Performance and Cost Evaluation of an Adaptive Encryption Architecture for Cloud Databases

Luca Ferretti, Fabio Pierazzi, Michele Colajanni, and Mirco Marchetti

Abstract—The cloud database as a service is a novel paradigm that can support several Internet-based applications, but its adoption requires the solution of information confidentiality problems. We propose 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. We demonstrate the feasibility and performance of the proposed solution through a software prototype. Moreover, we propose an original cost model that is oriented to the evaluation of cloud database services in plain and encrypted instances and that takes into account the variability of cloud prices and tenant workloads during a medium-term period.

Index Terms—Cloud database, confidentiality, encryption, adaptivity, cost model

1 Introduction

The cloud computing paradigm is successfully converging as the fifth utility [1], but this positive trend is partially limited by concerns about information confidentiality [2] and unclear costs over a medium-long term [3], [4].

We are interested in the database as a service paradigm (DBaaS) [5] 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 [6], [7] are inapplicable to the database paradigm. Other encryption schemes that allow the execution of SQL operations over encrypted data either have performance limits [8] or require the choice of which encryption scheme must be adopted for each database column and SQL operation [9]. 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 database 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 in [10]. 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 outer layers are dynamically adapted at runtime when new SQL operations are added to the workload.

Although this adaptive encryption architecture is attractive because it does not require to define at design time which database operations are allowed on each column, it poses novel issues in terms of applicability to a cloud context, and doubts about storage and network costs. We investigate each of these issues and we reach three original conclusions in terms of prototype implementation, performance evaluation, and cost evaluation.

We initially design the first proxy-free architecture for adaptive encryption of cloud databases that does not limit the availability, elasticity and scalability of a plain cloud database because multiple clients can issue concurrent operations without passing through some centralized component as in alternative architectures [10]. Then, we evaluate the performance of encrypted database services by assuming the standard TPC-C benchmark as the workload and by considering different network latencies. Thanks to this testbed, we show that most performance overheads of adaptively encrypted cloud databases are masked by network latencies that are typical of a geographically distributed cloud scenario.

Finally, we propose the first analytical cost estimation model for evaluating cloud database costs in plaintext and encrypted configurations from a tenant's point of view over a medium-term period. This model also considers the variability of cloud prices and of the database workload during the evaluation period, and allows a tenant to observe how adaptive encryption influences the costs related to storage and network usage of a database service. By applying the model to several cloud provider offers and related prices, the tenant can choose the best compromise between the data confidentiality level and consequent costs in his period of interest.

This paper is structured as following. Section 2 examines related solutions for data confidentiality and cost estimation in cloud database services, and compares them against our proposal. Section 3 describes the proposed adaptive encryption architecture for cloud database

• *The authors are with the Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, Modena, Italy. E-mail: {luca.ferretti, fabio.pierazzi, michele.colajanni, mirco.marchetti}@unimore.it.*

Manuscript received 15 Sept. 2013; revised 7 Mar. 2014; accepted 18 Mar. 2014. Date of publication 1 Apr. 2014; date of current version 30 July 2014.
Recommended for acceptance by I. Bojanova, R.C.H. Hua, O. Rana, and M. Parashar.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TCC.2014.2314644

services. Section 4 proposes the analytical cost model for the estimation of database service costs in a medium term where it is likely that workloads and cloud prices change. Section 5 presents experimental evaluations for different network scenarios, workload models and number of clients. Section 6 reports the results of the cost model applied to real cloud database providers over a three years horizon that is a typical view for tenant's investments. Section 7 outlines main conclusions and possible directions for further research.

2 Related Work

Improving the confidentiality of information stored in cloud databases represents an important contribution to the adoption of the cloud as the fifth utility because it addresses most user concerns. Our proposal is characterized by two main contributions to the state of the art: architecture and cost model.

Although data encryption seems the most intuitive solution for confidentiality, its application to cloud database services is not trivial, because the cloud database must be able to execute SQL operations directly over encrypted data without accessing any decryption key. Naïve solutions encrypt the whole database through some standard encryption algorithms that do not allow to execute any SQL operation directly on the cloud. As a consequence, the tenant has two alternatives: download the entire database, decrypt it, execute the query and, if the operation modifies the database, encrypt and upload the new data; decrypt temporarily the cloud database, execute the query, and re-encrypt it. The former solution is affected by huge communication and computation overheads, and consequent costs that would make cloud database services quite inconvenient; the latter solution does not guarantee data confidentiality because the cloud provider obtains decryption keys.

The right alternative is to execute SQL operations directly on the cloud database, without giving decryption keys to the provider. An initial solution presented in [5] is based on data aggregation techniques [8], that associate plaintext metadata to sets of encrypted data. However, plaintext metadata may leak sensitive information and data aggregation introduces unnecessary network overheads.

The use of *fully homomorphic encryption* [11] 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 [12], [13], [14]. For example, an encryption algorithm may support the order comparison command [12], but not a search operator [14]. 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 in [10] by proposing an adaptive encryption architecture that is founded on an intermediate and trusted proxy. This tenant's component, which mediates all the interactions between the clients and a possibly untrusted DBMS server, is fine for a locally distributed architecture but it cannot be applied to a cloud context. Indeed, any centralized component at the tenant side reduces the scalability and availability that are among the most important features of cloud services. A solution to this problem was presented in [9]: the proposed architecture allows multiple clients to issue concurrent SQL operations to an encrypted database without any intermediate trusted server, but it assumes that the set of SQL operations does not change after the database design. A first idea to integrate adaptive encryption schemes with a proxy-free architecture was proposed by the same authors in [15]. This paper develops the initial design through a prototype implementation, novel experimental results and an original cost model.

Indeed, besides data confidentiality, unclear costs are a main concern for cloud tenants. To this purpose, we propose an analytical cost model and a usage estimation methodology that allow a tenant to estimate the costs deriving from cloud database services 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. This model is another original contribution of this paper because previous research focuses on the costs of cloud computing from a provider's perspective [16], [17]. For example, the authors in [16] outline the problems related to the cost estimation of a cloud data center, such as servers, power consumption, and infrastructures, but they do not propose an analytical cost estimation model. CloudSim [18] can help a provider to estimate performance and resource consumptions of one or multiple cloud data center alternatives.

This paper has a focus on database services and takes an opposite direction by evaluating the cloud service costs from a tenant's point of view. This approach is quite original because related papers evaluate the pros and cons of porting scientific applications to a cloud platform, such as [4] focusing on specific astronomy software and a specific cloud provider (Amazon), and [3] presenting a composable cost estimation model for some classes of scientific applications. Besides the focus on a different context (scientific versus database applications), the proposed model can be applied to any cloud database service provider, and it takes into account that over a medium-term period the database workload and the cloud prices may vary.

3 Architecture Design

The proposed system supports adaptive encryption for public cloud database services, where distributed and concurrent clients can issue direct SQL operations. By avoiding an architecture based on intermediate servers [10] 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* represent the tenant information;
- *encrypted data* are the encrypted version of the plain data, and are stored in the cloud database;

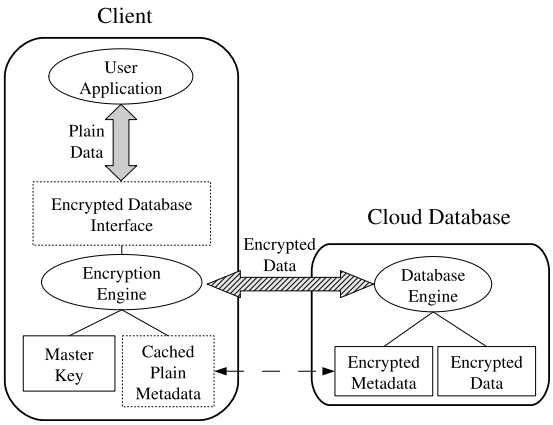

Fig. 1. Encrypted cloud database architecture.

- *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.

All data and metadata stored in the cloud database are encrypted. Any application running on a legitimate client can transparently issue SQL operations (e.g., SELECT, INSERT, UPDATE and DELETE) to the encrypted cloud database through the encrypted database interface. Data transferred between the user application and the encryption engine are not encrypted, whereas information is always encrypted before sending it to the cloud database. When an application issues a new SQL operation, the encrypted database interface contacts the encryption engine that retrieves the encrypted metadata and decrypts them with the master key. To improve performance, the plain metadata are cached locally by the client. After obtaining the metadata, the encryption engine is able to issue encrypted SQL statements to the cloud database, and then to decrypt the results. The results are returned to the user application through the encrypted database interface.

