This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2314644, IEEE Transactions on Cloud Computing

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 trade-off 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 workload during a medium-term period.

Index Terms—Cloud Database, Confidentiality, Encryption, Adaptivity, Cost Estimation 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, which allow the execution of SQL operations over encrypted data, either suffer from performance limits (e.g., [8]) or they require the choice of which encryption scheme must be adopted for each database column and SQL operations (e.g., [9]). These latter proposals are fine when the set of queries can be statically determined at design time, while in this paper we are interested to 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 proposed 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 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. 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 feasibility in a cloud context, and storage and network costs estimation. In this paper, we investigate each of these issues and we reach original conclusions in terms of prototype implementation, performance evaluation, and cost evaluation.

We implement the first proxy-free architecture for adaptive encryption of cloud databases. It does not limit the availability, elasticity and scalability of a plain cloud database, because concurrent clients can issue parallel operations without passing through some centralized component as in alternative architectures [10]. We evaluate the performance through this prototype implementation by assuming the standard TPC-C benchmark as the workload and different network latencies. Thanks to this testbed, we show that most performance overheads of adaptively encrypted cloud databases are masked by network latency values that are quite typical of a cloud scenario. Other performance evaluations carried out in [10] assumed a LAN scenario and no network latency.

Moreover, 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.

• *University of Modena and Reggio Emilia. E-mail:* {*luca.ferretti,fabio.pierazzi,michele.colajanni,mirco.marchetti*}*@unimore.it*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2314644, IEEE Transactions on Cloud Computing

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 services. Section 4 proposes the analytical cost model for the estimation of database service costs in a medium horizon where it is likely that cloud prices and workload 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 and methodology applied to real cloud database providers over a three year 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 any SQL operation directly on the cloud. As a consequence, the tenant has two alternatives for any SQL operation: downloading the entire database, decrypting it, executing the query and, if the operation modifies the databases, encrypting and uploading the new data; decrypting temporarily the cloud database, executing the query, and re-encrypting it. The former solution is affected by huge communication and computation overheads, and costs that would make the 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, but avoiding that the provider obtains the decryption key. An initial solution in this direction was presented in [5]. This proposal is based on data aggregation techniques [8], that associate plaintext metadata to sets of encrypted data to allow data retrieval. 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 cloud data, but existing implementations are affected by huge computational costs [11] to the extent that they would make impractical the execution time of SQL operations over a cloud database. 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 prevents the scalability and availability that are among the most important features of any cloud utility service. 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 intermediary 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, the cost is a further concern of possible cloud tenant organizations. To address this issue, 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 tends to analyze the costs of cloud computing from a provider's perspective (e.g., [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 previous papers are mainly interested to 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 vs. database

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2314644, IEEE Transactions on Cloud Computing

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 methods for public cloud database service, where distributed and concurrent clients can issue direct SQL operations. By avoiding an architecture based on one [10] or multiple intermediate servers between the clients and the cloud database, the proposed solution guarantees the same level of scalability and availability of the cloud service. Figure 1 shows a scheme of the proposed architecture where each client executes an *encryption engine* that manages encryption operations. This software module is accessed by external *user applications* through the *encrypted database interface*. The proposed architecture manages five types of information.

- *plain data* is the tenant information;
- *encrypted data* is stored in the cloud database;
- *plain metadata* represent the additional information that is necessary to execute SQL operations on encrypted data;
- *encrypted metadata* is the encrypted version of the metadata that are stored in the cloud database;
- *master key* is the encryption key of the encrypted metadata that is distributed to legitimate clients.

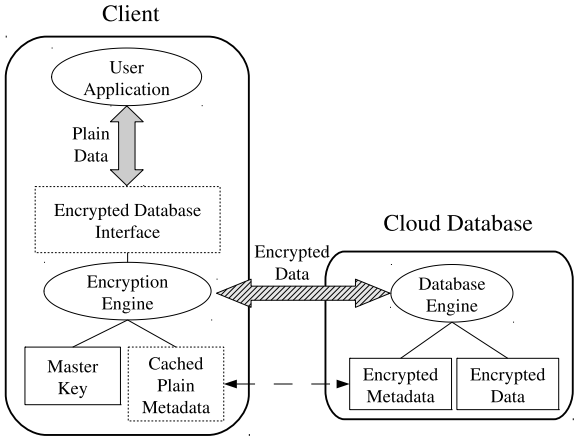

Fig. 1: Encrypted cloud database architecture

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 in plain format, 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 it through the master key. In order to improve performance, the plain metadata are cached locally by the client as a volatile information. After obtaining the metadata, the encryption engine is able to execute the SQL operation on encrypted data, 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 could be interested in accessing 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 server 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 (IND-CPA) [20], [21] because it does not reveal any information about the original plain value. 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* (Search): it supports equality check on full strings (i.e., the *LIKE* operator).
- *Plain*: it does not encrypt data; it is useful to support all SQL operators on non confidential data.

