Australian Journal of Basic and Applied Sciences, 9(15) Special 2015, Pages: 100-106

ISSN:1991-8178

Australian Journal of Basic and Applied Sciences

Journal home page: www.ajbasweb.com

Providing Data Confidentiality on EHM Using Adaptive Encryption Scheme in Cloud Database

N. Hemalatha and R. Nandhini

Agni college of Technology, Anna University, Computer Science and Engineering, A. Thiyagarajan, Chennai. India

ARTICLE INFO

Article history:
Received 10 March 2015
Received in revised from 20 March
Accepted 25 March 2015
Available online 10 April 2015

Keywords:
cloud database, cryptography, confidentiality, encryption.

ABSTRACT

For proving secure performance improvement and the cost reduction in cloud usage, we propose an architecture that takes advantage of adaptive encryption mechanisms to guarantee at runtime the best level of data confidentiality for any type of SQL operation. The users perception that the confidentiality of their data is endangered by internal and external attacks is limiting the usage of the public cloud database service. The use of cryptography is complicated by high computational costs and restrictions on supported SQL operations over encrypted data. The cost of the cloud usage will varies and its depends on time and resource usage ,if a project or a system reduces the time and resource usage there will be a decrease in cost therefore, users to such system will automatically increases.

© 2015 AENSI Publisher All rights reserved.

To Cite This Article: N. Hemalatha1 and R. Nandhini, Providing Data Confidentiality on EHM Using Adaptive Encryption Scheme in Cloud Database. ***Aust. J. Basic & Appl. Sci.,*** *9(15): 100-106, 2015*

INTRODUCTION

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 that allow the execution of SQL operations over encrypted data either have performance limits or require the choice of which encryption scheme must be adopted for each database column and SQL operation. These latter proposals are fine when the set of queries can be statically determined at design time, while we are interested in other common scenarios where the workload may change after the database design.

In this paper, we propose a novel architecture for adaptive encryption of public cloud databases that offers a proxy-free alternative to the system described in. The proposed architecture guarantees in an adaptive way the best level of data confidentiality for any database workload, even when the set of SQL queries dynamically changes. The adaptive encryption scheme, which was initially proposed for applications not referring to the cloud, encrypts each plain column to multiple encrypted columns, and each value is encapsulated in different layers of encryption, so that the outer layers guarantee higher confidentiality but support fewer computation capabilities with respect to the inner layers. The outer layers are dynamically adapted at runtime when new SQL operations are added to the workload.

Then, we evaluate the performance of encrypted database services by assuming the standard TPC-C benchmark as the workload and by considering different network latencies. Thanks to this test bed, we show that most performance overheads of adaptively encrypted cloud databases are masked by network latencies that are typical of a geographically distributed cloud scenario For example, an encryption algorithm may support the order comparison command but not a search operator. The drawback related to these feasible encryption algorithms is that in a medium-long term horizon, the database administrator cannot know at design time which database operations will be required over each database column. This issue is in part addressed in by proposing an adaptive encryption architecture that is founded on an intermediate and trusted proxy.

Related works:

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.

Corresponding Author: N. Hemalatha, Agni college of Technology, Anna University, Computer Science and Engineering, A. Thiyagarajan, Chennai. India

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 naive solutions encrypt the whole database through some standard encryption algorithms that do not allow to execute any SQL operation directly on the cloud. As a consequence, the tenant has two alternatives: download the entire database, decrypt it, execute the query and, if the operation modifies the database, encrypt and upload the new data; decrypt temporarily the cloud database, execute the query, and re-encrypt it. Secure DBaaS provides several original features that differentiate it from previous work in the field of security for remote database services Secure DBaaS relates more closely to works using encryption to protect data managed by untrusted databases. In such case, a main issue to address is that cryptographic techniques cannot be naively applied to standard DBaaS because DBMS can only execute SQL Operations over plaintext data.

Architecture design:

The proposed system supports adaptive encryption for public cloud database services, where distributed and concurrent

Clients can issue direct SQL operations. By avoiding an architecture based on intermediate servers between the clients and the cloud database, the proposed solution guarantees the same level of scalability and availability of the cloud service. Fig. 1 shows a scheme of the proposed architecture where each client executes an encryption engine that manages encryption operations. This software module is accessed by external user applications through the encrypted database interface. The proposed architecture manages five types of information.

