Scattered, Synchronized and Self Regulating Access to Encrypted Cloud Database

S. SOWMYA LAKSHMI¹, K. BHEEMALINGAPPA²

¹PG Scholar, Dept of CSE, Bheema Institute of Technology and Science, AP, India, Email: ssl92@rediffmail.com.
²Prof & HOD, Dept of CSE, Bheema Institute of Technology and Science, AP, India, Email: csehod@bitsadoni.com.

Abstract: The cloud database as a service is novel paradigms that can support several Internet-based applications, its adoption requires the solution of the information confidentiality problems. We proposed a novel architecture for adaptive encryption of public cloud databases that offers an interesting alternative to the tradeoff between the required data confidentiality level and the flexibility of the cloud database structures at time. We demonstrate the feasibility and performance of the proposed solution through a software prototype. A novel architecture for adaptive encryption of public cloud databases that offers an interesting alternative to the tradeoff between the required data confidentiality level and the flexibility of the cloud database structures at design time. This paper proposes a novel architecture for adaptive encryption of public cloud databases that offers a proxy-free alternative to the system. The project demonstrates the feasibility and performance of the proposed solution through a software prototype. The proposed architecture manages five types of information: plain data represent the tenant information; encrypted data are the encrypted version of the plain data, and are stored in the cloud database; plain metadata represent the additional information that is necessary to execute SQL operations on encrypted data; encrypted metadata are the encrypted version of the plain metadata, and are stored in the cloud database; master key is the encryption key of the encrypted metadata, and is known by legitimate clients.

Keywords: Cloud, Security, Confidentiality, Securedbaas, Database.

I. INTRODUCTION

The cloud computing paradigm is successfully converge as the fifth utility, but this positive trend is partially limited by concerns about information confidentiality and unclear costs over a medium-long term. We are interested in the database as a service paradigm (DBaaS) that poses several research challenges in terms of security and cost evaluation from a tenant’s point of view. Most results concerning encryption for cloud-based services are inapplicable to the database paradigm. Other encryption schemes that allow the execution of SQL operations over encrypted data either have performance limits or require the choice of which encryption scheme must be adopted for each database column and SQL operation. These latter proposals are fine when the set of queries can be statically determined at design time, while we are interested in other common scenarios where the workload may change after the data-base design. In this paper, we propose a novel architecture for adaptive encryption of public cloud databases that offers a proxy-free alternative to the system described. The proposed architecture guarantees in an adaptive way the best level of data confidentiality for any database workload, even when the set of SQL queries dynamically changes. The adaptive encryption scheme, which was initially proposed for applications not referring to the cloud, encrypts each plain column to multiple encrypted columns, and each value is encapsulated in different layers of encryption, so that the outer layers guarantee higher confidentiality but support fewer computation capabilities with respect to the inner layers.

The cloud can hold the user accountable for the data it outsources, and likewise, the cloud is itself accountable for the services it provides. The validity of the user who stores the data is also verified. Apart from the technical solutions to ensure security and privacy, there is also a need for law enforcement. Access control in clouds is gaining attention because it is important that only authorized users have access to valid service. A huge amount of information is being stored in the cloud, and much of this is sensitive information. Care should be taken to ensure access control of this sensitive information which can often be related to health, important documents (as in Google Docs or Dropbox) or even personal information (as in social networking). The use of fully homomorphism encryption would guarantee the execution of any operation over encrypted data, but existing implementations are affected by huge computational costs to the extent that the execution of SQL operations over a cloud database would become impractical. Other encryption algorithms characterized by acceptable computational complexity support a subset of SQL operators. For example, an encryption algorithm may support the order comparison command, but not a search operator. The drawback related to these feasible encryption algorithms is that in a medium-long term horizon, the database administrator cannot know at design time which database operations will be required over each database column.
This issue is in part addressed by proposing an adaptive encryption architecture that is founded on an intermediate and trusted proxy. Cloud computing is the delivery of computing as a service rather than a product, whereby shared resources, software, and information are provided to computers and other devices as a utility (like the electricity grid) over a network. Cloud computing provides computation, software, data access, and storage services that do not require end-user knowledge of the physical location and configuration of the system that delivers the services. Parallels to this concept can be drawn with the electricity grid, wherein end-users consume power without needing to understand the component devices or infrastructure required to provide the service. Cloud computing is different from hosting services and assets at ISP data center. It is all about computing systems are logically at one place or virtual resources forming a Cloud and user community accessing with intranet or Internet. So, it means Cloud could reside in-premises or off premises at service provider location. There are types of Cloud computing like 1. Public clouds 2. Private Clouds 3. Inter-clouds or Hybrid Clouds, say CIO and IT Leaders and expert in cloud computing. Cloud computing has been changing how most people use the web and how they store their files. It’s the structure that runs sites like Face book, Amazon and Twitter and the core that allows us to take advantage of services like Google Docs and Gmail. But how does it work.

