Hence, it is more revealing than OPE or DET alone.

### e. Order-preserving encryption (OPE)

Order comparison is a general operation such as for ranking, range checks, sorting in SQL-like databases. The plaintext inputs $m1$, $m2$, and $m3$ has preserved the order relationship between them, after encryption, i.e., if $m1 \leq m2 \leq m3$, then $c1 \leq c2 \leq c3$. The way of order information among the encrypted data items $c_i$ is revealed, but the data itself is not.OPE schemes is one of the first provably secure approach introduced by Boldyreva. This OPE scheme is computationally intensive as Paillier encryption. The interactive OPE approach by Popa. Solely relies on trades computation and symmetric cryptography overhead for latency (i.e., it includes more communication process) [15]. The mutable order-preserving encoding (mOPE) is referred by utilizing lightweight OPE scheme, as the order encodings are mutable. mOPE fulfills the ideal security (IND-OCPA) prove by Popa, after that no additional information than the order is revealed. mOPE is highly secure compare than any other OPE approach and, however, magnitude less computationally intensive than traditional OPE schemes. Even though mOPE appears to be more suitable secure for the IoT and that the status of opportunistic connectivity might have an impact on the performance of this protocol.

### f. Word search (SEARCH)

SEARCH [14] is used to perform searches on encrypted text to support operations like MySQL's LIKE operator. The text divided into keywords by using standard delimiters (or using a special keyword extraction function identified by the schema developer), for each column requiring SEARCH operation. After that the remove repetitions in these words, the positions of the words are randomly permuted. Then encrypt each of the words using Song (researcher) scheme, padding each word into the same size. The encryption process does not reveal to the DBMS server whether a certain word repeats in multiple rows. However, it leaks the number of keywords encrypted with SEARCH that has nearly as secure as RND. Using a user-defined function, the DBMS server checks if any of the word encryptions is any message match the token. All the server learns from searching is whether a token matched a message or not, and this happens only for the tokens requested by the user. The server would learn the same information when returning the result set to the users, so the overall search scheme reveals the minimum amount of additional information needed to return the result. An adversary may be able to estimate the number of distinct or duplicate words (e.g., by comparing the size of the SEARCH and RND ciphertexts for the same data).

## III. ENCRYPTED QUERY PROCESSING SYSTEMS

### 1. CryptDBs

In CryptDB, it has an encrypted system that affords incontestable confidentiality of these attacks for applications backed by using SQL databases [17]. CryptDB is a library file system dynamically linked whereas installing MYSQL database. It adds new components, besides MYSQL server such as parser, key table, MYSQL proxy encrypted data etc. By using a collection of efficient SQL-aware encryption schemes, it works by executing SQL queries over encrypted data. CryptDB sequences to perform encryption and decryption keys by using the user passwords, so that the data item can be allowed to access the password of one of the users to access that data. Its architecture is shown in Figure 1.A Data Base Administrator (DBA) never gets access to decrypted data, and yet but all servers are compromised, an adversary cannot decrypt the data of any user who is not logged in [18]. CryptDB performs by interrupting all SQL queries in a database proxy. It has rewrites queries to execute on encrypted data (CryptDB assumes that all queries go through the proxy). The proxy encrypts and decrypts all data, and modifies various query operators, while preserving the semantics of the query. In DBMS server process, it may never receive decryption keys to the plaintext, so that it never notices sensitive data. DBMS server compromises the procedure and then developers explain their SQL schema to describe different principals, whose keys can allow decrypting different parts of the database. To provide encryption keys to the proxy, they also make a small change to their applications. The proxy determines which parts of the database ought to be encrypted under which key. Therefore, CryptDB guarantees the data confidentiality to users that are not logged in during a compromise (e.g., user 2 in Figure 1).