As in related literature, the proposed architecture guarantees data confidentiality in a security model in which: the network is untrusted; tenant users are trusted, that is, they do not reveal information about plain data, plain metadata, and the master key; the cloud provider administrators are defined semi-honest or honest-but-curious [19], that is, they do not modify tenant's data and results of SQL operations, but they may access tenant's information stored in the cloud database. The remaining part of this section describes the adaptive encryption schemes (Section 3.1), the encrypted metadata stored in the cloud database (Section 3.2), and the main operations for the management of the encrypted cloud database (Section 3.3).

3.1 Adaptive Encryption Schemes

We consider SQL-aware encryption algorithms that guarantee data confidentiality and allow the cloud database engine to execute SQL operations over encrypted data. As each algorithm supports a specific subset of SQL operators, we refer to the following encryption schemes.

- *Random* (Rand): it is the most secure encryption because it does not reveal any information about the original plain value (IND-CPA) [20], [21]. It does not support any SQL operator, and it is used only for data retrieval.
- *Deterministic* (Det): it deterministically encrypts data, so that equality of plaintext data is preserved. It supports the equality operator.
- *Order Preserving Encryption* (Ope) [12]: it preserves in the encrypted values the numerical order of the original unencrypted data. It supports the comparison SQL operators (i.e., $=, <, \leq, >, \geq$).
- *Homomorphic Sum* (Sum) [13]: it is homomorphic with respect to the sum operation, so that the multiplication of encrypted integers is equal to the sum of plaintext integers. It supports the sum operator between integer values.
- *Search*: it supports equality check on full strings (i.e., the *LIKE* operator).
- *Plain*: it does not encrypt data, but it is useful to support all SQL operators on non confidential data.

If each column of the database was encrypted with only one algorithm, then the database administrator would have to decide at design time which operations must be supported on each database column. However, this solution is impractical for scenarios in which the database workload changes over time. As an example, if we consider a database supporting a web application for which features or security updates are released, data encryption prevents the application of any update that introduces new SQL operations that were not considered at database design time. Similarly, encryption prevents the execution of data analytics on the encrypted database and of user-defined queries that do not belong to a fixed workload (e.g., because the database is queried directly by tenant employees). This issue can be addressed through *adaptive* encryption schemes that support at runtime SQL operations while preserving the highest possible data confidentiality level on the encrypted data. To this purpose, the encryption algorithms are organized into structures called *onions*, where each onion is composed by an ordered set of encryption algorithms, called *(encryption) layers* [10]. Outer layers guarantee higher level of data confidentiality and support fewer operations on encrypted data. Hence, each onion supports a specific set of operations. We design the following onions:

- *Onion-Eq*: it supports the equality operator, and integrates Plain, Det and Rand layers.
- *Onion-Ord*: it supports the comparison operators (i.e., $=, <, \leq, >, \geq$), and integrates Plain, Ope and Rand layers.
- *Onion-Sum*: it supports the sum operator, and integrates Plain, Sum and Rand layers.
- *Onion-Search*: it supports the string equality operator (*LIKE*), and integrates the Plain, Search and Rand layers.
- *Onion-Single-Layer*: this is a special type of onion that supports only one encryption layer.

Each plaintext column is converted into one or more encrypted columns, each one corresponding to an onion. Each plaintext value is encrypted through all the layers of

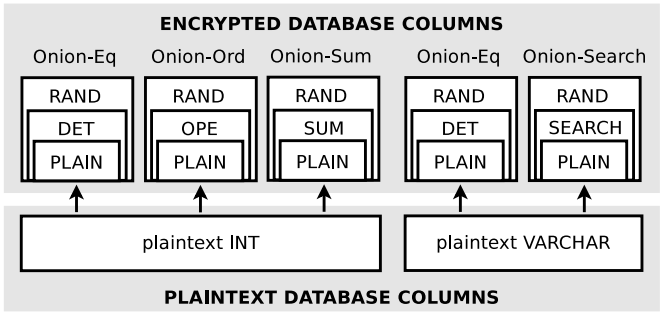

Fig. 2. Example of onion structures.

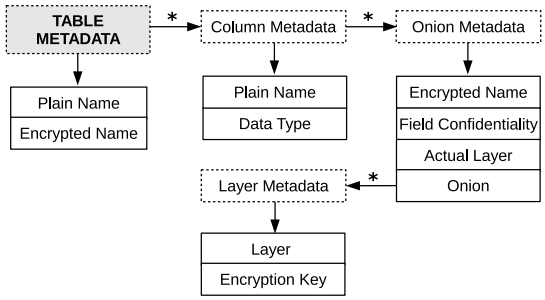

Fig. 3. Metadata structure.

its onions. For example, the plaintext values associated with Onion-Eq are encrypted with Det, then the Det value is encrypted with Rand. The most external layer of an onion is called *actual layer*, which corresponds to its strongest encryption algorithm. The cloud database can only see the actual layer of the onions, and has no access to inner layers nor to plaintext data. The first time that a new SQL operation is requested on a column, the outer layer of the appropriate onion is dynamically removed at runtime through the *adaptive layer removal* mechanism that exposes a layer supporting the requested operations. This layer becomes the new actual layer of the onion in the encrypted database. The layer removal mechanism is designed to ensure that the cloud provider can never access plaintext data. A detailed description is in Section 3.3.

Fig. 2 shows an example of the onions and layers structures by considering two plaintext columns having data types *int* and *varchar*. The integer column is encrypted with Onion-Eq, Onion-Ord, and Onion-Sum, and the string column is encrypted with Onion-Eq and Onion-Search. Each onion represents a column in the encrypted database structure. The actual layers of all the onions are set to Rand, that guarantees the best data confidentiality level but it does not allow computations on encrypted data. When an equality check is requested on the integer column the adaptive layer removal mechanism removes the Rand layer of Onion-Eq, thus leaving Det as its new actual layer.

3.2 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 table-level granularity to reduce communication overhead for retrieval, and to improve management of concurrent SQL operations [22]. We define all metadata information associated with a table as *table metadata*. Let us describe the structure of a table metadata by referring to Fig. 3.

Table metadata include 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 they also include a set of *column metadata* containing the name and the data type of the corresponding plain column (e.g., integer, varchar, datetime). Each set of column metadata is associated with as many sets of *onion metadata* as the number of onions associated with the column. Onion metadata describe all the encryption information about an onion and its layers, hence they are organized in a data structure that contains the following attributes:

- the *encrypted name* is the name of the encrypted column (i.e., the onion) in the encrypted cloud database;
- the *actual encryption layer* is the name of the most external layer of the encrypted data (e.g., Rand) stored in the column;
- the *field confidentiality* denotes which set of keys must be used to encrypt a column data, because only columns that share the same encryption keys can be joined; we identify three types of field confidentiality parameters: *self* denotes a private set of keys for the column, *multi-column* identifies the sharing of the same set of keys among two columns, *database* imposes the same set of keys on all columns of the same data type.
- the *onion* parameter identifies the type of onion that is used to encrypt data (e.g., Onion-Eq).

Each set of onion metadata is associated with as many sets of *layer metadata* as the number of layers required by the onion type. Each set of layer metadata includes an encryption key and a label identifying the corresponding encryption algorithm. The set of encryption keys for each onion is generated according to the field confidentiality parameter imposed on each encrypted column.

3.3 Encrypted Database Management

We now describe the three main operations involved in the encrypted database management: database creation, execution of SQL operations, and adaptive layer removal.

3.3.1 Database Creation

In the setup phase, the database administrator generates a *master key*, and uses it to initialize the architecture metadata. The master key is then distributed to legitimate clients. 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 data are the most important for this paper because they include the *set of onions* 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, they are automatically chosen by the encryption engine with respect to some tenant's policy. Typically, the default policy assumes that each column is associated with all the compatible onions, and the starting

layer of each onion is set to the strongest encryption algorithm. For example, integer columns are encrypted by default with Onion-Eq, Onion-Ord and Onion-Sum using Rand as the actual encryption layer (see Fig. 2).