Before we dig further into how does cloud computing work, first let’s understand what the term “cloud” refers to. The concept of the cloud has been around for a long time in many different incarnations in the business world. It mostly means a grid of computers serving as service oriented architecture to deliver software and data. Most websites and server-based applications run on particular computers or servers. What differentiates the cloud from the way those are set up is that the cloud utilizes the resources from the computers as a collective virtual computer, where the applications can run independently from particular computer or server configurations. They are basically floating around in a “cloud of resources”, making the hardware less important to how the applications work. With broadband internet, the need to have the software run on your computer or on a company’s site is becoming less and less essential a lot of the software that people use nowadays is completely web-based. The cloud takes advantage of that to bring it to the next level.

II. A NEW ADAPTIVE ENCRYPTION SCHEME

The proposed adaptive encryption schemes with a proxy free architecture. SQL-aware encryption algorithms that guarantee data confidentiality and allow the database server to execute SQL operations over encrypted data. As each algorithm supports of SQL operators, encryption schemes are referred. The encryption algorithms are organized into structures called onions; each plaintext value is encrypted through all the layers of its onions. Besides data confidentiality, the cost is addressed by an analytical cost model and a usage estimation methodology that allow a tenant to estimate the costs deriving from cloud database services. The cloud database service is characterized by plain, encrypted and adaptively encrypted databases over a medium-term horizon during which it is likely that both the database workload and the cloud prices change. Focus on database services and takes an opposite direction by evaluating the cloud service costs from a tenant’s point of view.

III. EXISTING AND PROPOSED SYSTEMS

A. Existing System

The cloud computing paradigm is successfully converging as the fifth utility, but this positive trend is partially limited by concerns about information confidentiality and unclear costs over a medium-long term. We are interested in the Database as a Service paradigm (DBaaS) that poses several research challenges in terms of security and cost evaluation from a tenant’s point of view. Most results concerning encryption for cloud-based services are in applicable to the database paradigm. Other encryption schemes, which allow the execution of SQL operations over encrypted data, either suffer from performance limits or they require the choice of which encryption scheme must be adopted for each database column and SQL operations.

B. Proposed System

The proposed architecture guarantees in an adaptive way the best level of data confidentiality for any database workload, even when the set of SQL queries dynamically changes. The adaptive encryption scheme, which was initially proposed for applications not referring to the cloud, encrypts each plain column into multiple encrypted columns, and each value is encapsulated into different layers of encryption, so that the outer layers guarantee higher confidentiality but support fewer computation capabilities with respect to the inner layers. We propose the first analytical cost estimation model for evaluating cloud database costs in plain and encrypted instances from a tenant’s point of view in a medium-term period. It takes also into account the variability of cloud prices and the possibility that the

![Fig.1.System Architecture.](image-url)
Scattered, Synchronized and Self Regulating Access to Encrypted Cloud Database

database workload may change during the evaluation period. This model is instance with respect to several cloud provider offers and related real prices. As expected, adaptive encryption influences the costs related to storage size and network usage of a database service. However, it is important that a tenant can anticipate the final costs in its period of interest, and can choose the best compromise between data confidentiality and expenses.