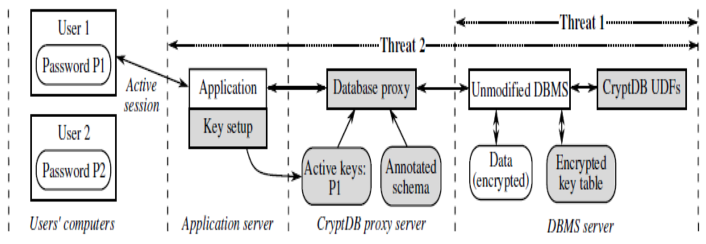


**Figure 1: CryptDB architecture [20]**

### 2. MONOMI

MONOMI is an advance system that runs analytical queries over encrypted data in large databases. Instead of shipping large pieces of encrypted data retransmit from the server. MONOMI evaluates queries on encrypted data at the database server as much as is practical, without providing the information server allowed to access the decoding keys. MONOMI is that the earliest system that may with securely and improve efficiency to execute analytical workloads over encrypted data. MONOMI [19] introduces split client/server query execution depend on CryptDB to help encrypted data under arbitrarily complicated queries. Additionally, a number of techniques that improve performance certainly types of queries, including spec-efficient encryption, pre-row computation, grouped homomorphic addition, and pre-filtering, are introduced. MONOMI architecture is shown in Figure 2, where MONIMI prototype is classified into three major components, designer, ODBC library, and encryption database.

- Designer- trusted client machine but un-trusted server during system setup.
- ODBC library- unmodified SQL queries during normal operation and uses the *planner* to determine the best

split client/server execution plan for the application's query.

- Encryption database- final point, library issues handle one or more queries to the encrypted database.

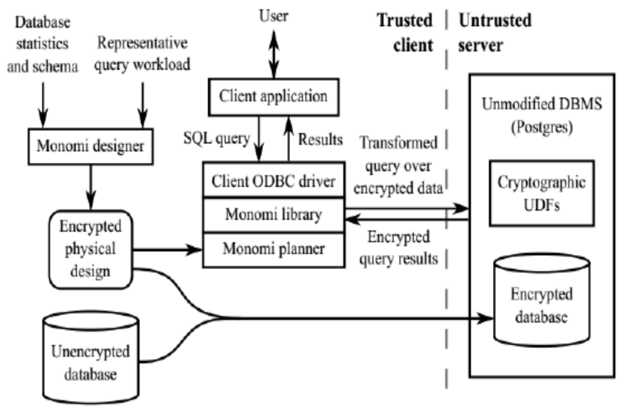


**Figure 2: MONOMI architecture**

### 1. SDB

SDB [22] provides a database proxy between the service provider and data owner, composed of database parser, rewriter, connector, and executor. In Query rewriter process, it has communicated with secret metastore for rewriting. The Connector is an interface where applications or users submit query and receive query results. The executor is fully responsible for passing results back to users or applications, handling all execution flow, through the connector interface. SDB submits and rewrites client-side query to SDB proxy parses the query and accesses the secret metastore to check whether if any operation data owner, over and above server-side query to service provider, as defined in the client and server protocols. These queries are rewritten with user-defined functions according to the operators included, like multiplication and key update. When the cloud server concludes operations, encrypted query results are transmitted back to SDB proxy, decrypted to original plaintext values and transmit back to data owner. The system architecture of SDB [21] is clearly given below. The architecture consists of two divisions [23]: (1) a lightweight SDB proxy at the data owner (DO) and (2) a relational engine with a collection of UDFs provided through SDB at the service provider (SP). This new architecture pushes the entire computations return back to the underlying engine through UDFs. Accordingly, SDB now enjoys whole the benefits like parallel-execution, fault-tolerance, and scalability determined by the underlying Spark SQL engine. Two categories of data reside on cloud database: encrypted values of sensitive data and plaintext of non-sensitive data. The data owner reserves the column keys to sensitive data in secret metastore. User-defined functions are registered on both cloud database and client database. At the SP process of the engine is responsible for, 1.Storing the secrete shares of sensitive data and the plaintext values of insensitive data. 2. Processing queries rewritten. 3. Returning back to the SDB proxy of the encrypted results [24].

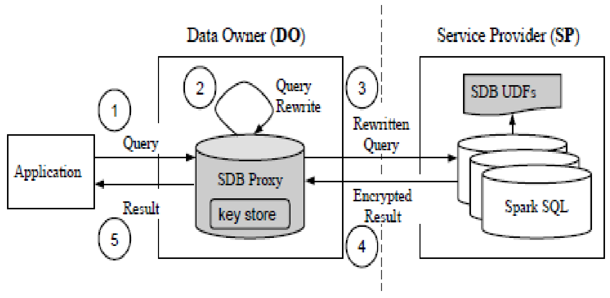


**Figure 3: SDB Architecture[25]**

## IV.EXPERIMENTAL RESULTS

To implement the different experiments perform to setup a lab environment, if two computers are setup to act as a client and a server. Ethernet cable provides 1000 Mbps bandwidth while two computers are directly connected using an Ethernet cable. Presently, two 2 machines runs Ubuntu 12.04 and each machine has Intel processor i7-3770 CPU @ 3.40GHz x 8, with 16GB RAM capacity. One machine runs using the client layer at the same time as the server layer runs another machine; both are written in C++ using the GMP library. The client holds querying records from the database and the server also holds the database to keep the entire information.