3.3.2 Execution of SQL Operations

When a user/application wants to execute an operation on the cloud database, the client encryption engine analyzes the SQL command structure and identifies which tables, columns and SQL operators are involved. The client issues a request for the table metadata for each involved table, and decrypts the metadata with the master key. Then, the client determines whether the SQL operators are supported by the actual layers of the onions associated with the involved columns. If required, the client issues a request for layer removal in order to support the SQL operators at runtime. By using the information stored in the table metadata, the client is able to encrypt the parameters of the SQL operations: tables and columns names, and constant values. The client issues this new statement called *encrypted SQL operation* to the cloud database which transparently executes it over encrypted data. The encrypted results are decrypted using information contained in the metadata.

3.3.3 Adaptive Layer Removal

The *adaptive layer removal* is the process that dynamically removes the external layer of an onion in order to adaptively support SQL operations issued by legitimate clients.

Let us describe the details of the adaptive layer removal mechanism by referring to the following example. We consider a table T with columns *id* of type int and *name* of type string, and a tenant client preparing to issue the following statement to the encrypted cloud database: *SELECT * FROM T WHERE* $id < 10$. The client encryption engine analyzes the SQL statement, and identifies that the operation $id < 10$ has to be executed on the encrypted database. Then, the client reads the metadata and checks whether there is the Onion-Ord attribute associated with the column *id* because this is the only onion supporting the operator $<$. If the actual layer of Onion-Ord associated with *id* is set to Rand, then the client dynamically invokes a stored procedure on the cloud database that removes at runtime the Rand layer of Onion-Ord of the column *id*, thus leaving the Ope layer exposed. The client can now encrypt the SELECT query that contains the operation $id < 10$ and issue the encrypted query to the encrypted database, that executes it on the Ope layer of Onion-Ord. Any new SQL operation involving an order comparison on the column id does not require to invoke again the layer removal procedure because the actual layer of Onion-Ord is Ope.

The cloud database can execute the adaptive layer removal if and only if a legitimate client invokes the stored procedure and gives to it the decryption key of the most external encryption layer. As each layer has a different encryption key, the data remain encrypted and the cloud provider cannot access plaintext data. For security reasons, we also assume that the adaptive layer removal mechanism does never expose the Plain layer of an onion.

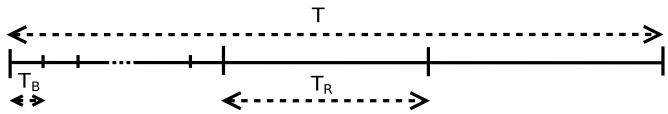

Fig. 4. Example of relationship among estimation (T), reservation (T_R) and billing (T_B) periods.

4 Cost Estimation of Cloud Database Services

We consider a tenant that is interested in estimating the cost of porting his database to a cloud platform. This porting is a strategic decision that must evaluate confidentiality issues and related costs over a medium-long term. For these reasons, we propose a model that includes the overhead of encryption schemes and the 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* [23], *EnterpriseDB* [24], *Windows Azure SQL Database* [25], and *Rackspace Cloud Database* [26].

4.1 Cost Model

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

$$Cost = f(Time, Pricing, Usage), \tag{1}$$

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 [27].
- *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 his subscription anytime; the *reservation* policy requires the tenant to commit in advance for a *reservation period*. Hence, we distinguish between *billing costs* that depend on resource usage and *reservation costs* denoting additional fees for commitment in exchange for lower pay-per-use prices [28]. Billing costs are billed periodically to the tenant every *billing period* T_B. Moreover, if the tenant adopts the reservation policy, the cloud provider requires the payment of the reservation cost at the beginning of each *reservation period* T_R. An example of the relationship among T (three years), T_R (one year) and T_B (one month) is represented in Fig. 4.

Pricing is the set of *reservation prices* and *billing prices*. Reservation prices $\{R_r\}$ refer to reservation periods, where $r = [1, \ldots, N_R]$ and $N_R = {}^{T}/_{T_R}$ is the number of reservation periods. Billing prices refer to billing periods $b = [1, \ldots, N_B]$, where $N_B = {}^{T}/_{T_B}$. Our model takes into account billing prices for all the resources considered by the main providers [28], [29], [30], [31], [32]: *uptime price* $\{p_b^h\}$, *storage price* $\{p_b^s\}$ and *network price* $\{p_b^n\}$.

Usage represents the amount of resources consumed by the tenant. It is defined as the set of *uptime usage* $\{h_b\}$, *storage usage* $\{s_b\}$ and *network usage* $\{n_b\}$.

The total cost of the cloud database service C can be estimated through the following equation:

$$C = \sum_{r=1}^{N_R} R_r + \sum_{b=1}^{N_B} \left[\mathcal{H}\left(h_b, p_b^h\right) + \mathcal{S}\left(s_b, p_b^s\right) + \mathcal{N}\left(n_b, p_b^n\right)\right], \quad (2)$$

where $\mathcal{H}(h_b, p_b^h)$ is the *uptime billing function*, $\mathcal{S}(s_b, p_b^s)$ is the *storage billing function* and $\mathcal{N}(n_b, p_b^n)$ is the *network billing function*. We observe that R_r represents *fixed* costs that do not vary with respect to *Usage*, while $\mathcal{H}, \mathcal{S}$ and $\mathcal{N}$ represent costs that vary with respect to database uptime, storage and network usage. Moreover, all prices may vary during the estimation period T due to price adjustments applied by the cloud provider. We observe that different cloud providers apply different billing criteria, thus we cannot detail the billing functions without losing generality. We propose detailed price models for popular cloud providers in Section 4.2.

4.2 Cloud Pricing Models

Popular cloud database providers adopt two different billing functions, that we call *linear* $\mathcal{L}$ and *tiered* $\mathcal{T}$. Let us consider a generic resource x. We define x_b as its usage at the b-th billing period and p_b^x as its price. If the billing function is linear, it can be computed as:

$$\mathcal{L}\left(x_b, p_b^x\right) = x_b \cdot p_b^x. \quad (3)$$

If the billing function is tiered, the cloud provider applies different prices to different ranges of resource usage. Let us define Z as the number of tiers, and $[\hat{x}_1, \ldots, \hat{x}_{Z-1}]$ as the set of thresholds that define all the tiers. The price is modeled as a piecewise function:

$$\hat{p}_b^x(x) = \begin{cases} \hat{p}_{b,1}^x, & x \le \hat{x}_1 \\ \hat{p}_{b,z+1}^x, & \hat{x}_z < x \le \hat{x}_{z+1},\ 1 \le z < Z-1 \\ \hat{p}_{b,Z}^x, & x \ge \hat{x}_{Z-1}, \end{cases} \quad (4)$$

where $[\hat{p}_{b,1}^x, \ldots, \hat{p}_{b,Z}^x]$ represents the set of prices associated with the tiers. If the resource usage is lower than the first threshold ($x_b \le \hat{x}_1$), then the billing function is defined as:

$$\mathcal{T}\left(x_b, p_b^x\right) = x_b \cdot \hat{p}_{b,1}^x. \quad (5)$$

Otherwise, we denote as $\tilde{x}_z$ the highest threshold that is lower than the usage x_b. Then the billing function can be defined as:

$$\mathcal{T}\left(x_b, p_b^x\right) = (x_b - \tilde{x}_z) \cdot \hat{p}_{b,z+1}^x + \sum_{j=0}^{z-1} (\hat{x}_{j+1} - \hat{x}_j) \cdot \hat{p}_{b,j+1.}^x \quad (6)$$

The uptime and the storage billing functions of *AmazonRDS* [23] are linear, while the network usage is a tiered billing function. On the other hand, the uptime billing functions of *AzureSQL* [25] is linear, while the storage and network billing functions are tiered.

4.3 Usage Estimation

While the uptime (h_b) is easily measurable, it is more difficult to estimate the usage of storage (s_b) and network (n_b) because 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. For clarity, we define s^p, s^e, s^a as the storage usage in the plaintext, encrypted, and adaptively encrypted databases for one billing period. Similarly, n^p, n^e, n^a represent network usage in the three configurations.

We assume that the tenant knows the database structure and the query workload, and that each column $a \in A$ stores r_a values. By denoting as v_a^p the average storage size of each plaintext value stored in column a, we estimate the storage of the plaintext database s^p as:

$$s^p = \sum_{a \in A} \left(r_a \cdot v_a^p\right). \quad (7)$$

This equation considers only the storage usage of tenant data, and it does not take into account disk usage of the database server, the operating system, and other necessary software. This assumption holds because we consider cloud DBaaS services. An estimate of storage size for Infrastructure as a Service should also include all these factors.