If each column of the database was encrypted through 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, consider a database supporting a web application for which feature or security updates are released. If the updates introduce new SQL operations that were not

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2314644, IEEE Transactions on Cloud Computing

considered at database design time, data encryption will prevent their execution. Moreover, a similar approach prevents data analytics on an encrypted database, since all the queries that were not considered at database design time cannot be executed. Finally, if the encrypted database is not subject to a well-defined workload (e.g. because it is queried by employees, rather than by applications) this encryption strategy cannot be applied. These issues can be addressed through *adaptive* encryption schemes, that support at runtime any SQL operation while preserving the maximum level of data confidentiality on the columns that are never involved in any operation. 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 allow less types of 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 support 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 its onions. For example, the plaintext values associated to Onion-Eq are encrypted through Det, then the Det value is encrypted through 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.

Figure 2 shows an example of the onions and layers structures. This figure considers two plaintext columns having data types *int* and *varstring*, respectively. The integer column is encrypted with Onion-Eq, Onion-Ord, and Onion-Sum, whereas the string column is associated to 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, then the adaptive layer removal mechanism removes the Rand layer of Onion-Eq, thus leaving Det as its new actual layer because it supports equality checks.

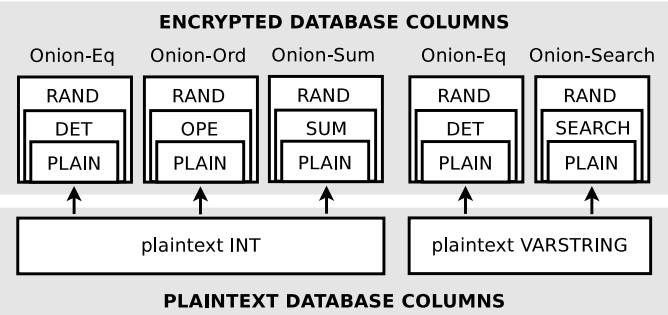

Fig. 2: Example of onion structures

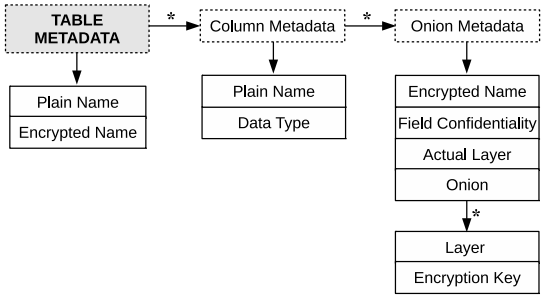

Fig. 3: Metadata structure

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

Table metadata includes the correspondence between the *plain table name* and the *encrypted table name* because each encrypted table name is randomly generated. Moreover, for each column of the original plain table it also includes a *column metadata* parameter containing the name and the data type of the corresponding plain column (e.g., integer, string, timestamp). Each column metadata is associated to one or more *onion metadata*, as many as the number of onions related to the column. The onion metadata is a complex data structure that must describe all the encryption information about an onion and its layers. The onion metadata contains the following attributes:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2314644, IEEE Transactions on Cloud Computing

- 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) that must correspond to one encryption layer available for that onion;
- 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); it must be one of the available onion types.

The onion metadata is associated with as many *encryption layer metadata* as the number of layers required by the onion type. Each layer metadata includes an encryption key and a label identifying the encryption algorithm that will be used. 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 describe the main operations involved in the encrypted database management: database creation, SQL commands execution, 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 (Section 3.2). 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. For example, by default integer columns are encrypted with Onion-Eq, Onion-Ord and Onion-Hom (Figure 2).

3.3.2 SQL commands execution

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 (e.g., $=$) are involved. The client issues a request for the table metadata (Section 3.2) 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 (Section 3.3.3) 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 guarantees the adaptivity of an encrypted cloud database by dynamically removing the external layer of an onion in order to guarantee at runtime support for SQL operations issued by legitimate clients.

We describe the details of the adaptive layer removal mechanism by referring to an example. Let us 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 to the column *id* because this is the only onion supporting the operator $<$. If the actual layer of Onion-Ord associated to *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 and execute the operation $(id < 10)$ 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 now Ope, and the cloud database does not re-encrypt the onion back to the upper layer (Rand).

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 remains 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.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2314644, IEEE Transactions on Cloud Computing

4 Cost estimation of cloud database services

We consider 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* [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 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 [28]. Billing costs are billed periodically to the tenant every *billing period* T_B. Moreover, if the tenant uses 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 Figure 4.