To implement the version of platform is on Apache Hadoop 2.4.1, Spark 1.1.0 and Hive 0.12.0 in respective manner and to analyze the different parameters that may affect the overall performance of encrypted query database systems. The parameters perform such as overhead performance, latency of the link, bandwidth, throughput, execution time, distribution of data in the database, size and number of records within the database, query result size and the use of multiple-condition queries. Etc.

The default values and their parameters in the experiments are considered with the performance result like Bandwidth - 1000 Mbps, Latency - 10 milliseconds, Database Size - 50000 records, Query Size - 1000 records, Record Size - 10 bytes, Data Distribution – equivalent performance. When compare to MONOMI database (represented as MDB), SDB and CryptDB using a simple synthetic dataset on that a range of queries are executed. Encrypted database system is to evaluate using query efficiency then illustrates with running time performance, computational overhead and throughput both implement on server side and client side.

To compare the performance of our prototype by running a TPC-H 2.14 benchmark based query over a TPC-H scale 10 data set. The eight kinds of queries are clearly described as below. The complex analytical queries are composed with the support of the TPC-H query workload for performing aggregates over expressions, sub-selects, complex expressions in the FROM, GROUP BY, WHERE, and HAVING clauses. To rapid up the performance of decryption process on the client side is to utilize by using multiple cores and homomorphic multiplication acts on the server side. Decryption process is speed up on the client side by caching the decryptions of repeating cipher-texts. A cache size of 512 elements performs with a random eviction policy. The dataset is a table which denotes T with three sensitive columns such as A, B, and C. In each column is randomly generated the integer values with a uniform distribution over the range [0, 1M]. To execute 8 queries that cover the different secure operators of three encrypted database systems:

Query 1 (Range): SELECT A, B, C from T WHERE A + B < q.
Query 2 (Count): SELECT COUNT (*) from T WHERE A+B < q.
Query 3 (Sum): SELECT SUM (A*B) from T WHERE A+B < q.
Query4 (Range): SELECT A, B FROM T WHERE A < 100
Query 5 (Join): SELECT SUM ($t_1$.B * $t_2$.A) FROM T as $t_1$, T as $t_2$ WHERE $t_1$.A = $t_2$.B.
Query 6 (Sum): SELECT SUM (c3-Hom) AS average FROM DBMS Table2
Query 7 (Range): SELECT A, B FROM T WHERE A < B
Query 8 (Count): SELECT count (A) FROM T WHERE A < 100

**Bandwidth:**

The bandwidth of the link modifies between the client and server that had a direct consequence on the execution time. The execution time reduces from an average range of 0.963 seconds for performing with a 0.5 Mbps link to 0.684 seconds for a 1 Mbps link, the reduction continues also for a 10 Mbps link where the time reaches 0.390 seconds. On the other hand, this point of results increases within no net gain performance, because on this bandwidth range; the latency of the link becomes the only dominant factor affecting time. In figure 4 effects can be clearly seen which depicts the execution time vs. the bandwidth of the link.