We define the query workload W as the set $\{(Q, \omega)\}$, where each couple (Q, ω) represents a query Q and the frequency ω with which the query is executed. We assume that ω is normalized, hence it is expressed as a rational number in the range $(0, 1]$ and the sum of all ω is equal to 1.

The tenant can estimate the average network consumption of a query on the plaintext database through the following equation:

$$n^p = k \cdot \sum_{(Q,\omega) \in W} \left(\omega \cdot \sum_{a \in Q} v_a^p\right), \quad (8)$$

where $a \in Q$ represents an attribute that is retrieved by the query Q, and k is a corrective factor that takes into account network overheads caused by communication protocols. We do not model the value of k because it depends on network and applications protocols used by the cloud database (e.g., a PostgreSQL ODBC connection over SSL). Hence we propose that the tenant evaluates it experimentally for a specific software configuration. An experimental evaluation of k for our prototype is presented in Section 6.1.

Encrypting the content of a database increments its storage size because encryption algorithms expand the plaintext data. We define Φ as the set of SQL-aware encryption algorithms available in the system [33] and $\phi \in \Phi$ as one encryption algorithm. It is then possible to compute the encrypted size v_a^e of the attribute a encrypted with the algorithm ϕ as:

$$v_a^e = \mathcal{E}_\phi\left(v_a^p\right), \quad (9)$$

where $\mathcal{E}_\phi$ is a function that depends on the algorithm ϕ. Table 1 summarizes the $\mathcal{E}_\phi$ functions for the encryption algorithms currently implemented in our prototype.

If the tenant does not use adaptive strategies (that is, each column is encrypted through only one SQL-aware encryption algorithm) he can estimate the storage size s^e and the network consumption n^e of the encrypted database by

TABLE 1
Functions $\mathcal{E}_\phi$ for Different Encryption Algorithms

Type	ϕ	$\mathcal{E}_\phi(v_a^p)$
Random	*IV — AES-CBC* [20]	$16 + v_a^p\lceil v_a^p/16\rceil$
	IV — Blowfish [21]	$8 + v_a^p\lceil v_a^p/8\rceil$
Deterministic	*AES-CBC* [20]	$v_a^p\lceil v_a^p/16\rceil$
	IV — Blowfish [21]	$v_a^p\lceil v_a^p/8\rceil$
Order preserving	*Boldyreva et all.* [12]	$v_a^p \times 2$
Sum	*Paillier* [13]	256

replacing v_a^p with v_a^e in Equations (7) and (8). If we consider adaptive encryption techniques, the storage overhead increases, because multiple layers of encryption are stacked over each plaintext database column. To model storage and network overheads for adaptively encrypted databases, we define θ as the ordered set of encryption algorithms that represents an onion. As an example, we may represent Onion-Eq by using $\theta_E = \langle \phi_D, \phi_R \rangle$, where $\phi_D, \phi_R \in \Phi$ represent Deterministic and Random encryption algorithms.

We can estimate the adaptively encrypted storage size v_a^a of attribute a through onion θ as:

$$v_a^a = \mathcal{F}\big(v_a^p, \theta\big), \tag{10}$$

where $\mathcal{F}$ is defined as the composition of the functions $\mathcal{E}_\phi$ for all the algorithms ϕ included in the ordered set θ. For example, if we consider Onion-Eq θ_E:

$$\mathcal{F}\big(v_a^p, \theta_E\big) = \mathcal{E}_{\phi_R} \circ \mathcal{E}_{\phi_D} = \mathcal{E}_{\phi_R}\big(\mathcal{E}_{\phi_D}\big(v_a^p\big)\big). \tag{11}$$

Since each plaintext column a may be associated with multiple onion-encrypted columns, we define Θ_a as a set of onions associated with a. As an example, let us estimate the storage size of a plaintext *bigint* column encrypted through the set of onions $\Theta_a = \{\theta_E, \theta_O, \theta_S\}$ (Onion-Eq, Onion-Ord and Onion-Sum). Since the storage size of a *bigint* is $v_a^p = 8$ bytes, the corresponding encrypted storage size can be computed as:

$$\begin{aligned}&\mathcal{F}(8, \theta_E) + \mathcal{F}(8, \theta_O) + \mathcal{F}(8, \theta_S)\\ &\quad = \mathcal{F}(8, \langle \phi_D, \phi_R \rangle) + \mathcal{F}(8, \langle \phi_O, \phi_R \rangle) + \mathcal{F}(8, \langle \phi_S, \phi_R \rangle)\\ &\quad = 16 + 24 + 272 = 312.\end{aligned}$$

The tenant can estimate the storage size of the adaptively encrypted database s^a by using Equations (7) and (11):

$$s^a = \sum_{a \in A}\left(r_a \cdot \sum_{\theta \in \Theta_a} \mathcal{F}\big(v_a^p, \theta\big)\right). \tag{12}$$

Network consumption is related to the storage size of adaptively encrypted data retrieved by the queries. Retrieving adaptively encrypted data related to a plaintext column a requires to choose one of the onions associated with a. Our design choice is to minimize the network overhead by selecting the onion with minimal storage overhead. Hence, we define $\mathcal{F}^\star$ as:

$$\mathcal{F}^\star\big(v_a^p\big) = \min\big\{\mathcal{F}\big(v_a^p, \theta\big) \,|\, \theta \in \Theta_a\big\}. \tag{13}$$

The tenant can estimate the network consumption of the adaptively encrypted database n^a by replacing v_a^p with $\mathcal{F}^\star(v_a^p)$ in Equation (8).

5 Performance Evaluation

This section aims to verify whether the overheads of adaptive encryption represent an acceptable compromise between performance and data confidentiality for the tenants of cloud database services. To this purpose, we design a suite of performance tests that allow us to evaluate the impact of encryption and adaptive encryption on response times and throughput for different network latencies and for increasing numbers of concurrent clients. The TPC-C standard benchmark is used as the workload model for the database services. The experiments are carried out in Emulab [34], which provides us with a set of machines in a controlled environment. Each client machine runs the Python client prototype of our architecture on a pc3000 machine having a single 3 GHz processor, 2 GB of RAM and two 10,000 RPM 146 GB SCSI disks. The database server is PostgreSQL 9.1 running on a d710 machine having a quad-core Xeon 2.4 GHz processor, 12 GB of RAM and a 7,200 RPM 500 GB SATA disk.

The current version of the prototype supports the main SQL operations (SELECT, DELETE, INSERT and UPDATE) and the WHERE clause. We consider three TPC-C compliant databases having 10 warehouses:

- *Plaintext (PLAIN)* is based on plaintext data.
- *Encrypted (ENC)* refers to a statically encrypted database where each column is encrypted at design time with only one encryption algorithm.
- *Adaptively encrypted (ADAPT)* refers to an encrypted database in which each column is encrypted with all the onions supported by its data type (Section 3.3).

In the ENC and ADAPT configurations each column is set to the highest encryption layer that supports the SQL operations of the TPC-C workload. During each TPC-C test lasting for 300 seconds, we monitor the number of executed TPC-C transactions, and the response times of all the SQL operations from the standard TPC-C workload. We repeat the test for each database configuration (PLAIN, ENC and ADAPT) for increasing number of clients (from 5 to 20), and for increasing network latencies (from 0 to 120 ms). To guarantee data consistency the three databases use repeatable read (snapshot) isolation level [35].

The experiments aim to evaluate the overhead caused by static and adaptive encryption in terms of system throughput and response time. In Figs. 5 and 6, we report the number of committed TPC-C transactions per minute executed on the three cloud database configurations for 5 and 20 concurrent clients, respectively. We can appreciate that in both cases, and in all other results not reported for space reasons, the throughput of the ENC database is close to that of the PLAIN database. Moreover, as the network latency increases, even the performance of the ADAPT database tends to that of the other two configurations, and it is quite

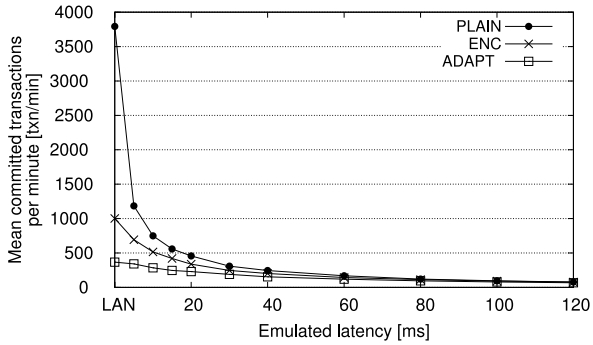

Fig. 5. TPC-C throughput with 5 clients.