- plain data: the informative content provided by the client users.
- encrypted data: the encrypted data that are stored in the cloud database.
- plain metadata: all the information required by the clients to manage encrypted data on the cloud database.
- encrypted metadata: the encrypted metadata that are stored in the cloud database.
- master key: the encryption key of the encrypted metadata. We assume that it is distributed to all legitimate clients.

The proposed system supports adaptive encryption for public cloud database services, where distributed and concurrent clients can issue direct SQL operations. By avoiding an architecture based on intermediate servers between the clients and the cloud database, the proposed solution guarantees the same level of scalability and availability of the cloud service. Fig. 1 shows a scheme of the proposed architecture where each client executes an encryption engine that manages encryption operations.

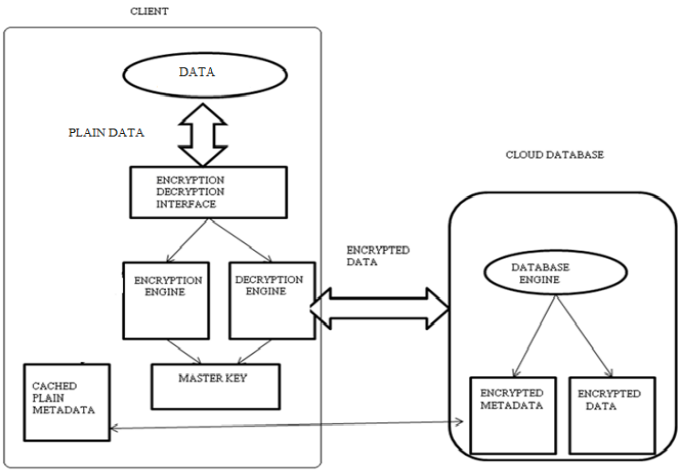


Fig. 1: Encrypted and Decrypted cloud database architecture.

This software module is accessed by external user applications through the encrypted database interface. The proposed architecture manages five types of information. 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 engines are not encrypted, whereas information is always encrypted before sending it to the cloud database. When an application issues a new SQL operation, the encrypted database interface contacts the encryption engine that retrieves the encrypted metadata and decrypts them with the master key. To improve performance, the plain metadata are cached locally by the client. After obtaining the metadata, the encryption engine is able to issue encrypted SQL statements to the cloud

database, and then to decrypt the results. The results are returned to the user application through the encrypted database interface.

3.1 Adaptive Encryption Techniques:

We consider SQL-aware encryption algorithms that guarantee data confidentiality and allow the cloud database server to carry out a large set of SQL operations over encrypted data. Each algorithm supports a specific subset of SQL operators.

This paper refers to the following encryption schemes.

Deterministic (Det): it deterministically encrypts data, so that the encryption of an input value always guarantees the same output value. It supports the equality operator.

Orders Preserving Encryption (OPE): this encryption scheme preserves in the encrypted values the numerical order of the original unencrypted data. It supports the following SQL operators: equal, unequal, less, less or equal, greater, greater or equal.

Sum: this encryption algorithm is homomorphic with respect to the sum operation: summing unencrypted data is equivalent to multiplying the correspondent encrypted values. It supports the sum operator between integer values.

Search: it supports equality check on full strings (i.e., the LIKE operator) that do not include fragments of words.

Random (Rand): it is a semantic secure encryption (INDCPA) that does not reveal any information of the original plain value. It does not support any SQL operator.

Plain: a special kind of "encryption" that leaves values unencrypted. It supports all SQL operators, and it is included to store publicly available data, or some anonymous values that do not require any data confidentiality.

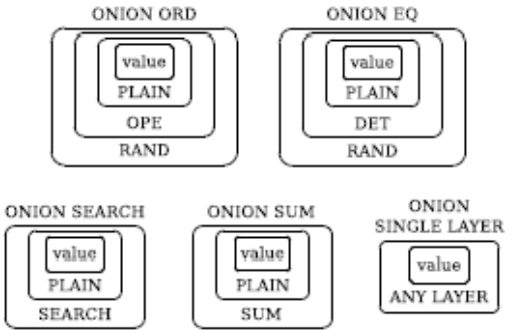


Fig. 2: Onions and layers structure.

In our architecture, each plain database column is encrypted into one or more encrypted columns, each one corresponding to a different Onion, depending on the SQL operations that must be supported on that column. The most external Encryption Layer of an Onion is called Actual Layer, which by default corresponds to its strongest encryption algorithm.