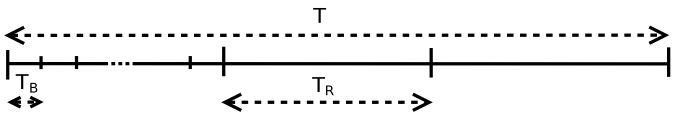

Fig. 4: Example of relationship among estimation (T), reservation (T_R) and billing (T_B) periods.

Pricing is the set of *reservation prices* and *billing prices*. Reservation prices $\{R_r\}$ refer to reservation periods $r = [1, \ldots, N_R]$, where $N_R = {}^{T}/_{T_R}$ 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]–[32]: *uptime price* $\{p_b^h\}$, *storage price* $\{p_b^s\}$ and *network price* $\{p_b^n\}$.

On the other hand, *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:

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

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 as x_b its usage at the *b-th* billing period and p_b^x its price. If the billing function is linear, it can be computed as:

$$\mathcal{L}(x_b, p_b^x) = p_b^x \cdot x_b \tag{3}$$

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 $[\hat{x}_1, \ldots, \hat{x}_{Z-1}]$ as the set of thresholds that define all the tiers. The price is modeled as a piecewise function:

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

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

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

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:

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2314644, IEEE Transactions on Cloud Computing

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 accurately the usage of storage (s_b) and network (n_b), 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 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 of the three configurations.

We assume that the tenant knows the database structure and the query workload and we assume 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:

$$s^p = \sum_{a \in A} (r_a \cdot v_a^p) \tag{7}$$

This equation considers only the storage usage of tenant data, and 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 the 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:

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

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 a single encryption algorithm. It is then possible to compute the encrypted size v_a^e of the attribute a encrypted through the algorithm ϕ as:

$$v_a^e = \mathcal{E}_\phi(v_a^p) \tag{9}$$

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.

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

TABLE 1: Functions $\mathcal{E}_\phi$ for different encryption algorithms

If the tenant does not make use of adaptive strategies, i.e. 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 replacing v_a^e to v_a^p 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 an 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:

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

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:

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

Since each plaintext column a may be associated to multiple onion-encrypted columns, we define Θ_a as a set of onions associated to 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:

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2314644, IEEE Transactions on Cloud Computing

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

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

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 to 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:

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

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

5 Performance Evaluation

This section aims to verify whether the overheads of adaptive encryption represent an acceptable compromise from the performance point of view for guaranteeing data confidentiality in 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 time 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 3GHz processor, 2GB of RAM and two 10,000 RPM 146GB SCSI disks. The server machine hosts a database server implemented in PostgreSQL 9.1 on a d710 machine having a quad-core Xeon 2.4 GHz processor, 12GB of RAM and a 7,200 RPM 500GB SATA disk. Each machine runs a Fedora 15 image.

The current version of the prototype supports the main SQL operations (SELECT, DELETE, INSERT and UPDATE) and the WHERE clause expressions. We consider three TPC-C compliant databases having ten warehouses and a scale factor of five.

- *Plaintext (PLAIN)* is based on plaintext data.
- *Encrypted (ENC)* refers to a statically encrypted database where each column is encrypted at design time through 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 two versions of encrypted databases, each column is set to the highest encryption layer required to support the respective 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). In order to guarantee data consistency, the three databases use repeatable read (snapshot) isolation level [35].

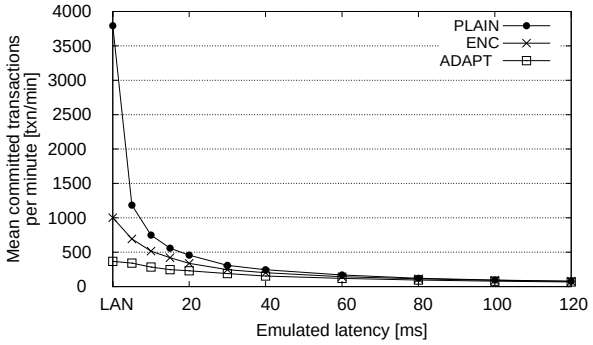

Fig. 5: TPC-C throughput with 5 clients

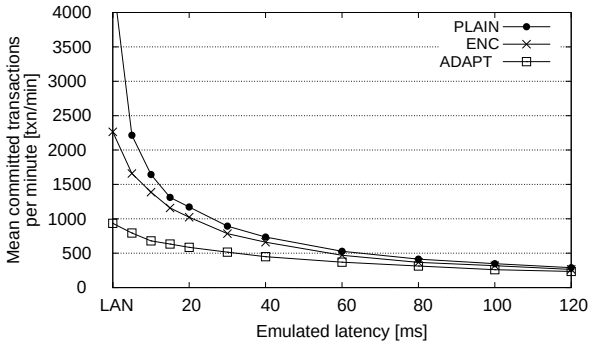

Fig. 6: TPC-C throughput with 20 clients

The experiments aim to evaluate the overhead caused by static and adaptive encryption in terms of system throughput and response time. In Figures 5 and 6, we report the number of committed TPC-C transactions per minute executed on the three versions of the cloud database service.