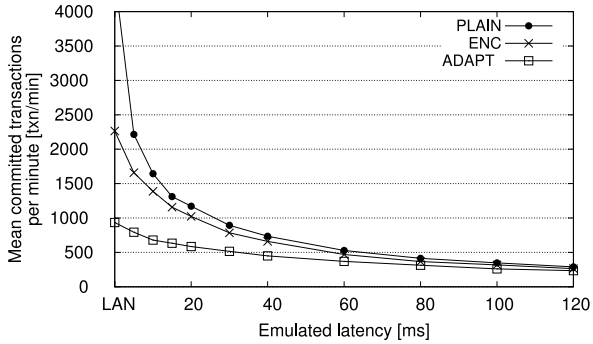

Fig. 6. TPC-C throughput with 20 clients.

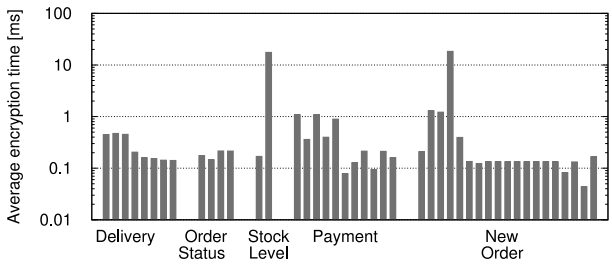

Fig. 7. Encryption times of the SQL operations composing the TPC-C workload (ENC configuration).

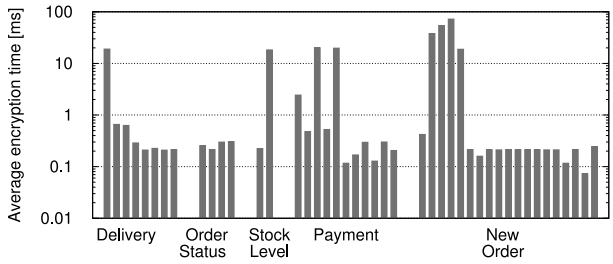

Fig. 8. Encryption times of the SQL operations composing the TPC-C workload (ADAPT configuration).

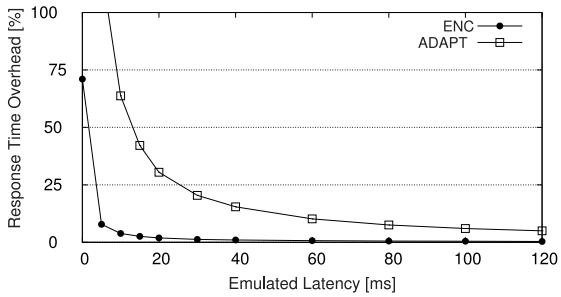

Fig. 9. SELECT response time overhead—10 clients.

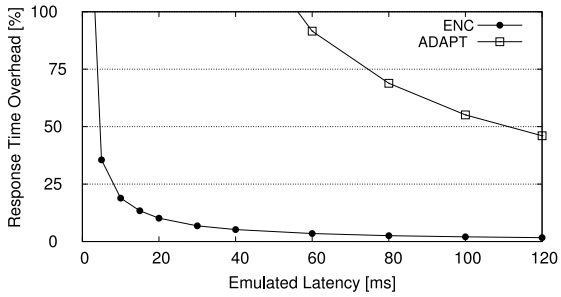

Fig. 10. INSERT response time overhead—10 clients.

close to them for latencies higher than 60 ms, which are realistic for typical cloud database scenarios. This is an extremely positive result because it demonstrates that adaptive encryption can be realistically used for cloud database services.

In Figs. 7 and 8, we consider the ENC and ADAPT configurations and we report the average encryption times required by each SQL operation composing the TPC-C workload. The results are grouped on the basis of the five transactions of the TPC-C workload. Most SQL operations have similar costs, but in the ADAPT configuration (Fig. 8) some operations require much higher encryption times with respect to ENC (Fig. 7). Further analyses demonstrate that the two peaks in ENC are related to SQL operations requiring Ope encryption, and that the majority of peaks in ADAPT are related to INSERT operations combined with Ope encryption. This is due to the encryption time of Ope that is two or three orders of magnitude higher than that of Rand and Det [10]. In the ADAPT configuration, every integer column is associated by default with Onion-Eq and Onion-Ord, which has an Ope layer. Hence, every insertion of a value into an integer column requires an Ope encryption, and this causes a significant increase of the overall encryption overhead. On the other hand, in the ENC configuration the Ope layer is associated only with the columns in which the TPC-C workload requires support for the order comparison operators.

We now investigate the impact of network latency on response time with regard to the overhead caused by static and adaptive encryption. We evaluate the response times of the most popular SELECT, DELETE, INSERT and UPDATE operations of the TPC-C workload for different numbers of clients. In Figs. 9 and 10 we report the response times overheads of the SELECT and INSERT operations for 10 clients as function of increasing network latencies. Overheads of the ENC and ADAPT configurations are measured with respect to the PLAIN database response time. In Fig. 9, we observe that the overhead of the SELECT query tends to be masked for latencies higher than 60 ms in both the ADAPT and ENC configurations. This is an important conclusion because the results of the SELECT query are representative of the performance related to the DELETE and UPDATE operations. On the other hand, the INSERT operation is critical in terms of overhead. As shown in Fig. 10, the ADAPT configuration has response times overheads much higher

TABLE 2
Validation of Storage Usage of TPC-C Compliant Databases

W	Estimated Storage [MB] (Error %)		
	PLAIN	ENC	ADAPT
1	99 (1.0)	187 (5.6)	273 (6.6)
5	453 (1.6)	859 (5.4)	1270 (6.3)
10	894 (1.5)	1698 (5.3)	2516 (6.2)

TABLE 3
Validation of Outgoing Network Usage of TPC-C Compliant Databases

	Network Usage [Bytes]		Error
	Estimated	Measured	
ENC	13175	13329	-1.2%
ADAPT	13671	13862	-1.4%

than those related to the ENC configuration for any network latency. This result has a twofold motivation: the higher number of encryptions required by the adaptive architecture to encrypt all parameters of the different onions; the high computational cost characterizing the Ope encryption as shown in previous histograms.

All experimental results show that network latencies higher than 60 ms, which are typical of most cloud database environments, make the adaptive encryption overhead almost negligible when considering the overall set of operations of the TPC-C benchmark. However, in the ADAPT configuration, for some SQL operations involving the Ope encryption or for the encryption of a high number of parameters through several encryption layers (e.g., INSERT), the impact on the response time is visible even for network latencies higher than 120 ms.

If the workload is characterized by many INSERT operations, we can conclude that it is a tenant's duty to solve the tradeoff between accepting adaptive encryption overheads and paying the costs related to an entire database re-encryption when workload changes in statically encrypted databases. It is likely that this tradeoff can be solved on the basis of the expected variability of the workload. Possible improvements can be achieved by parallel encryption algorithms that can leverage multi-threading over different cores, but this research is out of the scope of this paper.

On the positive hand, we observe that the presented ADAPT configuration represents a worst case scenario that is fully adaptive, because all database columns are encrypted with all the onions supported by its data type. On the other hand, the ENC configuration represents a best case scenario that is completely static, because the user manually defines the single encryption scheme to use on each database column. We observe that a tenant may choose a partially adaptive configuration in which a subset of columns are encrypted with adaptive strategies and others are statically encrypted. As a consequence, performance of adaptive encryption for many realistic workloads fall between the ENC and ADAPT scenarios presented in this section.

6 Cost Evaluation

In this section we demonstrate the feasibility of the proposed cost model by applying it to PLAIN, ENC and ADAPT configurations (see Section 5) in real cloud database services. We initially validate the usage estimation methodology presented in Section 4.3. We then analyze how costs vary for different cloud providers and resource usages. We finally evaluate tenant's costs over a medium-term period equal to three years by considering realistic resource usage increments and cloud price reductions.