C. Implementation Modules
1. Adaptive encryption
The proposed system supports adaptive encryption methods for public cloud database service, where distributed and concurrent clients can issue direct SQL operations. By avoiding an architecture based on one [or] multiple intermediate servers between the clients and the cloud database, the proposed solution guarantees the same level of scalability and availability of the cloud service. Fig.1 shows a scheme of the proposed architecture where each client executes an encryption engine that manages encryption operations. This software module is accessed by external user applications through the encrypted database interface. The proposed architecture manages five types of information:
- Plain data is the tenant information;
- Encrypted data is stored in the cloud database;
- Plain metadata represent the additional information that is necessary to execute SQL operations on encrypted data;
- Encrypted metadata is the encrypted version of the metadata that are stored in the cloud database;
- Master key is the encryption key of the encrypted metadata that is distributed to legitimate clients.

2. Metadata structure
Metadata include all information that allows a legitimate client knowing the master key to execute SQL operations over an encrypted database. They are organized and stored at a table-level granularity to reduce communication overhead for retrieval, and to improve management of concurrent SQL operations. We define all metadata information associated to a table as table metadata. Let us describe the structure of a table metadata. Table metadata includes the correspondence between the plain table name and the encrypted table name because each encrypted table name is randomly generated. Moreover, for each column of the original plain table it also includes a column metadata parameter containing the name and the data type of the corresponding plain column (e.g., integer, string, timestamp). Each column metadata is associated to one or more onion metadata, as many as the number of onions related to the column.

3. Encrypted database management
The database administrator generates a master key, and uses it to initialize the architecture metadata. The master key is then distributed to legitimate clients. Each table creation requires the insertion of a new row in the metadata table. For each table creation, the administrator adds a column by specifying the column name, data type and confidentiality parameters. These last are the most important for this paper because they include the set of onions to be associated with the column, the starting layer (denoting the actual layer at creation time) and the field confidentiality of each onion. If the administrator does not specify the confidentiality parameters of a column, then they are automatically chosen by the client with respect to a tenant’s policy. Typically, the default policy assumes that the starting layer of each onion is set to its strongest encryption algorithm.

4. Cost Estimation of cloud database services
A tenant that is interested in estimating the cost of porting its database to a cloud platform this porting is a strategic decision that must evaluate confidentiality issues and the related costs over a medium-long term. For these reasons, we propose a model that includes the overhead of encryption schemes and variability of database workload and cloud prices. The proposed model is general enough to be applied to the most popular cloud database services, such as Amazon Relational Database Service.

5. Cost model
The cost of a cloud database service can be estimated as a function of three main parameters:

\[
\text{Cost} = f (\text{Time}, \text{Pricing}, \text{Usage})
\]

\text{Time}: identifies the time interval T for which the tenant requires the service.

\text{Pricing}: refers to the prices of the cloud provider for subscription and resource usage; they typically tend to diminish during T.

\text{Usage}: denotes the total amount of resources used by the tenant; it typically increases during T. In order to detail the pricing attribute, it is important to specify that cloud providers adopt two subscription policies: the on-demand policy allows a tenant to pay-per-use and to withdraw its subscription anytime; the reservation policy requires the tenant to commit in advance for a reservation period. Hence, we distinguish between billing costs depending on resource usage and reservation costs denoting additional fees for commitment in exchange for lower pay-per-use prices. Billing costs are billed periodically to the tenant every billing period.