The results shown in Figures 5 and 6 are in funtion of increasing network latencies and refer to 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 systems, and it is quite close to them for latencies higher than 60ms, 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.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2314644, IEEE Transactions on Cloud Computing

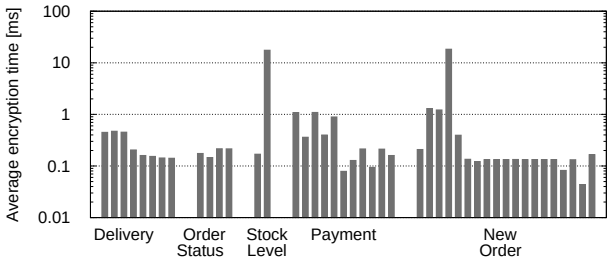

Fig. 7: Encryption times of the SQL operations composing the TPC-C workload (ENC configuration)

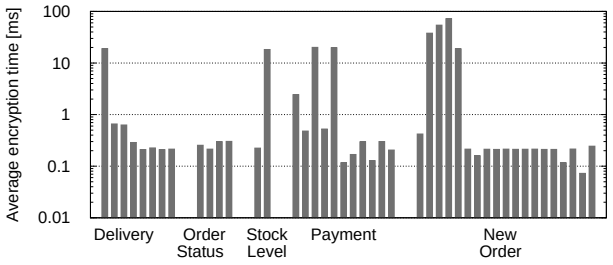

Fig. 8: Encryption times of the SQL operations composing the TPC-C workload (ADAPT configuration)

In Figures 7 and 8, we consider the ENC and ADAPT configurations and we report the mean 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 (Figure 8) some operations require much higher encryption times with respect to ENC (Figure 7). Further analyses demonstrate that the two peaks in ENC are related to SQL operations requiring Ope encryption, and that the majoriy of peaks in ADAPT are related to INSERT operations combined with the Ope encryption. This is due to the fact that the Ope algorithm requires an encryption time that is two or three orders of magnitude higher than Rand and Det algorithms [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 to the columns in which the TPC-C workload requires support for the order comparison operators.

We now investigate the impact of network latency on the SQL operations response time with regard to the overhead caused by static and adaptive encryption. We evaluate the response time of the most popular SELECT, DELETE, INSERT and UPDATE operations of the TPC-C workload for different numbers of clients. In Figures 9 and 10, we report as an example the response time overheads of the SELECT and INSERT operations for

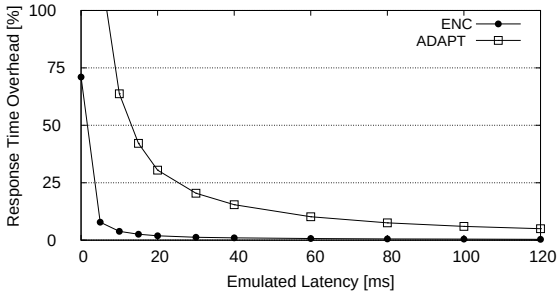

Fig. 9: SELECT rensponse time overhead - 10 clients

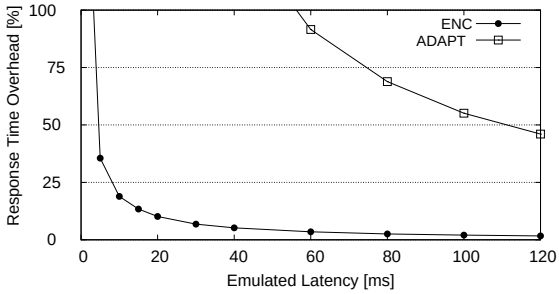

Fig. 10: INSERT response time overhead - 10 clients

10 active clients as a function of increasing network latencies. The overhead of the ENC and ADAPT configurations is measured with respect to the PLAIN database response time. In Figure 9, we observe that the overhead of the SELECT query tends to be masked for latencies higher than 60ms both in the ADAPT and the 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 Figure 10, the ADAPT configuration has response time overheads much higher than those related to the ENC configuration for any network latency. This result has a twofold motivation: the larger 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 motivated in the 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 standard benchmark. However, in the adaptively encrypted database 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2314644, IEEE Transactions on Cloud Computing

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 trade-off 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 trade-off 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 through 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 in the case of PLAIN, ENC and ADAPT configurations for real cloud database services. We initially validate the usage estimation methodology presented in Section 4.3. We then analyze the variations of costs for different cloud providers and resource usages. We finally evaluate tenant's costs over a mid-term period equal to three years by considering realistic resource usage increments and price reductions.

