ISSN:2320-0790

An International Journal of Advanced Computer Technology

PERFORMANCE AND COST EVALUATION OF AN ADAPTIVE ENCRYPTION ARCHITECTURE FOR CLOUD DATABASES

P.Raviteja[1], M.Uday Kumar[2], M. Parimala[3], Dr.G.Manoj Someswar[4]

[1]M.Tech. from Tirumala Engineering College, Hyderabad, Telangana, India
[2] M.Tech., Assistant Professor, Dept.of CSE, Tirumala Engineering College, Hyderabad, Telangana, India
[3]M.Tech., Associate Professor, Dept.of CSE, Tirumala Engineering College, Hyderabad, Telangana, India
[4]B.Tech., M.S.(USA), Ph.D., Director General & Scientist 'G', Global Research Academy, Hyderabad, Telangana, India

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

Keywords: Database as a Service(DBaaS), adaptive encryption scheme, Top Down Integration, Bottom Up Integration, Adaptive encryption, Metadata structure, Encrypted database management

I. INTRODUCTION

Although this adaptive encryption architecture is attractive because it does not require defining 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 system, 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. 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 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.

System Design

Figure 1: Class Diagram

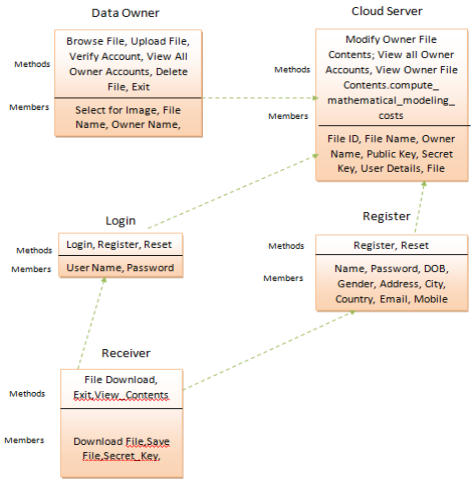
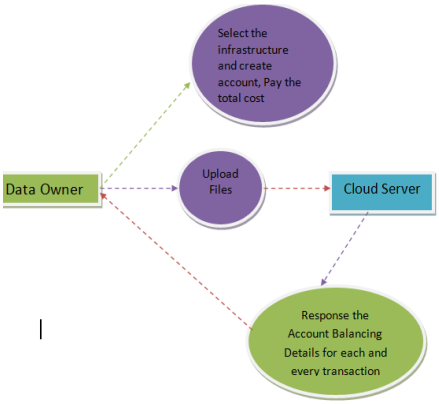


