Assimilation of Encrypted Database Services By Distributed, Concurrent and Maverick Access

Gotivada Soujanya[1]
[1] Andhra University, Visakhapatnam, Andhra Pradesh, India

Abstract- *Cloud computing is a method for delivering information technology services in which resources are retrieved from the internet through web-based tools and applications, as opposed to a direct connection to a server. The cloud database as a service is a novel paradigm that can support several internet based applications, but its adoption requires the solution of the information confidentiality issues. 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, Secured BaaS, 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 com- potation 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 Drop box) 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 centre. 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 experts in cloud computing.

II. RELATED WORK

In the field of security for remote database services, SecureDBaaS provides various features that differentiate it from the previous.

- SecureDBaaS provides data confidentiality by allowing a cloud database server to execute concurrent SQL operations over encrypted data.
- It gives the same availability, scalability and elasticity of the original cloud DBaaS because it does not require any intermediate server. Response times are affected by cryptographic overheads that for most SQL operations are masked by network latencies.
- Many clients, mainly geographically scattered, can access concurrently and independently a cloud database service.
- Tenant data and metadata stored by the cloud database are always encrypted and they does not require a trusted broker or a trusted proxy.
- It is applicable to different DBMS implementations because all adopted solutions are database agnostic and it is compatible with the most popular relational database servers.
- Secure storage systems and cryptographic file systems represent the earliest works in this field. Details of several papers and products are not explained because they do not support computations on encrypted data.

[1] "A View of Cloud Computing" M. Armbrust

This has developed with innovative ideas for new Internet services no longer require the large capital outlays in hardware to deploy their service or the human expense to operate it. Cloud Computing will grow, so developers should take it into account. Moreover:

a) Applications Software needs to both scale down rapidly as well as scale up, which is a new requirement. Such software also needs a pay-for-use licensing model to match needs of Cloud Computing.
b) Infrastructure Software needs to be aware that it is no longer running on bare metal but on VMs. Moreover, billing needs to build in from the start.
c) Hardware Systems should be designed at the scale of a container (at least a dozen racks), which will be is the minimum purchase size.

[2] "SPORC: Group Collaboration Using Untrusted Cloud Resources" A.J. Feldman, W.P. Zeller, M.J. Freedman, and E.W. Felten

He described Cloud-based services are an attractive deployment model for user-facing applications like word processing and calendaring. In SPORC, a server observes only encrypted data and cannot deviate from correct execution without being detected. SPORC allows concurrent, low-latency editing of shared state, permits disconnected operation, and supports dynamic access control even in the presence of concurrency acknowledgments.

[3] "Secure Untrusted Data Repository (SUNDR)" J. Li, M. Krohn, D. Mazie` res, and D. Shasha

He proposed SUNDR is a network file system designed to store data securely on untrusted servers. SUNDR's protocol achieves a property called fork consistency, which guarantees that clients can detect any integrity or consistency failures as long as they see each other's file modifications. Measurements of our implementation show performance that is usually close to and sometimes better than the popular NFS file system.

[4] "Depot: Cloud Storage with Minimal Trust" P. Mahajan, S. Setty, S. Lee, A. Clement, L. Alvisi, M. Dahlin, and M. Walfish

He described the design, implementation, and evaluation of a Depot, a cloud storage system that minimises trust assumptions. Depot began with an attempt to explore a radical point in the design space for cloud storage: Trust No One.

[5] "Providing Database as a Service" H. Hacigu¨ mu¨ s, B. Iyer, and S. Mehrotra

He proposed a new paradigm for data management in which a third party service provider hosts "database as a service" providing its customers seamless mechanisms to create, store, and access their databases at the host site. The authors introduced NetDB2, an internet-based database service built on top of DB2 that provides users with tools for application development, creating and loading tables, and performing queries and transactions.

[6] "Fully Homomorphic Encryption Using Ideal Lattices" C. Gentry

He proposed a fully homomorphism encryption scheme – i.e., a scheme that allows one to evaluate circuits over encrypted data without being able to decrypt. The circuit privacy of E2 immediately implies the (levelled) circuit privacy of our (levelled) fully homomorphism encryption scheme.

III. METHODOLOGY

This paper propose a novel architecture that integrates cloud database services with data confidentiality and the possibility of executing concurrent operations on encrypted data. This is the novel solution supporting geographically distributed clients to connect directly to an encrypted cloud database, and to execute concurrent and independent operations including those modifying the database structure.The proposed architecture has the further advantage of eliminating intermediate proxies that limit the elasticity, availability, andscalability properties that are intrinsic in cloud-based solutions. Secure DBaaS provides several original features that differentiate it from previous work in the field of security for remote database services.