6.1 Validation of the usage estimation

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

First we validate the storage 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 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%.

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 2: Validation of storage overhead due to encryption of TPC-C compliant databases.

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

TABLE 3: Estimation of outgoing network due to database encryption over a TPC-C workload.

Now we validate the network estimation model. We deploy PLAIN, ENC and ADAPT TPC-C compliant databases of 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 measured the network usage of the PLAIN database, and we obtain an average of 7162 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 estimations 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].

The left part of Table 4 reports the unit storage prices p^s, the unit network prices p^n, the total uptime cost $\mathcal{H}$, and the annual reservation price R reported as monthly cost. 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2314644, IEEE Transactions on Cloud Computing

Name	Pricing		Fixed costs		Result	PLAIN	ENC	ADAPT
	p^s	p^n	$\mathcal{R}/12$	$\mathcal{H}$				
Amazon RDS Small	0.10 $/GB	0.12 $/GB	14.08 $	18.25 $	C	54.33 $	73.36 $	83.41 $
					$\mathcal{S}$ ($\mathcal{S}/C$)	10.12 $ (18.6%)	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 $
					$\mathcal{S}$ ($\mathcal{S}/C$)	10.12 $ (7.0%)	18.99 $ (11.6%)	28.14 $ (16.2%)
					$\mathcal{N}$ ($\mathcal{N}/C$)	12.00 $ (8.3%)	22.03 $ (6.2%)	22.93 $ (6.3%)
Amazon RDS HM:2XL	0.10 $/GB	0.12 $/GB	131.50 $	181.04 $	C	334.54 $	353.56 $	363.61 $
					$\mathcal{S}$ ($\mathcal{S}/C$)	10.12 $ (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 $
					$\mathcal{S}$ ($\mathcal{S}/C$)	7.60 $ (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	695.25 $	706.93 $	714.51 $
					$\mathcal{S}$ ($\mathcal{S}/C$)	7.60 $ (1.1%)	14.24 $ (2.0%)	21.11 $ (3.0%)
					$\mathcal{N}$ ($\mathcal{N}/C$)	9.50 $ (1.1%)	17.44 $ (2.5%)	18.15 $ (2.5%)

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.

cost $\mathcal{H}$ may vary depending on the chosen instance (e.g., [28], [31]).

The right part of Table 4 reports the estimation of the billing costs. For each instance, we estimate the monthly cost C of the PLAIN, ENC and ADAPT configurations, expressed in $ units, 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 which is the impact of static and adaptive database encryption on the monthly billing costs for different instance classes. This table confirms that the influence of $\mathcal{S}$ and $\mathcal{N}$ on C is mainly due to increments in storage and network usages.

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, this means that the cost overhead due to encryption configurations 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 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 $ units, and the estimated monthly *cost percentage increase* $C_\%$ (i.e., the cost overhead) 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% 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 1000 GB, the value $C_\%$ 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. Our 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 a realistic application of the proposed cost model in which a tenant wants to estimate the costs of moving its e-commerce database to an Amazon RDS Large instance [23] for the upcoming *three years*.

We assume that the tenant initial database size is 200 GB, the initial month is March 2014, and that the workload in terms of committed transactions per day is characterized by the trend shown in Figure 11. The average network usage determined by this workload trend is approximately 2 GB/day, with a peak of about

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2314644, IEEE Transactions on Cloud Computing