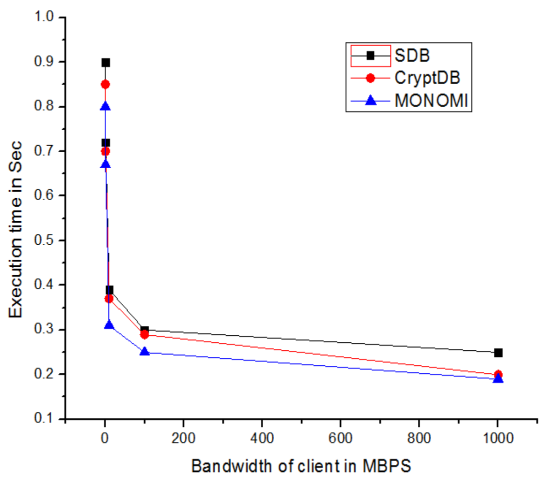


Figure 4: Correlation between the execution time and client bandwidth

**Latency:**

The latency differs in the experiments yielded a pronounced effect on the time required for performing a query to be transmitted. It results to be return back to the client side – from this time forth is known as execution time. To evaluate the execution time by subtracting the time of the first packet then the TCP three way handshake and the final packet carries data corresponding to the query. The client and server is exchanged messages using Roundtrips, sequence of the messages are transmitted within one direction as the number of times and one or more messages are returned as a response. The effect of latency amplifies by the number of roundtrips, because for each roundtrip, the latency value should be added to the execution time. The increase of latency versus the increase in execution time for the three databases as shown in figure 5.2; notice the linear relationship between the two variables.

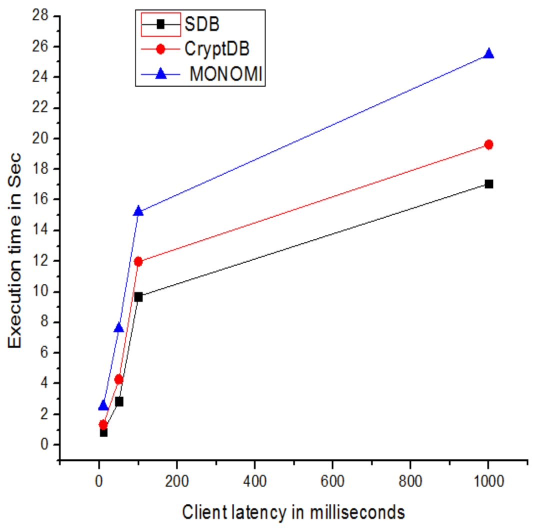


Figure 5.2: The correlation between the client latency and the execution time

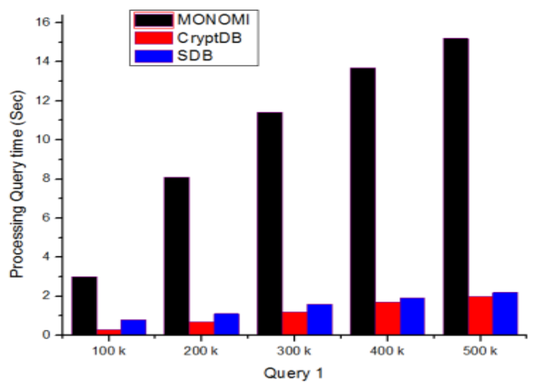
**Execution Time:**

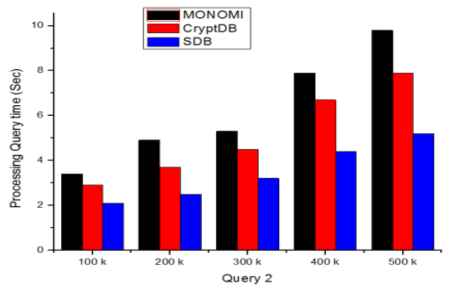
The queries selectivity controls by the parameter q. A smaller q provides a smaller query result and a more selective query. The execution times are compared with SDB, MONONI (MDB) and CryptDB. Three components of the cost are explained as follows.

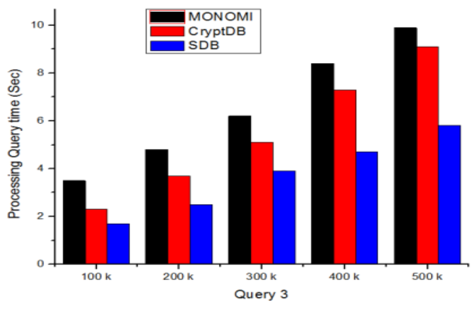
- **Client cost:** the client layer executes at the time taken by using in executing client protocols, any post decryption processing and result decryption performance.
- **Server cost:** the server layer executes at the time taken by using the executing server protocols.
- **DB Access:** the DBMS handle the time taken for processing queries and retrieving the data from the server layer.