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 database workload may change during the evaluation period. This model is instanced 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.

IV. ARCHITECTURE DESIGN

SecureDBaaS was implemented to allow multiple and independent clients to connect directly to the trusted cloud DBaaS without any intermediate server. Fig1 describes the architecture. We conclude that a tenant organisation acquires a cloud database service for an untrusted DBaaS provider. The tenant then deploys one or more machines each of them. This client allows a user to connect to the cloud DBaaS to administer it, to read and write data, and even to create and modify the database tables after creation.

We assume the same security model that is commonly adopted by the literature in this field where tenant users are trusted, the network is untrusted, and the cloud provider is honest-but-curious, that is, cloud service operations are executed correctly, but tenant information confidentiality is at risk. For these reasons, tenant data, data structures, and metadata must be encrypted before exiting from the client. A thorough presentation of the security model adopted in this paper is in Appendix A, available in the online supplemental material.

The information managed by SecureDBaaS includes plaintext data, encrypted data, metadata, and encrypted metadata. Plaintext data consist of information that a tenant wants to store and process remotely in the cloud DBaaS. To prevent an untrusted cloud provider from violating confidentiality of tenant data stored in plain form, SecureDBaaS adopts multiple cryptographic techniques to transform plaintext data into encrypted tenant data and encrypted tenant data structures because even the names of the tables and of their columns must be encrypted. SecureDBaaS clients produce also a set of metadata consisting of information required to encrypt and decrypt data as well as other administration information. Even metadata are encrypted and stored in the cloud DBaaS.

SecureDBaaS moves away from existing architectures that store just tenant data in the cloud database, and save metadata in the client machine or split metadata between the cloud database and a trusted proxy. When considering scenarios where multiple clients can access the same database concurrently, these previous solutions are quite inefficient. For example, saving metadata on the clients would require onerous mechanisms for metadata synchronization, and the practical impossibility of allowing multiple clients to access cloud database services independently. Solutions based on a trusted proxy are more feasible, but they introduce a system bottleneck that reduces availability, elasticity, and scalability of cloud database services.

SecureDBaaS proposes a different approach where all data and metadata are stored in the cloud database. SecureDBaaS clients can retrieve the necessary metadata from the untrusted database through SQL statements, so that multiple instances of the SecureDBaaS client can access to the untrusted cloud database independently with the guarantee of the same availability and scalability properties of typical cloud DBaaS. Encryption strategies for tenant data and innovative solutions for metadata management and storage are described in the following two sections.

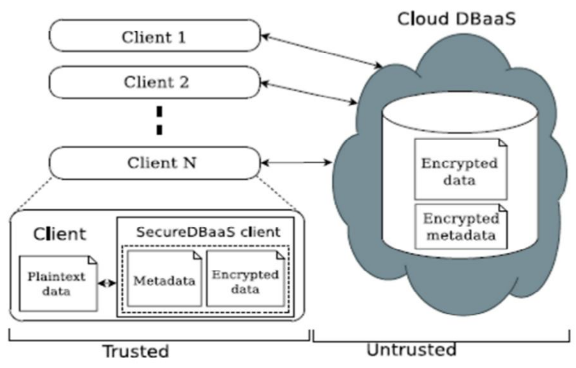


Figure 1.

1. DATA MANAGEMENT

It assumes that tenant data are saved in a relational database. It has to preserve the confidentiality of the stored data and even of the database structure because table and column names may yield information about saved data. This paper distinguishes the strategies for encrypting the database structures and the tenant data. Encrypted tenant data are stored through secure tables into the cloud database. To allow transparent execution of SQL statements, each Plaintext table is transformed into a secure table because the cloud database is untrusted. Secure DBaaS offers three field confidentiality attributes:

- Column (COL) is the default confidentiality level that should be used when SQL statements operate on one column; the values of this column are encrypted through a randomly generated encryption key that is not used by any other column.
- Multicolumn (MCOL) should be used for columns referenced by join operators, foreign keys, and other operations involving two columns; the two columns are encrypted through the same key.
- Database (DBC) is recommended when operations involve multiple columns; in this instance, it is convenient to use the special encryption key that is generated and implicitly shared among all the columns of the database characterized by the same secure type.The choice of the field confidentiality levels makes it possible to execute SQL statements over encrypted data while allowing a tenant to minimize key sharing.