6. Cloud pricing models
Popular cloud database providers adopt two different billing functions that we call linear L and tiered T. Let us consider a generic resource x, we define as \( x_b \) its usage at the b-th billing period and \( p_x b \) its price. If the billing function is tiered, the cloud provider uses different prices for different ranges of resource usage. Let us define \( Z \) as the number of tiers, and \( \{x_1, \ldots, x_{Z-1}\} \) as the set of thresholds that define all the tiers. The uptime and the storage billing functions of Amazon RDS are linear, while the network usage is a tiered billing function. On the other hand, the uptime billing functions of Azure SQL is linear, while the storage and network billing functions are tiered.
7. Usage Estimation

The uptime is easily measurable; it is more difficult to estimate accurately the usage of storage and network, since they depend on the database structure, the workload and the use of encryption. We now propose a methodology for the estimation of storage and network usage due to encryption. For clarity, we define sp, se, sa as the storage usage in the plaintext, encrypted, and adaptively encrypted databases for one billing period. Similarly, np, ne, na represent network usage of the three configurations. We assume that the tenant knows the database structure and the query workload and we assume that each column a stores ra values. By denoting as vp the average storage size of each plaintext value stored in column a, we estimates the storage of the plaintext database.

IV. EXPERIMENTAL RESULTS

We demonstrate the applicability of SecureDBaaS to different cloud DBaaS solutions by implementing and handling encrypted database operations on emulated and real cloud infrastructures. The present version of the SecureDBaaS prototype supports PostgreSQL, MySQL, and SQL Server relational databases. As a first result, we can observe that porting SecureDBaaS to different DBMS required minor changes related to the database connector, and minimal modifications of the codebase. We refer to Appendix C, available in the online supplemental material, for an in-depth description of the prototype implementation. Other tests are oriented to verify the functionality of SecureDBaaS on different cloud database providers. Experiments are carried out in Xeround, Postgres plus Cloud Database, Windows SQL Azure, and also on an IaaS provider, such as Amazon EC2, that requires a manual setup of the database. The first group of cloud providers offers ready-to-use solutions to tenants, but they do not allow a full access to the database system. For example, Xeround provides a standard MySQL interface and proprietary APIs that simplify scalability and availability of the cloud database, but do not allow a direct access to the machine. This prevents the installation of additional software, the use of tools, and any customization. On the positive side, SecureDBaaS using just standard SQL commands can encrypt tenant data on any cloud database service.

Some advanced computation on encrypted data may require the installation of custom libraries on the cloud infrastructure. This is the case of Postgres plus Cloud that provides SSH access to enrich the database with additional functions. The next set of experiments evaluates the performance and the overheads of our prototype. We use the Emulab test bed that provides us a controlled environment with several machines, ensuring repeatability of the experiments for the variety of scenarios to consider in terms of workload models, number of clients, and network latencies. As the workload model for the database, we refer to the TPC-C benchmark. The DBMS server is PostgreSQL 9.1 deployed on a quad-core Xeon having 12 GB of RAM. Clients are connected to the server through a LAN, where we can introduce arbitrary network latencies to emulate WAN connections that are typical of cloud services the experiments evaluate the overhead of encryption, compare the response times of plain versus encrypted database operations, and analyze the impact of network latency. We consider two TPC-C compliant databases with 10 warehouses that contain the same number of tuples: plain tuples consist of 1,046 MB data, while SecureDBaaS tuples have size equal to 2,615 MB because of encryption overhead. Both databases use repeatable read (snapshot) isolation level.

![Fig.2. Encryption times of TPC-C benchmark operations grouped by the transaction class.](image1)

![Fig.3. Plain versus encrypted SELECT and DELETE operations.](image2)

In the first set of experiments, we evaluate the overhead introduced when one SecureDBaaS client executes SQL operations on the encrypted database. Client and database server are connected through a LAN where no network latency is added. To evaluate encryption costs, the client measures the execution time of the 44 SQL commands of the TPC-C benchmark. Encryption times are reported in the histogram of the Fig. 2 that has a logarithmic Y-axis. TPC-C operations are grouped on the basis of the class of transaction:
Scattered, Synchronized and Self Regulating Access to Encrypted Cloud Database

Order Status, Delivery, Stock Level, Payment, and New Order. From this figure, we can appreciate that the encryption time is below 0.1 ms for the majority of operations, and below 1 ms for almost all operations but two. The exceptions are represented by two operations of the Stock Level and Payment transactions where the encryption time is two orders of magnitude higher. This high overhead is caused by the use of the order preserving encryption that is necessary for range queries.