Each data type is characterized by a default set of supported Onions, depending on the operations supported by the data type and the compatibility between the encryption algorithms and the data type itself. Each database column can be defined through three parameters: column name, data type, and confidentiality parameters. The confidentiality parameters of a column define the set of Onions to be associated with it, and their starting Actual Layers. The Onions associated to a column must be compatible with the column data type. For example, integer columns can be associated to Onion-Eq, Onion-Ord and Onion-Sum, because integer values support equality checks, order comparisons and sums, but they cannot be associated to Onion-Search, which manages the string equality operator. At the time of a table creation, the database administrator (DBA) has the possibility to specify only a column's name and data type, as in normal relational databases, because our architecture can automatically choose the default set of Onions with regard to the column data type.

3.2 Metadata Structure:

Metadata include all information that allows a legitimate client knowing the master key to execute SQL operations over an encrypted database. They are organized and stored at table-level granularity to reduce communication over head for retrieval, and to improve management of concurrent SQL operations. We define all metadata information associated with a table as table metadata. Let us describe the structure of a table metadata by referring to fig.3 Table metadata include the correspondence between the plain table name and the encrypted table name because each encrypted table name is randomly generated. Moreover, for each column of the original plain table they also include a set of column metadata containing the name and the data type of the corresponding plain column (e.g., integer, varchar, datetime).

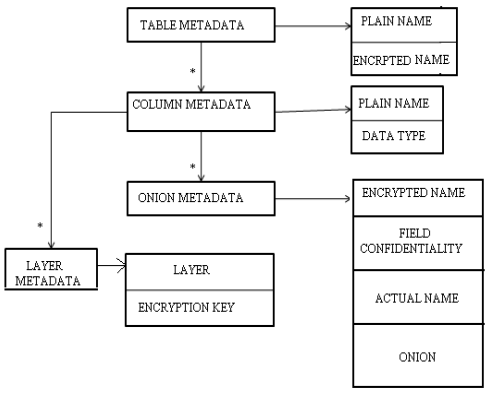


Fig. 3: MetadataStructure.

External layer of the encrypted data (e.g., Rand)stored in the column the field confidentiality denotes which set of keys must be used to encrypt a column data, because only columns that share the same encryption keys can be joined; we identify three types of field confidentiality parameters: self denotes a private set of keys for the column, multi-column identifies the sharing of the same set of keys among two columns, database imposes the same set of keys on all columns of the same data type, the onion parameter identifies the type of onion that is used to encrypt data (e.g., Onion-Eq).Metadata generated by Secure DBaaS contain all the information that is necessary to manage SQL statements over the encrypted database in a way transparent to the user. Metadata management strategies represent an original idea because Secure DBaaS is the first architecture storing all metadata in the untrusted cloud database together with the encrypted tenant data. Secure DBaaS uses two types of metadata. Database metadata are related to the whole database. There is only one instance of this metadata type for each database. Table metadata are associated with one secure table. Each table metadata contains all information that is necessary to encrypt and decrypt data of the associated secure table. This design choice makes it possible to identify which metadata type is required to execute any SQL statement

3.3 Adaptive Layer Removal:

The adaptive layer removal is the process that dynamically removes the external layer of an onion in order to adaptively support SQL operations issued by legitimate clients. Let us describe the details of the adaptive layer removal mechanism by referring to the following example.

We consider a table T with columns id of type int and name of type string, and a tenant client preparing to issue the following statement to the encrypted cloud database: SELECT FROM T WHERE id < 10. The client encryption engine analyzes the SQL statement, and identifies that the operation id < 10 has to be executed on the encrypted database. Then, the client reads the metadata and checks whether there is the Onion-Ord attribute associated with the column id because this is the only onion supporting the operator<.

If the actual layer of Onion-Ord associated with id is set to Rand, then the client dynamically invokes a stored procedure on the cloud database that removes at runtime the Rand layer of Onion-Ord of the column id, thus leaving the Ope layer exposed. The client can now encrypt the SELECT query that contains the operation id < 10 and issue the encrypted query to the encrypted database that executes it on the Ope layer of Onion-Ord.

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

Example of relationship among estimation (T), reservation (TR) and billing (TB) per.

Algorithm:

An cipher text-policy attribute based encryption scheme consists of four fundamental algorithms: Setup, Encrypt, KeyGen, and Decrypt. In addition, we allow for the option of a fifth algorithm Delegate.

Setup: The setup algorithm takes no input other than the implicit security parameter. It outputs the public parameters PK and a master key MK.