6.1 Validation of the Usage Estimation

To validate the usage estimation model, we perform several experiments based on the TPC-C benchmark.

First of all, we validate the storage usage estimation model. We deploy nine TPC-C compliant databases of three different sizes: 1, 5 and 10 warehouses (the number of warehouses is the TPC-C parameter that influences the initial database size). For each size, we generate three database configurations: PLAIN, ENC and ADAPT. Results are summarized in Table 2. Estimated storage of PLAIN, ENC and ADAPT are calculated by using the analytical model presented in Section 4.3. For each estimated value, we report the estimation error with respect to the measured database size. Errors are expressed as a percentage. We observe that the proposed model always overestimates the database size. However, errors show that estimations are close to measured sizes. For PLAIN databases, the error is always below 2%, while for ENC and ADAPT databases the error is always between 5% and 6%.

We then validate the network usage estimation model. We deploy PLAIN, ENC and ADAPT TPC-C compliant databases, each having 10 warehouses. We observe that network consumption is invariant with respect to the number of warehouses, because it only depends on encryption and query workload. We measure the network usage of the PLAIN database, and we obtain an average of 7,162 Bytes per transaction. By using Equation (8), we estimate $n^p = k \cdot 548$. Hence, we determine $k = 13.07$. Then we use this value of k to determine the estimated network usage of ENC and ADAPT configurations. We compare these values with the experimentally measured network usages. Results are summarized in Table 3. Estimations are quite accurate, since we achieve errors of -1.2% and -1.4% for the ENC and ADAPT configurations, respectively.

The validation demonstrates the efficacy of the proposed analytical usage estimation methodology in the TPC-C workload. Costs evaluations proposed in the following sections are based on the same usage estimations.

6.2 Analysis of Cloud Database Costs

We analyze cloud database costs with respect to different cloud provider offers and different storage and network usages. We consider a billing period equal to one month, and 24/7 availability (730 uptime hours per month).

We initially estimate the monthly costs of a cloud database service in the PLAIN, ENC and ADAPT configurations with respect to a plaintext storage usage of 100 GB and a plaintext network usage of 100 GB. In Table 4, we report the results for the following cloud instances: *Small*, *Large*, and *High Memory: Double Extra Large* from Amazon RDS [28]; *Premium P1* and *Premium P2* from SQL Azure [31].

TABLE 4
Cost Evaluation of Cloud Database Services for One Billing Period (Month) with 24/7 Uptime, 100 GB of Plaintext Storage Size and 100 GB of Plaintext Network Usage

Name	Pricing		Fixed costs		Result	PLAIN	ENC	ADAPT
	p^s	p^n	$R/12$	$\mathcal{H}$				
Amazon RDS Small	0.10 $/GB	0.12 $/GB	14.08 $	18.25 $	C	54.33 $	73.36 $	83.41 $
					$\mathcal{S}\ (\mathcal{S}/C)$	10.00 $ (18.4%)	18.99 $ (25.9%)	28.14 $ (33.7%)
					$\mathcal{N}\ (\mathcal{N}/C)$	12.00 $ (22.1%)	22.03 $ (30.0%)	22.93 $ (27.5%)
Amazon RDS Large	0.10 $/GB	0.12 $/GB	52.75 $	69.35 $	C	144.10 $	163.12 $	173.17 $
					$\mathcal{S}\ (\mathcal{S}/C)$	10.00 $ (7.0%)	18.99 $ (11.6%)	28.14 $ (16.2%)
					$\mathcal{N}\ (\mathcal{N}/C)$	12.00 $ (8.3%)	22.03 $ (13.5%)	22.93 $ (13.2%)
Amazon RDS HM:2XL	0.10 $/GB	0.12 $/GB	131.50 $	181.04 $	C	334.54 $	353.56 $	363.61 $
					$\mathcal{S}\ (\mathcal{S}/C)$	10.00 $ (3.0%)	18.99 $ (5.4%)	28.14 $ (7.7%)
					$\mathcal{N}\ (\mathcal{N}/C)$	12.00 $ (3.6%)	22.03 $ (6.2%)	22.93 $ (6.3%)
Azure SQL Premium P1	0.075 $/GB	0.095 $/GB	0.00 $	337.63 $	C	354.63 $	369.31 $	376.88 $
					$\mathcal{S}\ (\mathcal{S}/C)$	7.50 $ (2.1%)	14.24 $ (3.9%)	21.11 $ (5.6%)
					$\mathcal{N}\ (\mathcal{N}/C)$	9.50 $ (2.7%)	17.44 $ (4.7%)	18.15 $ (4.8%)
Azure SQL Premium P2	0.075 $/GB	0.095 $/GB	0.00 $	675.25 $	C	692.25 $	706.93 $	714.51 $
					$\mathcal{S}\ (\mathcal{S}/C)$	7.50 $ (1.1%)	14.24 $ (2.0%)	21.11 $ (3.0%)
					$\mathcal{N}\ (\mathcal{N}/C)$	9.50 $ (1.4%)	17.44 $ (2.5%)	18.15 $ (2.5%)

The left part of Table 4 reports the storage prices p^s, the network prices p^n, the total uptime costs $\mathcal{H}$, and the annual reservation prices R reported as monthly costs. We observe that the storage and network prices do not change for different instances of the same cloud provider, whereas the reservation cost R and the uptime cost $\mathcal{H}$ may vary depending on the chosen instance. The right part of Table 4 reports the estimations of the billing costs. For each instance, we estimate the monthly cost C of the PLAIN, ENC and ADAPT configurations, expressed in USD [$], and the influence of storage cost $\mathcal{S}$ and of network cost $\mathcal{N}$ on the billing cost C (i.e., $\mathcal{S}/C$ and $\mathcal{N}/C$) that are expressed as percentages. The results from Table 4 show the impact of static and adaptive database encryption on the monthly billing costs for different instance classes.

It is interesting to observe that the cost differences between plain and encrypted cloud databases tend to remain constant for different instances of the same cloud provider, because their unit prices p^s and p^n do not vary. On the other hand, higher reservation and uptime costs related to more powerful instances result in a reduced impact of $\mathcal{S}$ and $\mathcal{N}$ on C. Since encryption affects only storage and network usages, the cost overhead due to encryption is partially masked in the case of higher reservation and uptime prices. For example, the cost overhead between the PLAIN and ADAPT configurations of Amazon RDS Large instance is $\approx 20.2\%$, whereas the same overhead in the Double Extra Large instance is only $\approx 8.7\%$.

We now analyze how different network and storage usages affect the costs of PLAIN, ENC and ADAPT cloud databases. In Table 5, we estimate the monthly costs of Amazon RDS Large [28] and SQL Azure Premium P1 [31] for different combinations of plaintext storage and network

TABLE 5
Monthly Costs Increases for Different Combinations of Plaintext Storage and Network Usages

Result	Plaintext Network	Amazon RDS (Large)					SQL Azure (Premium P1)				
		Plaintext Storage Size [GB]					Plaintext Storage Size [GB]				
		10	50	100	500	1000	10	50	100	500	1000
$PLAIN$ C [$]	10 GB	124.30	128.30	133.30	173.30	223.30	339.33	342.33	346.08	376.08	413.58
	50 GB	129.10	133.10	138.10	178.10	228.10	343.13	346.13	349.88	379.88	417.38
	100 GB	135.10	139.10	144.10	184.10	234.10	347.88	350.88	354.63	384.63	422.13
	500 GB	183.10	187.10	192.10	232.10	282.10	385.88	388.88	392.63	422.63	460.13
	1000 GB	243.10	247.10	252.10	292.10	342.10	433.38	436.38	440.13	470.13	507.63
ENC $C_{\%}$[%]	10 GB	1.5%	4.3%	7.5%	26.5%	40.7%	0.4%	1.2%	2.2%	9.2%	16.5%
	50 GB	4.6%	7.1%	10.1%	28.1%	41.6%	1.4%	2.1%	3.1%	9.9%	17.1%
	100 GB	8.1%	10.4%	13.2%	29.9%	42.7%	2.5%	3.2%	4.1%	10.8%	17.9%
	500 GB	27.9%	29.2%	30.8%	41.0%	49.6%	10.5%	11.1%	11.8%	17.4%	23.3%
	1000 GB	41.6%	42.4%	43.4%	49.7%	55.6%	18.5%	19.0%	19.6%	24.1%	28.9%
$ADAPT$ $C_{\%}$[%]	10 GB	2.3%	7.9%	14.4%	53.0%	81.7%	0.7%	2.2%	4.2%	18.3%	33.1%
	50 GB	5.6%	10.9%	17.1%	54.0%	81.9%	1.7%	3.2%	5.1%	19.0%	33.6%
	100 GB	9.4%	14.4%	20.2%	55.2%	82.2%	2.9%	4.4%	6.3%	19.9%	34.3%
	500 GB	30.8%	34.1%	37.9%	62.6%	83.7%	11.6%	12.9%	14.5%	26.3%	39.0%
	1000 GB	45.7%	47.9%	50.6%	68.5%	85.0%	20.3%	21.4%	22.8%	32.9%	43.9%