To evaluate the performance overhead of encrypted SQL operations, we focus on the most frequently executed SELECT, INSERT, UPDATE, and DELETE commands of the TPC-C benchmark. In Figs. 3 and 4, we compare the response times of SELECT and DELETE, and UPDATE and INSERT operations, respectively. The Y-axis reports the Box plots of the response times expressed in ms (at a different scale), while the X-axis identifies the SQL operations. In SELECT, DELETE, and UPDATE operations, the response times of SecureDBaaS SQL commands is almost doubled, while the INSERT operation is, as expected, more critical from the computational point of view and it achieves a tripled response time with respect to the plain version. This higher overhead is motivated by the fact that an INSERT command has to encrypt all columns of a tuple, while an UPDATE operation encrypts just one or few values.

The second set of the experiments is oriented to evaluate the impact of network latency and concurrency on the use of a cloud database from geographically distant clients. To this purpose, we emulate network latencies through the traffic shaping utilities available in the Linux kernel by introducing synthetic delays from 20 to 150 ms in the client-server connection. These values are representative of round-trip times in continental (in the range of 40-60 ms) and intercontinental (in the range of 80-150 ms) connections, that are expected when a cloud-based solution is deployed. Table 1 report the response times of the most frequent SQL operations in the plain and encrypted cases for 20, 40, and 80 ms latencies. The last column of this table also reports the absolute and percentage overhead introduced by SecureDBaaS. These experimental results demonstrate that the response times of the SQL operations issued to a remote database are dominated by network latencies even in well connected regions. Each response time is two orders of magnitude higher than the corresponding time of a plain SQL operation in a LAN environment. Thanks to this effect, the overhead of SecureDBaaS for the most common SELECT operation falls from 57 percent to 1.31 percent and to 0.26 percent in correspondence of network latencies equal to 20 ms and 80 ms, respectively.

### TABLE 1: Response Times and Overheads of SQL Operations for Different Network Latencies

<table>
<thead>
<tr>
<th>Network delay</th>
<th>SQL command</th>
<th>Plaintext response time</th>
<th>Encrypted response time</th>
<th>Overhead (absolute and percentage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 ms</td>
<td>SELECT</td>
<td>20.67 ms</td>
<td>20.94 ms</td>
<td>0.27 ms 1.31%</td>
</tr>
<tr>
<td></td>
<td>DELETE</td>
<td>20.66 ms</td>
<td>20.97 ms</td>
<td>0.31 ms 1.50%</td>
</tr>
<tr>
<td></td>
<td>UPDATE</td>
<td>20.67 ms</td>
<td>21.12 ms</td>
<td>0.45 ms 2.18%</td>
</tr>
<tr>
<td></td>
<td>INSERT</td>
<td>20.85 ms</td>
<td>21.61 ms</td>
<td>0.76 ms 3.65%</td>
</tr>
<tr>
<td>40 ms</td>
<td>SELECT</td>
<td>40.64 ms</td>
<td>40.90 ms</td>
<td>0.26 ms 0.64%</td>
</tr>
<tr>
<td></td>
<td>DELETE</td>
<td>40.65 ms</td>
<td>40.92 ms</td>
<td>0.27 ms 0.66%</td>
</tr>
<tr>
<td></td>
<td>UPDATE</td>
<td>40.62 ms</td>
<td>41.08 ms</td>
<td>0.46 ms 1.13%</td>
</tr>
<tr>
<td></td>
<td>INSERT</td>
<td>40.82 ms</td>
<td>41.56 ms</td>
<td>0.74 ms 1.81%</td>
</tr>
<tr>
<td>80 ms</td>
<td>SELECT</td>
<td>80.76 ms</td>
<td>80.97 ms</td>
<td>0.21 ms 0.26%</td>
</tr>
<tr>
<td></td>
<td>DELETE</td>
<td>80.67 ms</td>
<td>81.01 ms</td>
<td>0.34 ms 0.42%</td>
</tr>
<tr>
<td></td>
<td>UPDATE</td>
<td>80.65 ms</td>
<td>81.09 ms</td>
<td>0.44 ms 0.55%</td>
</tr>
<tr>
<td></td>
<td>INSERT</td>
<td>80.86 ms</td>
<td>81.63 ms</td>
<td>0.77 ms 0.95%</td>
</tr>
</tbody>
</table>