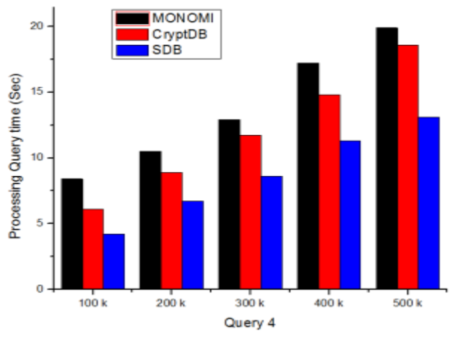
When the table size increases from 100K to 500K rows, figure 6 depicts the results for the eight queries. To observe the performances about figure 6,

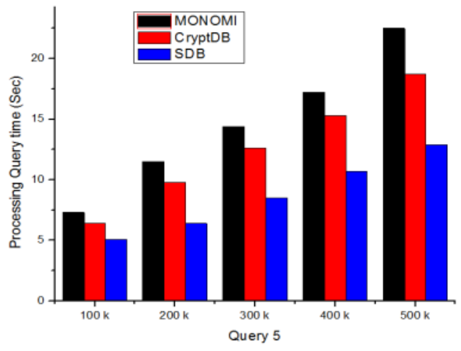
- DB access times take place at very small. Because of about 1% query selectivity, the secure index is moderately effective in filtering away several irrelevant rows.
- The server cost of SDB is faster than that of MDB. Two reasons for handle MDB:
    a. The server is only computing parts of a query for MDB.
    b. MDB utilizes different homomorphic encryption schemes which allow computation on encrypted data to be done in an efficient manner.
- The client cost of MDB is higher than that of SDB for the first three queries, for the reason that the computation involved for evaluating a query and it has to be carried out by the client under MDB. The client cost of MDB is higher than that of CryptDB, as a few of the homomorphic encryption functions.
- SDB and MDB demonstrate comparable total costs for the join query.

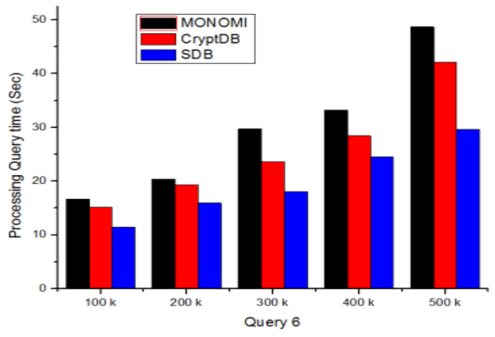


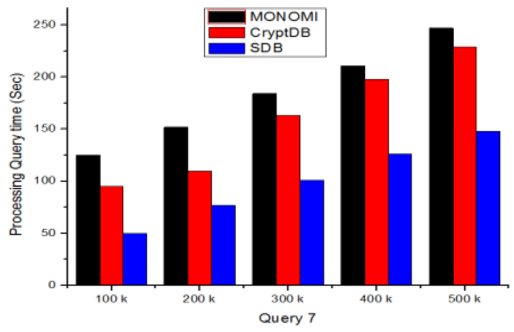












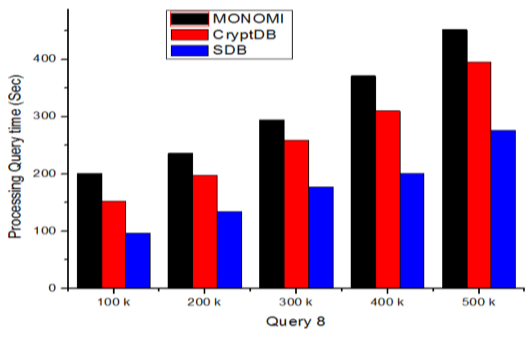


**Figure 6: Execution times of SDB ,MDB and CryptDB for the 8 sample queries**

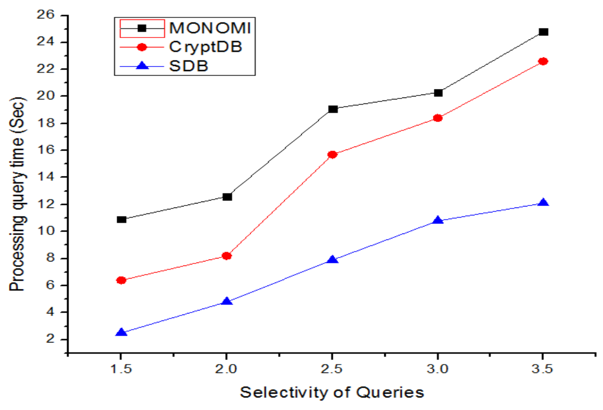


**Figure 7: Execution times vs. selectivity**

In figure 7 depicts the execution times of SDB, CryptDB and MONOMI (MDB) for about the range of query as perform the selectivity of the query that modify (by adjusting the parameter q) from 0 1.5 % (more selective) to 3.5 % (less selective). In Figure 6 observe that the relative costs of the three components remain. Particularly, SDB provides much smaller query execution times as well as most of the costs under SDB are borne by the server.