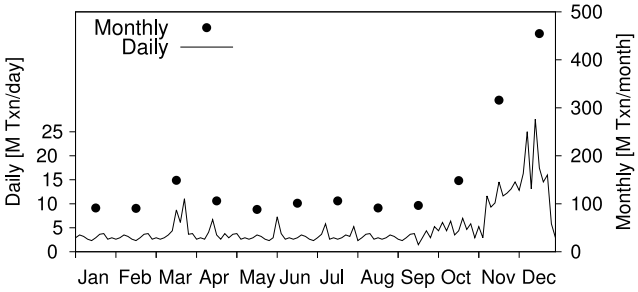

Fig. 11. Workload trend in terms of committed transactions per day (and month) during a year for a typical e-commerce workload.

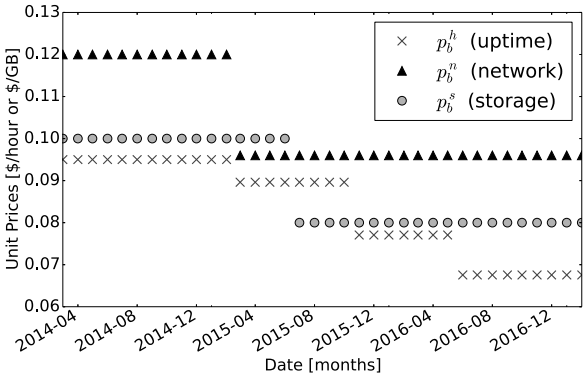

Fig. 12. Expected price reductions for Amazon RDS from March 2014 to February 2017.

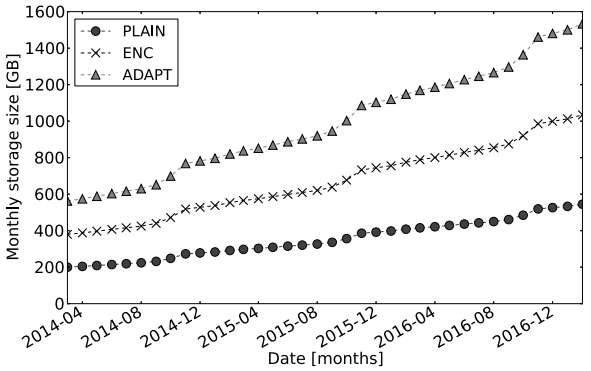

Fig. 13. Expected storage usage in the DYNAMIC ($+2\%$) scenario for the upcoming three years.

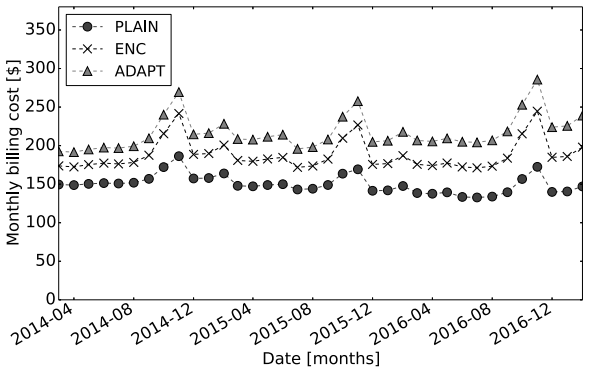

Fig. 14. Billing costs of the DYNAMIC ($+2\%$) scenario in the three years period.

usages. The cloud prices related to these two instances are reported in the left part of Table 4. In Table 5 we report the billing cost C of the PLAIN configuration, expressed in USD [$], and the estimated monthly *cost overhead* $C_{\%}$ in the ENC and ADAPT configurations, expressed as percentage. The results of this table show that in Amazon RDS $C_{\%}$ is always lower than 60 and 90 percent in the ENC and ADAPT configurations respectively, whereas in SQL Azure it is always lower than 30% (ENC) and 50% (ADAPT). In SQL Azure Premium P1 the values of $C_{\%}$ are lower than those of Amazon RDS Large because in P1 the fixed costs $\mathcal{H}$ and R are higher and the unit prices p^s and p^n are lower. This conclusion is valid even for other instances characterized by high reservation and uptime costs, which are not reported for space reasons.

Moreover, we observe that for plaintext storage sizes equal to 1,000 GB, $C_{\%}$ remains stable for increasing plaintext network consumptions. This result is motivated by the fact that for higher database sizes, the storage cost $\mathcal{S}$ becomes dominant with respect to the network cost $\mathcal{N}$, especially in the ADAPT configuration. This stability of cost overheads for almost any network usage is a quite interesting result because in real database workloads the network usage is usually characterized by unpredictable variability. This conclusion can reduce the tenant concerns about the monthly bill for cloud services.

6.3 Cost Evaluation over a Medium-Term Period

We now consider an application of the proposed cost model in a realistic scenario in which a tenant wants to estimate the costs of moving his e-commerce database to the cloud for the upcoming three years. In particular, we consider a scenario in which the storage and network usages tend to increase because of higher numbers of database operations and/or customers, and the cloud prices tend to decrease over a three years horizon. In this DYNAMIC scenario, we assume that the initial month for the cost estimation is March 2014. The tenant initial database size is 200 GB, and the cloud database service is subject to the TPC-C workload [36], and the workload intensity in terms of committed transactions per day is characterized by the trend shown in Fig. 11. The average network usage determined by this workload is approximately 2 GB/day, with a peak of about 15 GB/day during the holiday season beginning at Thanksgiving. The cloud prices are related to an Amazon RDS Large instance [23], and we assume that the tenant can leverage the likely cloud price reductions. By assuming that in the period of interest the Amazon RDS price reductions of the past three years [27] repeat their trend in the upcoming three years, we represent the expected uptime (p_b^h), network (p_b^n) and storage (p_b^s) prices in Fig. 12.

The first analysis refers to a monthly workload increase equal to 2% with respect to the trend shown in Fig. 11. We observe that the TPC-C standard defines a relation between the number of committed transactions and the storage growth over time. According to this relation and the considered workload, Fig. 13 reports the expected storage usage for the upcoming three years referring to the PLAIN, ENC and ADAPT configurations. As expected, the storage usages of the ENC and ADAPT configurations are higher than that of the PLAIN configuration, and this difference tends to increase. It is interesting to evaluate the final effect caused

TABLE 6
Costs of the Cloud Database Service during the Three Years Period in STATIC and DYNAMIC Scenarios

	STATIC [$]		DYNAMIC (+2%) [$]				DYNAMIC (+4%) [$]				DYNAMIC (+9%) [$]			
	Yearly	Total	Y 1	Y 2	Y 3	Total	Y 1	Y 2	Y 3	Total	Y 1	Y 2	Y 3	Total
PLAIN	2,464	7,392	2,530	2,338	2,180	7,048	2,554	2,404	2,309	7,267	2,613	2,567	2,631	7,812
ENC	2,785	8,357	2,910	2,776	2,724	8,410	2,954	2,898	2,965	8,817	3,063	3,202	3,567	9,833
ADAPT	3,015	9,045	3,184	3,112	3,151	9,448	3,234	3,261	3,457	9,952	3,359	3,634	4,221	11,214

by the opposite contributions related to price reductions and workload increase. By reporting the monthly billing costs of PLAIN, ENC and ADAPT databases in Fig. 14, we can observe that, despite a workload increase equal to 2 percent per month, the billing costs of ADAPT configuration remain approximately stable.

If we extend the cost evaluation of the three years period to different monthly workload increases, then we can evaluate the so called *break even point* for each database configuration, that is, the value of monthly workload increase at which the annual costs of the cloud database service remain approximately constant. In this evaluation we refer to the price reductions reported in Fig. 12. By computing the break even point, the tenant can estimate whether the monthly workload increase will cause increments or decrements of the annual costs. Since PLAIN, ENC and ADAPT configuration determine different resource usages, the break even point is different for each configuration.

