**Implementation Modules:**

1. **Adaptive encryption**
2. **Metadata structure**
3. **Encrypted database management**
4. **Cost Estimation of cloud database services**
5. **Cost model**
6. **Cloud pricing models**
7. **Usage Estimation**

**Adaptive encryption:**

The proposed system supports adaptive encryption methods for public cloud database service, where distributed and concurrent clients can issue direct SQL

operations. By avoiding an architecture based on one [or] multiple intermediate servers between the clients and the cloud database, the proposed solution guarantees the same level of scalability and availability of the cloud service. 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.

**Metadata structure:**

Metadata include all information that allows a legitimate client knowing the master key to execute SQL operations over an encrypted database. They are organized and

stored at a table-level granularity to reduce communication overhead for retrieval, and to improve management of concurrent SQL operations. We define all metadata

information associated to a table as *table metadata*. Let us describe the structure of a table metadata .Table metadata includes the correspondence between the *plain table name* and the *encrypted table name* because each encrypted table name is randomly generated. Moreover, for each column of the original plain table

it also includes a *column metadata* parameter containing the name and the data type of the corresponding plain column (e.g., integer, string, timestamp). Each column metadata is associated to one or more *onion metadata*, as many as the number of onions related to the column.

**Encrypted database management:**

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

**Cost Estimation of cloud database services:**

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

**Cost model:**

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

Cost = f(T ime, Pricing,Usage) where:

• *Time*: identifies the time interval T for which the tenant requires the service.

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

• *Usage*: denotes the total amount of resources used by the tenant; it typically increases during T .In order to detail the *pricing* attribute, it is important

to specify that cloud providers adopt two subscription

policies: the *on-demand* policy allows a tenant to payper-use and to withdraw its subscription anytime; the *reservation* policy requires the tenant to commit in advance for a *reservation period*. Hence, we distinguish between *billing costs* depending on resource usage and *reservation costs* denoting additional fees for commitment in exchange for lower pay-per-use prices. Billing costs are billed periodically to the tenant every *billing* *period.*

**Cloud pricing models:**

Popular cloud database providers adopt two different billing functions, that we call *linear* L and *tiered* T . Let us consider a generic resource x, we define as xb its usage at the *b-th* billing period and px b its price. If the billing function is tiered, the cloud provider uses different prices for different ranges of resource usage. Let

us define Z as the number of tiers, and [ˆx1, . . . , ˆxZ−1] as the set of thresholds that define all the tiers. The uptime and the storage billing functions of *Amazon RDS* are linear, while the network usage is a tiered billing function. On the other hand, the uptime billing functions of *Azure SQL* is linear, while the

storage and network billing functions are tiered.

**Usage Estimation:**

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