Encrypt (PK; M; A): The encryption algorithm takes as input the public parameters PK, a message M, and an access structure A over the universe of attributes. The algorithm will encrypt M and produce a cipher text CT such that only a user that possesses a set of attributes that satises the access structure will be able to decrypt the message. We will assume that the cipher text implicitly contains A.

Key Generation (MK; S): The key generation algorithm takes as input the master key MK and a set of attributes S that describe the key. It outputs a private key SK.

Decrypt (PK; CT; SK): The decryption algorithm takes as input the public parameters PK, a cipher text CT, which contains an access policy A, and a private key SK, which is a private key for a set S of attributes.

If the set S of attributes satises the access structure A then the algorithm will decrypt the cipher text and return a message.

Implementation:

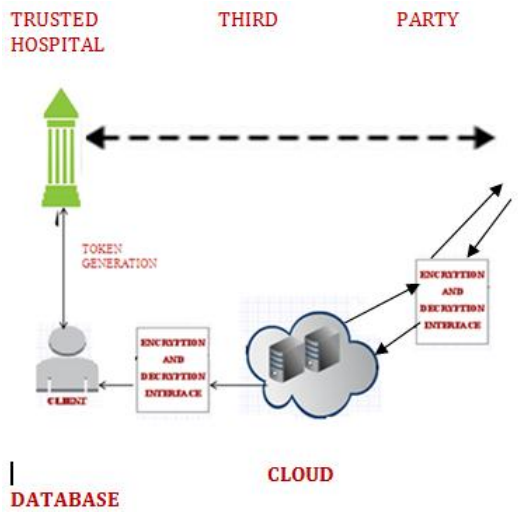
Cloud-assisted electronic health (E-Health) monitoring, which applies the prevailing electronic communications and cloud computing technologies to provide feedback decision support, has been considered as a revolutionary approach to improving the quality of healthcare service while lowering the healthcare cost. Unfortunately, it also poses a serious risk on both clients' privacy and intellectual property of monitoring service providers, which could deter the wide adoption of E-Health technology.

This paper is to address this important problem and design a cloud assisted privacy preserving mobile health monitoring system to protect the privacy of the involved parties and their data. Moreover, the outsourcing decryption technique and a newly proposed key private proxy re-encryption are adapted to shift the computational complexity of the involved parties to the cloud without compromising clients' privacy and service providers' intellectual property. Finally, our security and performance

analysis demonstrates the effectiveness of our proposed design.

To facilitate our discussion, we first elaborate our cloud assisted E-Health monitoring system. CAM consists of four parties: the cloud server (simply the cloud), the company who provides the E-Health monitoring service (i.e., the healthcare service provider), the individual clients (simply clients), and a semi-trusted authority (TA). The company stores its encrypted monitoring data or program in the cloud server. Individual clients collect their medical data and store them in their mobile devices, which then transform the data into attribute vectors. The attribute vectors are delivered as inputs to the monitoring program in the cloud server through a mobile (or smart) device. A semi-trusted authority is responsible for distributing private keys to the individual clients and collecting the service fee from the clients according to a certain business model such as pay-as-you-go business model.

We also do not assume that an individual client colludes with other clients. Our security model does not consider the possible side-channel attack due to the co-residency on shared resources either because it could be mitigated with either system level protection or leakage resilient cryptography CAM assumes an honest but curious model, which implies all parties should follow the prescribed actions and cannot be arbitrarily malicious



Cost estimation of cloud database services:

We consider a tenant that is interested in estimating the cost of porting his database to a cloud platform. This porting is a strategic decision that must evaluate confidentiality issues and related costs over a medium-long term. For these reasons, we propose a model that includes the overhead of encryption schemes and the variability of database work- load 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 Enterprise DB .Windows Azure SQL Database and Rack space Cloud Database.

6.1 Cost Model:

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

$$Cost = f(Time, Pricing, Usage), \quad (1)$$

Where: Time identifies the time interval T for which the ten- ant 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. The total cost of the cloud database service C can be estimated through the following equation:

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

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 xb as its usage at the both billing period and px as its price. If the billing function is

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

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. The current version of the prototype supports the main SQL operations (SELECT, DELETE, INSERT and UPDATE) and the WHERE clause. We consider three TPC-C compliant databases having 10 warehouses: Plaintext (PLAIN) is based on plaintext data. Encrypted (ENC) refers to a statically encrypted data- base where each column is encrypted at design time with only one encryption algorithm. Adaptively encrypted (ADAPT) refers to an encrypted database in which each column is encrypted with all the onions supported by its data type. In the ENC and ADAPT configurations each column is set to the highest encryption layer that supports the SQL operations of the TPC-C workload. During each TPC-C test lasting for 300 seconds, we monitor the number of executed TPC-C transactions, and the response times of all the SQL operations from the standard TPC-C work- load. We repeat the test for each database configuration (PLAIN, ENC and ADAPT) for increasing number of clients (from 5 to 20), and for increasing network latencies (from 0 to 120 ms). To guarantee data consistency the three databases use repeatable read (snapshot) isolation level.