The last set of experiments assess the performance of SecureDBaaS in realistic cloud database scenarios, as well as its ability to support multiple, distributed, and independent

---

**Fig.4. Plain versus encrypted UPDATE and INSERT operations**

The second set of the experiments is oriented to evaluate the impact of network latency and concurrency on the use of a cloud database from geographically distant clients. To this purpose, we emulate network latencies through the traffic shaping utilities available in the Linux kernel by introducing synthetic delays from 20 to 150 ms in the client-server connection. These values are representative of round-trip times in continental (in the range of 40-60 ms) and intercontinental (in the range of 80-150 ms) connections, that are expected when a cloud-based solution is deployed. Table 1 report the response times of the most frequent SQL operations in the plain and encrypted cases for 20, 40, and 80 ms latencies. The last column of this table also reports the absolute and percentage overhead introduced by SecureDBaaS. These experimental results demonstrate that the response times of the SQL operations issued to a remote database are dominated by network latencies even in well connected regions. Each response time is two orders of magnitude higher than the corresponding time of a plain SQL operation in a LAN environment. Thanks to this effect, the overhead of SecureDBaaS for the most common SELECT operation falls from 57 percent to 1.31 percent and to 0.26 percent in correspondence of network latencies equal to 20 ms and 80 ms, respectively.

**TABLE 1: Response Times and Overheads of SQL Operations for Different Network Latencies**

<table>
<thead>
<tr>
<th>Network delay</th>
<th>SQL command</th>
<th>Plaintext response time</th>
<th>Encrypted response time</th>
<th>Overhead (absolute and percentage)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 ms</td>
<td>SELECT</td>
<td>20.67 ms</td>
<td>20.94 ms</td>
<td>0.27 ms 1.31%</td>
</tr>
<tr>
<td></td>
<td>DELETE</td>
<td>20.66 ms</td>
<td>20.97 ms</td>
<td>0.31 ms 1.50%</td>
</tr>
<tr>
<td></td>
<td>UPDATE</td>
<td>20.67 ms</td>
<td>21.12 ms</td>
<td>0.45 ms 2.18%</td>
</tr>
<tr>
<td></td>
<td>INSERT</td>
<td>20.85 ms</td>
<td>21.61 ms</td>
<td>0.76 ms 3.65%</td>
</tr>
<tr>
<td>40 ms</td>
<td>SELECT</td>
<td>40.64 ms</td>
<td>40.90 ms</td>
<td>0.26 ms 0.64%</td>
</tr>
<tr>
<td></td>
<td>DELETE</td>
<td>40.65 ms</td>
<td>40.92 ms</td>
<td>0.27 ms 0.66%</td>
</tr>
<tr>
<td></td>
<td>UPDATE</td>
<td>40.62 ms</td>
<td>41.08 ms</td>
<td>0.46 ms 1.13%</td>
</tr>
<tr>
<td></td>
<td>INSERT</td>
<td>40.82 ms</td>
<td>41.56 ms</td>
<td>0.74 ms 1.81%</td>
</tr>
<tr>
<td>80 ms</td>
<td>SELECT</td>
<td>80.76 ms</td>
<td>80.97 ms</td>
<td>0.21 ms 0.26%</td>
</tr>
<tr>
<td></td>
<td>DELETE</td>
<td>80.67 ms</td>
<td>81.01 ms</td>
<td>0.34 ms 0.42%</td>
</tr>
<tr>
<td></td>
<td>UPDATE</td>
<td>80.65 ms</td>
<td>81.09 ms</td>
<td>0.44 ms 0.55%</td>
</tr>
<tr>
<td></td>
<td>INSERT</td>
<td>80.86 ms</td>
<td>81.63 ms</td>
<td>0.77 ms 0.95%</td>
</tr>
</tbody>
</table>