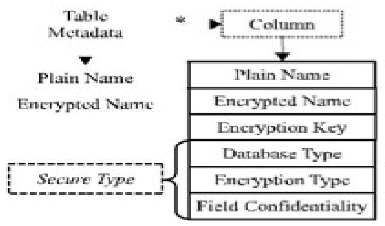
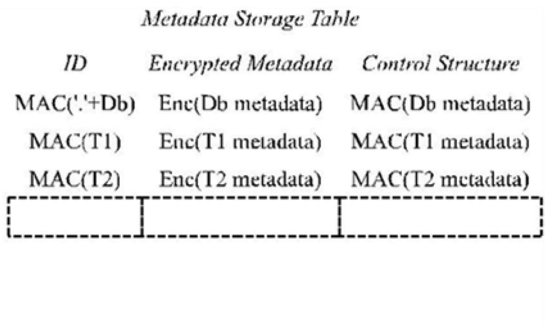


Figure 2. Structure of table metadata

2. METADATA MANAGEMENT

Metadata generated by Secure DBaaS contain all the information that is necessary to manage SQL statements over the encrypted database in a way transparent to the user. Metadata management strategies represent an original idea because Secure DBaaS is the first architecture storing all metadata in the untrusted cloud database together with the encrypted tenant data. Secure DBaaS uses two types of metadata.1) Database metadata are related to the whole database. There is only one instance of this metadata type for each database.2) Table metadata are associated with one secure table. Each table metadata contains all information that is necessary to encrypt and decrypt data of the associated secure table.



Metadata Storage Table

ID	*Encrypted Metadata*	*Control Structure*
MAC('.'+Db)	Enc(Db metadata)	MAC(Db metadata)
MAC(T1)	Enc(T1 metadata)	MAC(T1 metadata)
MAC(T2)	Enc(T2 metadata)	MAC(T2 metadata)

Figure 3.

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:

Cost = f (Time, Pricing, and Usage) where:

Time: Identifies the time interval T for which the tenant requires the service.

Pricing: Refers to the prices of the cloud provider for subscription and resource usage; they typically tend to diminish during T.

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 payper-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 xb its usage at the b-th billing period and px 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 [x1, . . . ,xZ−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 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.

8. DATAFLOW DIAGRAM

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business Function. The technique starts with an overall picture of the business and continues by analysing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way. As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analysing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

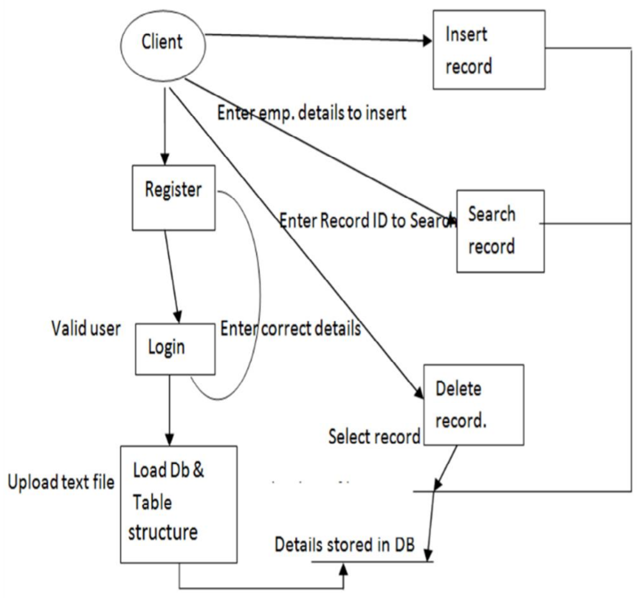


Figure 4.

V. EXPERIMENTAL RESULTS

We demonstrate the applicability of Secure DBaaS 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 Postgre SQL, My Sql, 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 My Sql 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.

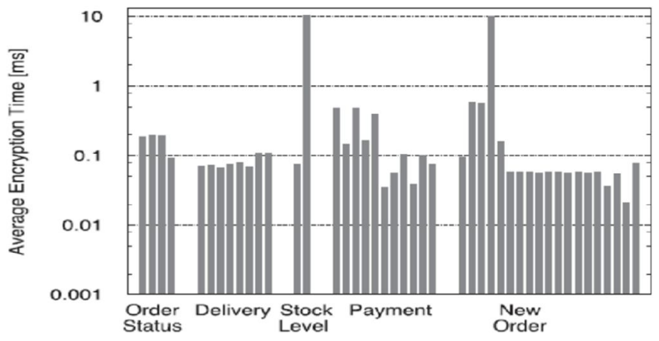


Figure 5. Encryption Times of TPC-C Benchmark Operations Grouped by the Transaction Class.

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 PostgreSQL9.1 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. 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.