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	123.35	127.35	132.35	172.35	222.35	334.70	337.70	341.45	371.45	408.95
	50 GB	128.15	132.15	137.15	177.15	227.15	338.50	341.50	345.25	375.25	412.75
	100 GB	134.15	138.15	143.15	183.15	233.15	343.25	346.25	350.00	380.00	417.50
	500 GB	182.15	186.15	191.15	231.15	281.15	381.25	384.25	388.00	418.00	455.50
	1000 GB	242.15	246.15	251.15	291.15	341.15	428.75	431.75	435.50	465.50	503.00
ENC $C_{\%}$[%]	10 GB	1.5%	4.3%	7.6%	26.7%	40.9%	0.4%	1.2%	2.2%	9.3%	16.7%
	50 GB	4.6%	7.2%	10.2%	28.2%	41.8%	1.4%	2.2%	3.1%	10.0%	17.3%
	100 GB	8.1%	10.5%	13.3%	30.0%	42.9%	2.5%	3.3%	4.2%	11.0%	18.1%
	500 GB	28.0%	29.4%	30.9%	41.1%	49.8%	10.6%	11.2%	12.0%	17.6%	23.5%
	1000 GB	41.8%	42.6%	43.5%	49.9%	55.8%	18.7%	19.2%	19.8%	24.3%	29.2%
$ADAPT$ $C_{\%}$[%]	10 GB	2.4%	8.0%	14.5%	53.3%	82.1%	0.7%	2.3%	4.2%	18.5%	33.5%
	50 GB	5.7%	11.0%	17.2%	54.3%	82.3%	1.7%	3.3%	5.2%	19.3%	34.0%
	100 GB	9.5%	14.5%	20.3%	55.5%	82.5%	2.9%	4.5%	6.4%	20.2%	34.7%
	500 GB	31.0%	34.2%	38.1%	62.9%	84.0%	11.7%	13.0%	14.7%	26.6%	39.4%
	1000 GB	45.9%	48.1%	50.8%	68.7%	85.2%	20.5%	21.6%	23.0%	33.2%	44.3%

TABLE 5: Monthly costs increases for different combinations of plaintext storage and network usages

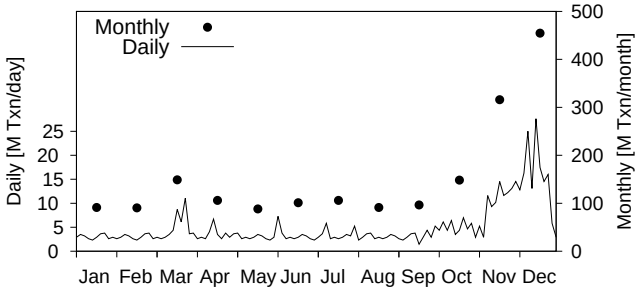

Fig. 11: Workload trend in terms of committed transactions per day (and month) during a year for a typical e-commerce workload.

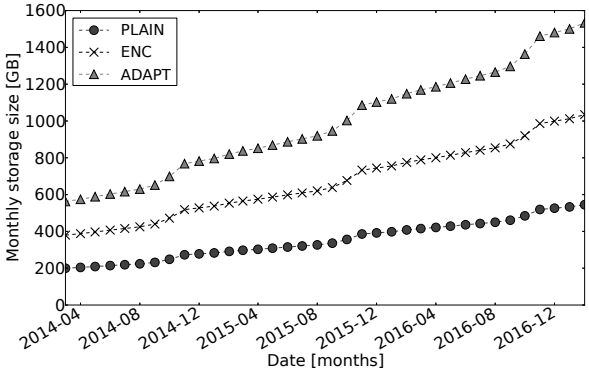

Fig. 12: Expected storage usage in the DYNAMIC scenario for the upcoming three years (from March 2014 to February 2017).

15 GB/day during the holiday season after Thanksgiving.

We consider two scenarios: STATIC and DYNAMIC. The former assumes that during the whole 3-years period the storage size of the database remains equal to 200 GB, the network usage depends on the trend of Figure 11, and the cloud prices do not vary over time. In this scenario, the cloud prices are related to an Amazon RDS Large instance evaluated at the end of February 2014, as in Table 4.

The DYNAMIC scenario considers that, over a three years horizon, the storage and network usages and the cloud prices change. In particular, its workload intensity varies over time due to increasing number of customers: we assume a *monthly workload increase* equal to 2% with respect to the trend shown in Figure 11. The TPC-C standard [36] estimates that there is a relation between the number of committed transactions and the storage growth. According to this relation and the workload of the DYNAMIC scenario, Figure 12 reports the expected storage size usage for the upcoming three years for the PLAIN, ENC and ADAPT databases. We observe that the storage usage of the ENC and ADAPT configurations is higher than that of the PLAIN configuration, and that their difference tends to increase over time.

Moreover, the DYNAMIC scenario considers that the cloud prices tend to decrease [27]. We can assume that the Amazon RDS price reductions of the past three years [27] repeat their trend, and we represent the expected prices for the upcoming three years in Figure 13, where the Y-axis ranges from $0.06 to $0.13. We observe that the storage (p_b^s) and network (p_b^n) prices decrease only once during the whole 3-years period, whereas the uptime price (p_b^h) decreases more frequently.

In Figure 14 we report the monthly billing costs of PLAIN, ENC and ADAPT databases for the DYNAMIC scenario. We can observe that, despite storage and network usage increases, the monthly costs remain approximately stable. As expected, the monthly costs of the ENC and ADAPT configurations are higher because their storage and network usages always overcome the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2314644, IEEE Transactions on Cloud Computing

	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

TABLE 6: Overall costs of the cloud database service during the 3-years period in STATIC and DYNAMIC scenarios.