The last set of experiments assess the performance of SecureDBaaS in realistic cloud database scenarios, as well as its ability to support multiple, distributed, and independent

---

**Fig.5. TPC-C performance (20 concurrent clients).**

The last set of experiments assess the performance of SecureDBaaS in realistic cloud database scenarios, as well as its ability to support multiple, distributed, and independent
clients. The test bed is similar to that described previously, but now the runs are repeated by varying the number of concurrent clients (from 1 to 40) and the network latencies (from plain LAN to delays reaching 150 ms). All clients execute concurrently the benchmark for 300 seconds. The results in terms of throughput refer to three types of database operations:

- Original TPC-C: the standard TPC-C benchmark;
- Plain-SecureDBaaS: SecureDBaaS that use plain encryption, that is, all SecureDBaaS functions and data structures with no encryption; it allows us to evaluate the overhead of SecureDBaaS without the cost of cryptographic operations;
- SecureDBaaS: SecureDBaaS referring to the highest confidentiality level.

Fig. 5 shows the system throughput referring to 20 clients issuing requests to SecureDBaaS as a function of the network latency. The Y-axis reports the number of committed transactions per minute during the entire experiment. This figure shows two important results:

- If we exclude the cryptographic costs, SecureDBaaS does not introduce significant overheads. This can be appreciated by verifying that the throughput of plain SecureDBaaS and original TPC-C overlies for any realistic Internet delay (>20 ms);
- As expected, the number of transactions per minute executed by SecureDBaaS is lower than those referring to original TPC-C and plain-SecureDBaaS, but the difference rapidly decreases as the network latency increases to the extent that is almost nullified in any network scenario that can be realistically referred to a cloud database context.

Figs 6 and 7 shows the throughput for increasing numbers of concurrent clients in contexts characterized by 40 ms and 80 ms network latencies, respectively.

**V. CONCLUSION**

We address the data privacy concerns by proposing a novel cloud database model that uses adaptive encryption techniques with no intermediate servers. This scheme provides tenants with the best level of privacy for any database workload that is to change in a medium-term period. We investigate the feasibility and performance of the proposed architecture through a large set of experiments based on a software prototype subject. Our results analysis proved that the cloud networks semantic that are typical of cloud database environments hide most overheads related to static and adaptive encryption. We address the data confidentiality concerns by proposing a novel cloud database architecture that uses adaptive encryption techniques with no intermediate servers. This scheme provides tenants with the best level of confidentiality for any database workload that is likely to change in a medium-term period. We investigate the feasibility and performance of the proposed architecture through a large set of experiments based on a software prototype subject to the TPC-C standard benchmark. Our results demonstrate that the network latencies that are typical
Scattered, Synchronized and Self Regulating Access to Encrypted Cloud Database

of cloud database environments hide most overheads related to static and adaptive encryption. Moreover, we propose a model and a methodology that allow a tenant to estimate the costs of plain and encrypted cloud database services even in the case of workload and cloud price variations in a medium-term horizon. By applying the model to actual cloud provider prices, we can determine the encryption and adaptive encryption costs for data confidentiality. Future research could evaluate the proposed or alternative architectures for multi-user key distribution schemes and under different threat model hypotheses.

VI. REFERENCES