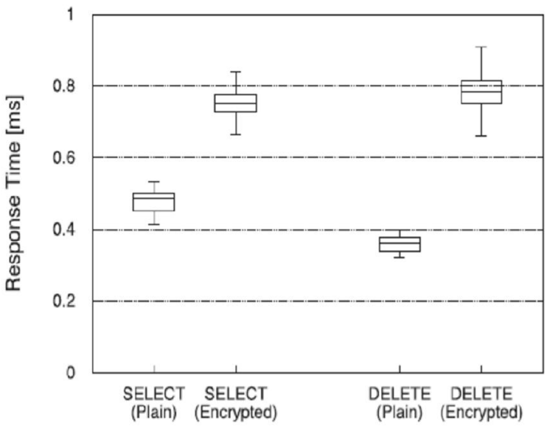


Figure 6. Plain versus Encrypted SELECT and DELETE operations.

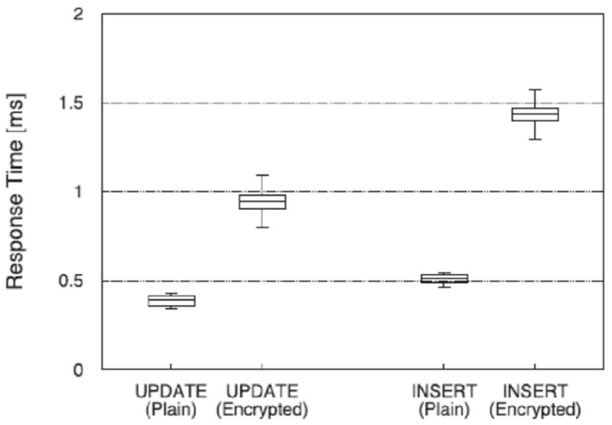


Figure 7. Plain versus encrypted UPDATE and INSERT operations

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: 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. 4 and 5, 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.

Table 1. Response Times and Overheads of SQL Operations for Different Network Latencies

Network delay	SQL com-mand	Plaintext response time	Encrypted response time	Overhead (absolute and percentage)
LAN	SELECT	0.478 ms	0.753 ms	0.275 ms 57%
	DELETE	0.369 ms	0.783 ms	0.414 ms 112%
	UPDATE	0.397 ms	0.951 ms	0.554 ms 140%
	INSERT	0.517 ms	1.442 ms	0.925 ms 179%
20 ms	SELECT	20.67 ms	20.94 ms	0.27 ms 1.31%
	DELETE	20.66 ms	20.97 ms	0.31 ms 1.50%
	UPDATE	20.67 ms	21.12 ms	0.45 ms 2.18%
	INSERT	20.85 ms	21.61 ms	0.76 ms 3.65%
40 ms	SELECT	40.64 ms	40.90 ms	0.26 ms 0.64%
	DELETE	40.65 ms	40.92 ms	0.27 ms 0.66%
	UPDATE	40.62 ms	41.08 ms	0.46 ms 1.13%
	INSERT	40.82 ms	41.56 ms	0.74 ms 1.81%
80 ms	SELECT	80.76 ms	80.97 ms	0.21 ms 0.26%
	DELETE	80.67 ms	81.01 ms	0.34 ms 0.42%
	UPDATE	80.65 ms	81.09 ms	0.44 ms 0.55%
	INSERT	80.86 ms	81.63 ms	0.77 ms 0.95%

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. 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.

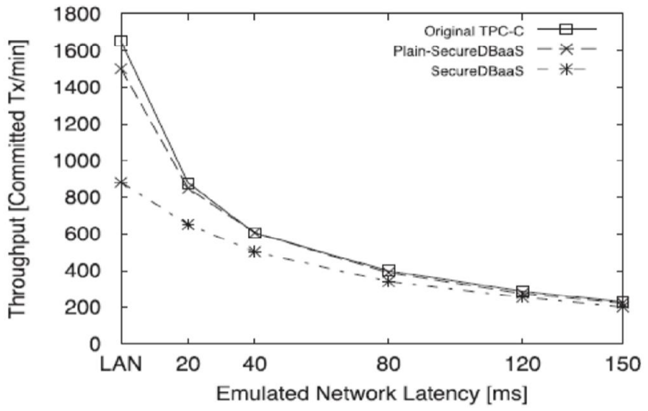


Figure 8. TPC-C performance (20 concurrent clients).

Fig.6 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.

Fig.7 and Fig. 8 shows the throughput for increasing numbers of concurrent clients in contexts characterized by40 ms and 80ms network latencies, respectively.