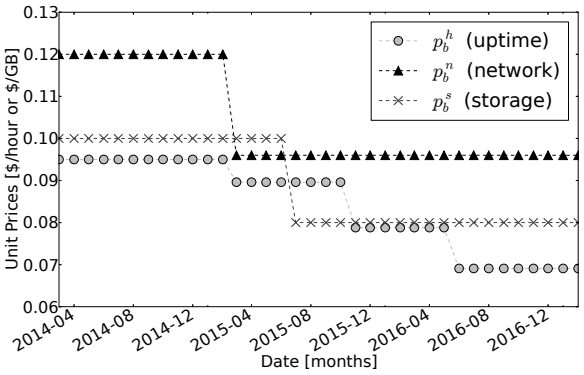

Fig. 13: Expected price reductions for Amazon RDS (from March 2014 to February 2017).

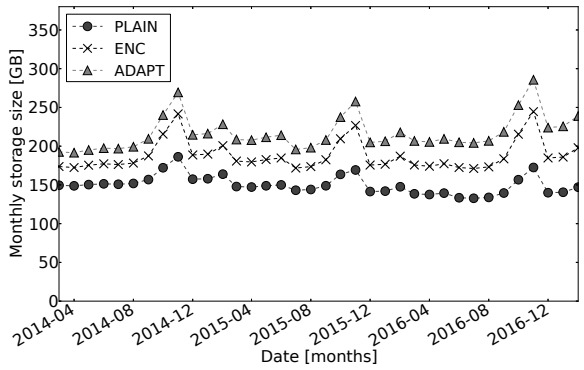

Fig. 14: Billing costs of the DYNAMIC scenario in the 3-year period (from March 2014 to February 2017).

PLAIN usages.

The previous analysis shows that the costs trend depends on two opposing factors, because costs both increase due to usage growth and decrease due to cloud price reductions. We propose a *break even point* estimation, that is the value of monthly workload intensity increase for which price reductions balance the increasing costs deriving from usage growth.

An analysis of the break even points for the considered scenario is presented in Table 6, in which we report the annual and triennial costs in the STATIC and DYNAMIC configurations, with respect to the PLAIN, ENC and ADAPT databases. In particular, we consider three DYNAMIC configurations characterized by different monthly workload increases: 2%, 4% and 9%, that are the break even points of the PLAIN, ENC and ADAPT databases respectively.

By analyzing the DYNAMIC scenario with the 2% monthly workload increase, we can observe that the annual costs decrease in the PLAIN and ENC configurations during the three years, whereas the annual costs remain approximately stable for the ADAPT configuration. This is due to the price reductions (Figure 13) that balance the storage and network usage increases in the PLAIN and ENC configurations, whereas in the ADAPT one the annual costs remains approximately constant because adaptive encryption determines higher storage and network usages.

From Table 6 we can also observe that the STATIC scenario always underestimates the triennial costs of the ENC and ADAPT configurations with respect to the DYNAMIC scenarios. The difference between the triennial costs of the STATIC and DYNAMIC scenarios increases for higher monthly workload increases.

If the tenant is able to estimate the monthly workload increase for his cloud database, he can estimate the trend of the annual costs during the 3-year period for the PLAIN, ENC and ADAPT configurations by computing the break even points.

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2314644, IEEE Transactions on Cloud Computing

workload and cloud price variations in a mid-term horizon. By instantiating the model with actual cloud provider prices, we can determine the encryption and adaptive encryption cost of data confidentiality. From the research point of view, it would be also interesting to evaluate the proposed or alternative architectures 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 Computer Systems*, 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*. O'Reilly Media, Incorporated, 2009.

[3] H.-L. Truong and S. Dustdar, "Composable cost estimation and monitoring for computational applications in cloud computing environments," *Procedia Computer Science*, vol. 1, no. 1, pp. 2175 – 2184, 2010, iCCS 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. 2008 ACM/IEEE Conf. Supercomputing*, ser. SC '08. Piscataway, NJ, USA: IEEE Press, 2008, pp. 50:1–50:12.

[5] H. Hacigümüş, B. Iyer, and S. Mehrotra, "Providing database as a service," in *Proc. 18th IEEE Int'l Conf. Data Engineering*, Feb. 2002.

[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. Computer and communications security*. ACM, 2010, pp. 735–737.

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

[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. Management of data*, June 2002.

[9] L. Ferretti, M. Colajanni, and M. Marchetti, "Distributed, concurrent, and independent access to encrypted cloud databases," *IEEE Trans. Parallel and Distributed Systems*, vol. 25, no. 2, 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.