**Overall efficiency:**

To the most excellent of our facts, MONOMI is the first system that knows how to efficiently execute encrypted data under the TPC-H queries, it difficult to make a head-to-head comparison with state-of the- art schemes for encrypted query processing. For instance, the only TPC-H queries that CryptDB be capable of execute are 2, 4, 1, and 7. However, to provide a few form of comparison, to make a highly developed version of SDB that can able to execute the same TPC-H queries at the same time as MONOMI. This highly developed version of SDB called Secure DataBase, executes as much of the query on the server as possible, using only techniques establish in the original SDB design, and executes the rest of the query on the client side. The MONOMI bars in Figure 8 show the slowdown imposed by MONOMI. The bar SDB in Figure 4 shows that MONOMI outperforms this approach by query execution times.

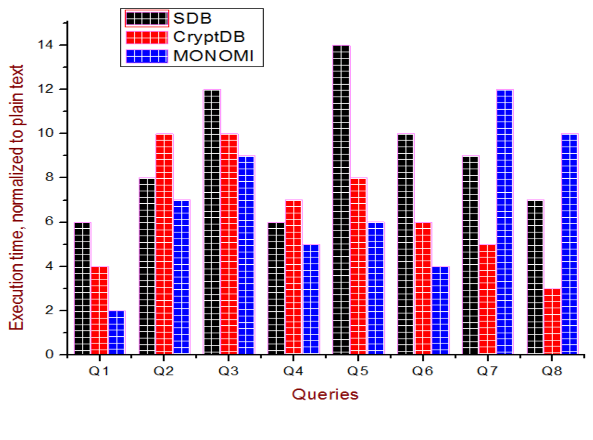


Figure 8: Execution time of TPC-H queries under various systems (8 queries)

**Throughput:**

The throughput of TPC-C queries perform multiple cores on the server differs from one to eight in figure 9. CryptDB's overhead, the server throughput is to measure for eight kinds of SQL queries seen in TPC-H. The results shown in figure 11 for handling MONOMI design, CryptDB, and SDB. The MONOMI carry out each query over data encrypted with RND by decrypting the relevant data using a UDF, re-encrypting the result (if updating rows) and executing the plaintext under the query. As a result of SDB throughput penalty is much enormous for queries that involve a SUM and for incrementing UPDATE statements; HOM addition queries that occupy at the server side. Intended for the other kinds of queries that form a superior part of the TPC-H mix, the throughput overhead is modest. For almost all queries, the MONOMI design performs poor for the reason that the DBMS's indexes lying on the RND-encrypted data are ineffective performance for operations on the plaintext data. It is agreeably astonishing that the higher security of SDB over CryptDB and MONOMI also carries superior performance.

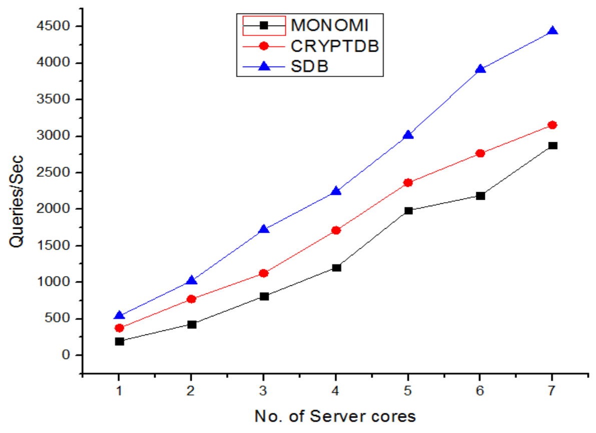


Figure 9: Throughput of different kinds of SQL queries from the TPC-H query mix running under SDB, the MONOMI design and CryptDB.