An analysis of the break even points is presented in Table 6, in which we report the annual and triennial costs with respect to the PLAIN, ENC and ADAPT configurations and different monthly workload increases: $2\%, 4\%$ and 9%. Lines of the table with a gray background show annual costs that are almost constant for the three different configurations, hence they identify the monthly workload increases that represent the break event points for ADAPT, ENC and PLAIN. We observe that the break even points of the two encrypted configurations correspond to monthly workload increases that are lower than the break even point of the plaintext configuration. This occurs because, even if the three configurations are subject to the same workload, the actual resource usages of ENC and ADAPT are higher due to encryption and adaptivity overheads. Just as a term of comparison, in Table 6 we report also the STATIC scenario in which we assume that during the three years period the storage size of the database remains equal to 200 GB, the network usage depends only on the trend of Fig. 11, and cloud prices do not vary with respect to the Amazon RDS Large prices at the end of February 2014 (second line in Table 4). We highlight that the differences between costs estimations of the STATIC and DYNAMIC scenarios are higher for encrypted configurations, especially if adaptive encryption strategies are used. A tenant can leverage the break even point analysis to quantify the annual and triennial savings or major expenses when the monthly workload increase is below or over the break even point.

7 Conclusions

There are two main tenant concerns that may prevent the adoption of the cloud as the fifth utility: data confidentiality and costs. This paper addresses both issues in the case of cloud database services. These applications have not yet received adequate attention by the academic literature, but they are of utmost importance if we consider that almost all important services are based on one or multiple databases.

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 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.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Comput. Syst.*, vol. 25, no. 6, pp. 599–616, 2009.

[2] T. Mather, S. Kumaraswamy, and S. Latif, *Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2009.

[3] H.-L. Truong and S. Dustdar, "Composable cost estimation and monitoring for computational applications in cloud computing environments," *Procedia Comput. Sci.*, vol. 1, no. 1, pp. 2175–2184, 2010.

[4] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: The montage example," in *Proc. ACM/IEEE Conf. Supercomputing*, 2008, pp. 1–12.

[5] H. Hacigümüş, B. Iyer, and S. Mehrotra, "Providing database as a service," in *Proc. 18th IEEE Int. Conf. Data Eng.*, Feb. 2002, pp. 29–38.

[6] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *Proc. 17th ACM Conf. Comput. Commun. Security*, 2010, pp. 735–737.

[7] Google. (2014, Mar.). Google Cloud Platform Storage with server-side encryption [Online]. Available: http://googlecloudplatform.blogspot.it/2013/08/google-cloud-storage-now-provides.html.

[8] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model," in *Proc. ACM SIGMOD Int'l Conf. Manage. Data*, Jun. 2002, pp. 216–227.

[9] L. Ferretti, M. Colajanni, and M. Marchetti, "Distributed, concurrent, and independent access to encrypted cloud databases," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 437–446, Feb. 2014.

[10] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting confidentiality with encrypted query processing," in *Proc. 23rd ACM Symp. Operating Systems Principles*, Oct. 2011, pp. 85–100.
[11] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st ACM Symp. Theory Comput.*, May. 2009, pp. 169–178.
[12] A. Boldyreva, N. Chenette, and A. O'Neill, "Order-preserving encryption revisited: Improved security analysis and alternative solutions," in *Proc. 31st Annu. Int. Conf. Adv. Cryptology*, Aug. 2011, pp. 578–595.
[13] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. 17th Int. Conf. Theory Appl. Cryptographic Tech.*, May 1999, pp. 223–238.
[14] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Security Privacy*, May 2000, pp. 44–55.
[15] L. Ferretti, F. Pierazzi, M. Colajanni, and M. Marchetti, "Security and confidentiality solutions for public cloud database services," presented at the 7th Int. Conf. Emerging Security Information, Systems and Technology, Barcelona, Spain, Aug. 2013.
[16] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, Jan. 2008.
[17] L. Popa, S. Ratnasamy, G. Iannaccone, A. Krishnamurthy, and I. Stoica, "A cost comparison of data center network architectures," in *Proc. ACM Int. Conf. Emerging Netw. Experiments Technol.*, 2010, pp. 16:1–16:12.
[18] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.
[19] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[20] J. Daemen and V. Rijmen, *The Design of Rijndael: AES – The Advanced Encryption Standard*. New York, NY, USA: Springer, 2002.
[21] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (blowfish)," in *Proc. Cambridge Security Workshop Fast Softw. Encryption*, Dec. 1993, pp. 191–204.
[22] L. Ferretti, M. Colajanni, and M. Marchetti, "Supporting security and consistency for cloud database," in *Proc. 4th Int. Symp. Cyberspace Safety Security*, Dec. 2012, pp. 179–193.
[23] Amazon Web Services. (2014, Mar.). Amazon relational database service [Online]. Available: http://aws.amazon.com/rds.
[24] EnterpriseDB. (2014, Mar.). Postgres plus cloud database [Online]. Available: http://enterprisedb.com/cloud-database.
[25] Microsoft. (2014, Mar.). Windows azure SQL database [Online]. Available: http://www.windowsazure.com/en-us/services/data-management.
[26] Rackspace. (2014, Mar.). Rackspace cloud database [Online]. Available: http://www.rackspace.com/cloud/databases.
[27] Amazon. (2014, Mar.). Amazon web services blog [Online]. Available: http://aws.typepad.com/aws/price-reduction/.
[28] Amazon RDS Pricing. (2014, Mar.). Amazon relational database pricing [Online]. Available: http://aws.amazon.com/rds/pricing.
[29] EnterpriseDB. (2014, Mar.). Postgres plus cloud database pricing [Online]. Available: http://www.enterprisedb.com/cloud-database/pricing-amazon.
[30] Microsoft. (2014, Mar.). Windows azure SQL database web & business pricing [Online]. Available: http://www.windowsazure.com/en-us/pricing/details/sql-database/#service-webandbusiness.
[31] Microsoft (2014, Mar.). Windows azure SQL database premium pricing [Online]. Available: http://www.windowsazure.com/en-us/pricing/details/sql-database/#service-premium.
[32] Rackspace. (2014, Mar.). Rackspace cloud database pricing [Online]. Avaialble: http://www.rackspace.com/cloud/databases/pricing.
[33] L. Ferretti, M. Colajanni, and M. Marchetti, "Access control enforcement of query-aware encrypted cloud databases," in *Proc. Fifth IEEE Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2013, pp. 717–722.
[34] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," in *Proc. 5th USENIX Conf. Operating Syst. Design Implementation*, Dec. 2002, pp. 255–270.
[35] A. Fekete, D. Liarokapis, E. O'Neil, P. O'Neil, and D. Shasha, "Making snapshot isolation serializable," *ACM Trans. Database Systems*, vol. 30, no. 2, pp. 492–528, 2005.
[36] TPC. (2014, Mar.). Tpc-c on-line transaction processing benchmark [Online]. Available: http://www.tpc.org/tpcc.

Luca Ferretti received the master's degree in computer engineering from the University of Modena and Reggio Emilia, Italy, in 2012, where he is currently working toward the PhD degree at the International Doctorate School in Information and Communication Technologies (ICT). His research focuses on information security, and cloud architectures and services.

Fabio Pierazzi received the master's degree in computer engineering from the University of Modena and Reggio Emilia, Italy, in 2013, where he is currently working toward the PhD degree at the International Doctorate School in Information and Communication Technologies (ICT). His research interests include security analytics and cloud architectures.

Michele Colajanni received the master's degree in computer science from the University of Pisa, and the PhD degree in computer engineering from the University of Roma in 1992. Since 2000, he has been a full professor of computer engineering at the University of Modena and Reggio Emilia. He manages the Interdepartment Research Center on Security and Safety (CRIS), and the Master in "Information Security: Technology and Law." His research interests include security of large scale systems, performance and prediction models, Web and cloud systems.

Mirco Marchetti received the PhD degree in Information and Communication Technologies (ICT) in 2009. He holds a postdoc position at the Interdepartment Research Center on Security and Safety (CRIS) at the University of Modena and Reggio Emilia. He is interested in intrusion detection, cloud security, and in all aspects of information security.