[11] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st ACM Symp. Theory of computing*, May 2009.

[12] A. Boldyreva, N. Chenette, and A. O'Neill, "Order-preserving encryption revisited: Improved security analysis and alternative solutions," in *Proc. Advances in Cryptology – CRYPTO 2011*. Springer, Aug. 2011.

[13] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Advances in Cryptology – EUROCRYPT99*. Springer, May 1999.

[14] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symposium on Security and Privacy.*, May 2000.

[15] L. Ferretti, F. Pierazzi, M. Colajanni, and M. Marchetti, "Security and confidentiality solutions for public cloud database services," in *Proc. Seventh Int'l Conf. Emerging Security Information, Systems and Technologies*, Aug. 2013.

[16] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *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'l Conf. Emerging Networking Experiments and Technologies*, 2010.

[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 university press, 2004.

[20] J. Daemen and V. Rijmen, *The design of Rijndael: AES – the advanced encryption standard*. Springer, 2002.

[21] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (blowfish)," in *Proc. Cambridge Security Work. Fast Software Encryption*, Dec. 1993.

[22] L. Ferretti, M. Colajanni, and M. Marchetti, "Supporting security and consistency for cloud database," in *Proc. Fourth Int'l Symp. Cyberspace Safety and Security*. Springer, Dec. 2012.

[23] Amazon Web Services, "Amazon Relational Database Service," http://aws.amazon.com/rds, Mar. 2014.

[24] EnterpriseDB, "Postgres Plus Cloud Database," http://enterprisedb.com/cloud-database, Mar. 2014.

[25] Microsoft, "Windows Azure SQL Database," http://www.windowsazure.com/en-us/services/data-management, Mar. 2014.

[26] Rackspace, "Rackspace Cloud Database," http://www.rackspace.com/cloud/databases, Mar. 2014.

[27] Amazon, "Amazon Web Services Blog," http://aws.typepad.com/aws/price-reduction/, Mar. 2014.

[28] Amazon RDS Pricing, "Amazon Relational Database Pricing," http://aws.amazon.com/rds/pricing, Mar. 2014.

[29] EnterpriseDB, "Postgres Plus Cloud Database Pricing," http://www.enterprisedb.com/cloud-database/pricing-amazon, Mar. 2014.

[30] Microsoft, "Windows Azure SQL Database Web & Business Pricing," http://www.windowsazure.com/en-us/pricing/details/sql-database/#service-webandbusiness, Mar. 2014.

[31] ——, "Windows Azure SQL Database Premium Pricing," http://www.windowsazure.com/en-us/pricing/details/sql-database/#service-premium, Mar. 2014.

[32] Rackspace, "Rackspace Cloud Database Pricing," http://www.rackspace.com/cloud/databases/pricing, Mar. 2014.

[33] L. Ferretti, M. Colajanni, and M. Marchetti, "Access control enforcement of query-aware encrypted cloud databases," in *Proc. Fifth IEEE Int'l Conf. Cloud Computing Technology and Science*, Dec. 2013.

[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. Fifth USENIX Conf. Operating Systems Design and Implementation*, Dec. 2002.

[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, 2005.

[36] TPC, "Tpc-c on-line transaction processing benchmark." http://www.tpc.org/tpcc, Mar. 2014.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2014.2314644, IEEE Transactions on Cloud Computing

Luca Ferretti is a Ph.D. student at the International Doctorate School in Information and Communication Technologies (ICT) of the University of Modena and Reggio Emilia, Italy. He received the Master Degree in computer engineering from the same University in 2012. His research focuses on information security, and cloud architectures and services. Home page: http://weblab.ing.unimo.it/people/ferretti

Fabio Pierazzi is a Ph.D. student at the International Doctorate School in Information and Communication Technologies (ICT) of the University of Modena and Reggio Emilia, Italy. He received the Master Degree in Computer Engineering from the same University in 2013. His research interests include security analytics and performance evaluation of cloud services. Home page: http://weblab.ing.unimo.it/people/fpierazzi

Michele Colajanni is full professor in computer engineering at the University of Modena and Reggio Emilia since 2000. He received the Master degree in computer science from the University of Pisa, and the Ph.D. degree in computer engineering from the University of Roma in 1992. 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. Home page: http://weblab.ing.unimo.it/people/colajanni

Mirco Marchetti received his Ph.D. in Information and Communication Technologies (ICT) in 2009. He holds a post-doc position at the Interdepartment Center for Research on Security and Safety (CRIS) of the University of Modena and Reggio Emilia. He is interested in intrusion detection, cloud security and in all aspects of information security. Home page: http://weblab.ing.unimo.it/people/marchetti