## V.CONCLUSION

The effort of realizing theoretical cryptographic approaches under real-world conditions sheds light on undiscovered weaknesses and creates opportunities for improvements in cryptosystems. IoT applications need a more suspicious system design due to the challenges of the ecosystem with regard to resource constraints. From this paper reviews, SQL queries execute over encrypted data performing with the encrypted database systems. Compare to encrypted query system with the purpose of hardware encryption database system is much better than software encryption database system like Chiperbase, TrustedDB,

## VI.REFERNCES

[1]. D. J. Abadi, S. R. Madden, and N. Hachem. Column-stores vs. rowstores: how different are they really? In Proc. of SIGMOD, pages 967–980, Vancouver, Canada, June 2008.

[2]. S. Agrawal, S. Chaudhuri, and V. R. Narasayya. Automated selection of materialized views and indexes in SQL databases. In Proc. of the 26th VLDB, pages 496–505, Cairo, Egypt, Sept. 2000.

[3]. A. Arasu, S. Blanas, K. Eguro, R. Kaushik, D. Kossmann, R. Ramamurthy, and R. Venkatesan. Orthogonal security with Cipherbase. In Proc. of the 6th CIDR, Asilomar, CA, Jan. 2013.

[4]. S. Bajaj and R. Sion, "TrustedDB: A Trusted Hardware Based Database with Privacy and Data Confidentiality," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '11), pp. 205-216, 2011.

[5]. S. Bajaj and R. Sion, "TrustedDB: A Trusted Hardware Based Outsourced Database Engine," Proc. Int'l Conf. Very Large DataBases (VLDB), 2011.

[6]. D. Bogdanov et al. A universal toolkit for cryptographically secure privacy-preserving data mining. In PAISI, 2012.

[7]. F. Emek_ci and D. Agrawal et al. Privacy preserving query processing using third parties. In ICDE, 2006.

[8]. C. Gentry. Fully homomorphic encryption using ideal lattices. In STOC, 2009.

[9]. C. Gentry et al. Fully homomorphic encryption with polylog overhead. In EUROCRYPT, 2012.

[10]. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In STOC, 1987.

[11]. M. Bellare, M. Fischlin, A. O"Neill, and T. Ristenpart, "Deterministic encryption: Definitional equivalences and constructions without random oracles," in *Proc. CRYPTO 2008*, 2008, pp. 360-378.

[12]. J. Daemen and V. Rijmen, "Rijndael: The advanced encryption standard," *Dr. Dobb's Journal*, pp. 137-139, 2001.

[13]. E. Damiani, S. D. C. di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati, "Balancing confidentiality and efficiency in untrusted relational DBMSs," in *Proc. the 10th ACM Conference on Computer and Communications Security*, Washington, DC, October 2003.

[14]. Gentry, C. A fully homomorphic encryption scheme. Ph.D. thesis. Stanford University: AAI3382729, Advisor: Dan Boneh. 2009.

[15]. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT '99). ACM, New York, 1999, 223-238.

[16]. Halevi, S. and Rogaway, P. A tweakable enciphering mode. In Advances in Cryptology (CRYPTO '03). ACM, New York, 2003.

[17]. Boldyreva, A., Chenette, N., Lee, Y., and O'Neill, A. Order-preserving symmetric encryption. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT '09). ACM, New York, 2009, 224-231.

[18]. Paillier, P. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. 1999.

[19]. Smart, N., Vercauteren, F. Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes. 2009.

[20]. Snook, M. Integer-Based Fully Homomorphic Encryption. 2011.

[21]. Popa R A, Redfield C, Zeldovich N, et al. CryptDB: protecting confidentiality with encrypted query processing[C]//Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles. ACM, 2011: 85-100.

[22]. Tu S, Kaashoek M F, Madden S, et al. Processing analytical queries over encrypted data[C]//Proceedings of the VLDB Endowment. VLDB Endowment, 2013, 6(5): 289-300.

[23]. Fu Z, Ren K, Shu J, et al. Enabling personalized search over encrypted outsourced data with efficiency improvement[J]. IEEE transactions on parallel and distributed systems, 2016, 27(9): 2546-2559.

[24]. Mattsson U, Blomkvist K. Data type preserving encryption: U.S. Patent 7,418,098[P]. 2008-8-26.

[25]. Popa R A, Zeldovich N, Balakrishnan H. CryptDB: A practical encrypted relational DBMS[J]. 2011.