Figure:2

Data Flow Diagram

Level:0



Level: 1

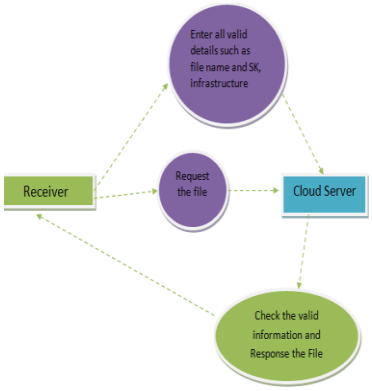


Figure 3: Sequence Diagram

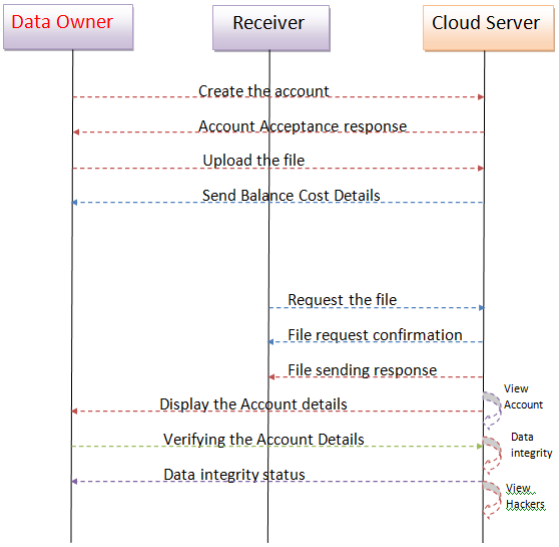
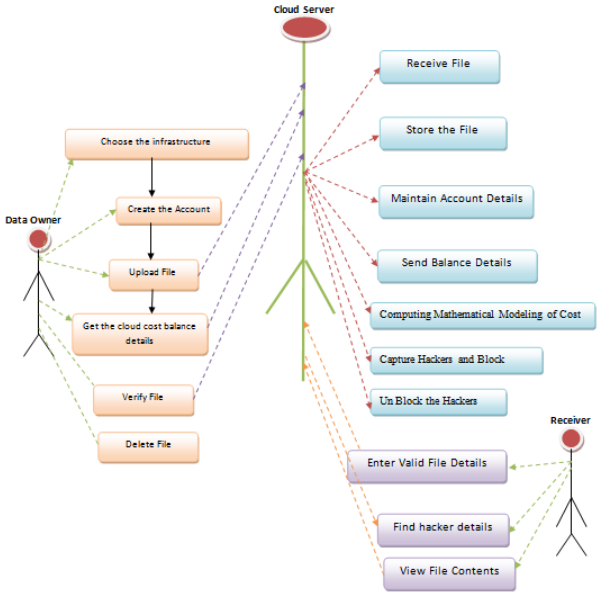


Figure 4: Use Case Diagram



PRELIMINARY INVESTIGATION

The first and foremost strategy for development of a project starts from the thought of designing a mail enabled platform for a small firm in which it is easy and convenient of sending and receiving messages, there is a search engine ,address book and also including some entertaining games. When it is approved by the organization and our project guide the first activity, ie. preliminary investigation begins. The activity has three parts:
Request Clarification
Feasibility Study
Request Approval

REQUEST CLARIFICATION

After the approval of the request to the organization and project guide, with an investigation being considered, the project request must be examined to determine precisely what the system requires.

Here our project is basically meant for users within the company whose systems can be interconnected by the Local Area Network(LAN). In today's busy schedule man need everything should be provided in a readymade manner. So taking into consideration of the vastly use of the net in day to day life, the corresponding development of the portal came into existence.

FEASIBILITY ANALYSIS

An important outcome of preliminary investigation is the determination that the system request is feasible. This is possible only if it is feasible within limited resource and time. The different feasibilities that have to be analyzed are:

Operational Feasibility
Economic Feasibility
Technical Feasibility
Operational Feasibility

Operational Feasibility deals with the study of prospects of the system to be developed. This system operationally eliminates all the tensions of the Admin and helps him in effectively tracking the project progress.[1] This kind of automation will surely reduce the time and energy, which previously consumed in manual work. Based on the study, the system is proved to be operationally feasible.

Economic Feasibility

Economic Feasibility or Cost-benefit is an assessment of the economic justification for a computer based project. As hardware was installed from the beginning & for lots of purposes thus the cost on project of hardware is low. Since the system is a network based, any number of employees connected to the LAN within that organization can use this tool from at anytime. The Virtual Private Network is to be developed using the existing resources of the organization. So the project is economically feasible.[2]

Technical Feasibility

According to Roger S. Pressman, Technical Feasibility is the assessment of the technical resources of the organization. The organization needs IBM compatible machines with a graphical web browser connected to the Internet and Intranet. The system is developed for platform Independent environment. Java Server Pages, JavaScript, HTML, SQL server and WebLogic Server are used to develop the system. The technical feasibility has been carried out. The system is technically feasible for development and can be developed with the existing facility.[3]

REQUEST APPROVAL

Not all request projects are desirable or feasible. Some organization receives so many project requests from client users that only few of them are pursued. However, those projects that are both feasible and desirable should be put into schedule.[4] After a project request is approved, it cost, priority, completion time and personnel requirement is estimated and used to determine where to add it to any project list. Truly speaking, the approval of those above factors, development works can be launched.