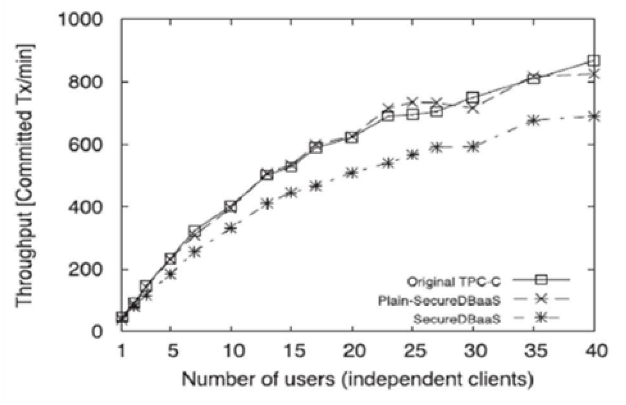


Figure 9. TPC-C performance (20 concurrent clients)

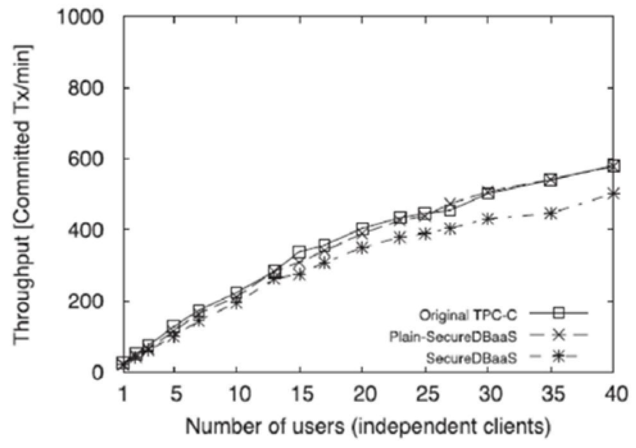


Figure 10. TPC-C performance (latency equal to 80ms)

These measures are optimistic representations of continental and intercontinental delays. The Y-axis represents the number of committed TPC-C transactions per minute executed by the clients. The trends of the SecureDBaaS lines are close to those of the original TPC-C database, thus demonstrating that SecureDBaaS encrypted database does not affect scalability with respect to the plain database. Even more important, the network latencies tend to mask cryptographic overheads for any number of clients. For example, the overheads of SecureDBaaS with 40 concurrent clients decreases from 20 percent in a 40-ms scenario to 13 percent in a realistic scenario, where the client-server latency is equal to 80 ms. This result is important because it confirms that SecureDBaaS is a valid and practical solution for guaranteeing data confidentiality in real cloud database services.

VI. 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 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 deter- mine 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.

REFERENCES

[1] M. Armbrust et al., “A View of Cloud Computing,” Comm. of the ACM, vol. 53, no. 4, pp. 50-58, 2010.

[2] A.J. Feldman, W.P. Zeller, M.J. Freedman, and E.W. Felten, “SPORC: Group Collaboration Using Untrusted Cloud Resources,” Proc. Ninth USENIX Conf. Operating Systems Design and Implementation, Oct. 2010.

[3] J. Li, M. Krohn, D. Mazie` res, and D. Shasha, “Secure Untrusted Data Repository (SUNDR),” Proc. Sixth USENIX Conf. Operating Systems Design and Implementation, Oct. 2004.

[4] P. Mahajan, S. Setty, S. Lee, A. Clement, L. Alvisi, M. Dahlin, and M. Walfish, “Depot: Cloud Storage with Minimal Trust”, ACMTrans. Computer Systems, vol. 29,

no. 4, article12, 2011.

[5] H. Hacigu¨ mu¨ s¸, B. Iyer, and S. Mehrotra, “Providing Database as a Service”, Proc. 18th IEEE Int’l Conf. Data Eng.Feb.2002.

[6] C. Gentry, “Fully Homomorphic Encryption Using Ideal Lattices”, Proc. 41st Ann. ACM Symp. Theory of Computing May 2009.

[7] R.A. Popa, C.M.S. Redfield, N. Zeldovich, and H. Balakrishnan, “CryptDB: Protecting Confidentiality with Encrypted Query Processing,” Proc. 23rd ACM Symp. Operating Systems Principles, Oct. 2011.

[8] J. Li and E. Omiecinski, “Efficiency and Security TradeOff in Supporting Range Queries on Encrypted Databases,” Proc. 19thAnn. IFIP WG 11.3 Working Conf. Data and Applications Security, Aug. 2005.

[9] V.Ganapathy, D. Thomas, T. Feder, H. Garcia-Molina, and R.Motwani, “Distributing Data for Secure Database Services,” Proc. Fourth ACM Int’l Workshop Privacy and Anonymity in the Information Soc., Mar. 2011.