SYSTEM DESIGN AND DEVELOPMENT

INPUT DESIGN

Input Design plays a vital role in the life cycle of software development, it requires very careful attention of developers. The input design is to feed data to the application as accurate as possible. So inputs are supposed to be designed effectively so that the errors occurring while feeding are minimized. According to Software Engineering Concepts, the input forms or screens are designed to provide to have a validation control over the input limit, range and other related validations.

This system has input screens in almost all the modules. Error messages are developed to alert the user whenever he commits some mistakes and guides him in the right way so that invalid entries are not made. Let us see deeply about this under module design.

Input design is the process of converting the user created input into a computer-based format. The goal of the input design is to make the data entry logical and free from errors. The error is in the input are controlled by the input design. The application has been developed in user-friendly manner. The forms have been designed in such a way during the processing the cursor is placed in the position where must be entered. The user is also provided with in an option to select an appropriate input from various alternatives related to the field in certain cases.

Validations are required for each data entered. Whenever a user enters an erroneous data, error message is displayed and the user can move on to the subsequent pages after completing all the entries in the current page.

OUTPUT DESIGN

The Output from the computer is required to mainly create an efficient method of communication within the company primarily among the project leader and his team members, in other words, the administrator and the clients. The output of VPN is the system which allows the project leader to manage his clients in terms of creating new clients and assigning new projects to them, maintaining a record of the project validity and providing folder level access to each client on the user side depending on the projects allotted to him. After completion of a project, a new project may be assigned to the client. User authentication procedures are maintained at the initial stages itself. A new user may be created by the administrator himself or a user can himself register as a new user but the task of assigning projects and validating a new user rests with the administrator only.

The application starts running when it is executed for the first time. The server has to be started and then the internet explorer in used as the browser. The project will run on the local area network so the server machine will serve as the

administrator while the other connected systems can act as the clients. The developed system is highly user friendly and can be easily understood by anyone using it even for the first time.

SYSTEM TESTING

TESTING METHODOLOGIES

The following are the Testing Methodologies:
Unit Testing.
Integration Testing.
User Acceptance Testing.
Output Testing.
Validation Testing.

Unit Testing

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. [5]This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

Integration Testing

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.[6]

The following are the types of Integration Testing:

1.Top Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.[7]

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

2. Bottom-up Integration

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

The low-level modules are combined into clusters into clusters that perform a specific Software sub-function. A driver (i.e.) the control program for testing is written to coordinate test case input and output. The cluster is tested. Drivers are removed and clusters are combined moving upward in the program structure. The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality.[8]

User Acceptance Testing

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. [9]The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format.[10] Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

Validation Checking

Validation checks are performed on the following fields.

Text Field

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.[11]

Numeric Field

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. [12] Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

Preparation of Test Data

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data.[13] While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

Using Live Test Data

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the

systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.[14]

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

Using Artificial Test Data

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.[15]

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package "Virtual Private Network" has satisfied all the requirements specified as per software requirement specification and was accepted.

USER TRAINING

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed.[16] For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

MAINTENANCE

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user's requirements during the process of system development. [17]Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

TESTING STRATEGY

A strategy for system testing integrates system test cases and design techniques into a well planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation .[18] A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

SYSTEM TESTING

Software once validated must be combined with other system elements (e.g. Hardware, people, and database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.[19]

UNIT TESTING

In unit testing different are modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. [20] Using the detailed design description as a guide, important Conrail paths are tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module. [21]

In due course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this. The future holds a lot to offer to the development and refinement of this project.

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.

RESULTS & CONCLUSION

There are two main tenant concerns that may prevent the adoption of the cloud as the fifth utility: data confidentiality and costs. This system 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 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 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. Hacig¨um¨us¸, 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. Hacig¨um¨us¸, 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.