Performance evaluation:

The experiments aim to evaluate the overhead caused by static and adaptive encryption in terms of system throughput and response time. We report the number of committed TPC-C transactions per minute executed on the three cloud database configurations for 5and 20 concurrent clients, respectively. We can

appreciate that in both cases, and in all other results not reported for space reasons, the throughput of the ENC database is close to that of the PLAIN database. Moreover, as the network latency increases, even the performance of the ADAPT database tends to that of the other two configurations

Table: Validation of storage usage of TPC-C Compliant Databases

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

All experimental results show that network latencies higher than 60 ms, which are typical of most cloud database environments, make the adaptive encryption overhead almost negligible when considering the overall set of operations of the TPC-C benchmark. However, in the ADAPT configuration, for some SQL operations involving the Ope encryption or for the encryption of a high number of parameters through several encryption layers (e.g., INSERT), the impact on the response time is visible even for network latencies higher than 120 ms. If the workload is characterized by many INSERT operations, we can conclude that it is a tenant's duty to solve the tradeoff between accepting adaptive encryption overheads and paying the costs related to an entire database re- encryption when workload changes in statically encrypted databases. It is likely that this tradeoff can be solved on the basis of the expected variability of the workload. Possible improvements can be achieved by parallel encryption algorithms that can leverage multi-threading over different cores, but this research is out of the scope of this paper. On the positive hand, we observe that the presented ADAPT configuration represents a worst case scenario that is fully adaptive, because all database columns are encrypted with all the onions supported by its data type. On the other hand, the ENC configuration represents a best case scenario that is completely static, because the user manually defines the single encryption scheme to use on each database column.
Analytical usage estimation methodology in the TPC-C workload. Costs evaluations proposed in the following sections are based on the same usage estimations.

Cost evaluation:

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

Validation of the Usage Estimation:

To validate the usage estimation model, we perform several experiments based on the TPC-C benchmark. First of all, we validate the storage usage estimation model. We deploy nine TPC-C compliant databases of three different sizes: 1, 5 and 10 warehouses (the number of warehouses is the TPC-C parameter that influences the initial database size). For each size, we generate three data- base 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 errors is always below 2%, while for ENC and ADAPT databases the error is always between 5% and 6%. We then validate the network usage estimation model. We deploy PLAIN, ENC and ADAPT TPC-C compliant databases, each having 10 warehouses. We observe that network consumption is invariant with respect to the number of warehouses, because it only depends on encryption and query workload. We measure the network 7,162 Bytes per transaction. By using Equation (8). We estimate np ¼ k° 548. Hence, we determine k ¼ 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. Estimations are quite accurate, since we achieve errors of 1:2% and 1:4% for the ENC and ADAPT configurations ,respectively. The validation demonstrates the efficacy of the proposed analytical usage estimation methodology in the TPC-C workload. Costs evaluations proposed in the following sections are based on the same usage estimations.

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 data- base 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. We report the results for the following cloud instances: Small, Large, and High Memory: Double Extra Large from Amazon RDS Premium P1 and Premium P2 from SQL Azure.

Conclusions:

We proposed an architecture that supports adaptive data confidentiality in cloud database environments. Adaptive encryption mechanisms have two main benefits: they guarantee at run- time the maximum level of data confidentiality for any SQL workload, and they simplify database configuration at design time. However, they are affected by high computational costs with respect to non adaptive encryption schemes. This paper demonstrated that applying adaptive encryption methods to cloud database services is a suitable solution, because network latency masks the overhead caused by adaptive encryption for most SQL operations. Our results also show that the overhead of some SQL operations requiring more encryption steps and more parameters. If the workload is characterized by many similar operations, the present alternative is to accept this cost when data confidentiality is more important than performance. As a future solution, we are also studying encryption parallelization solutions that can leverage multi- threading over different processor cores.

REFERENCES

Boldyreva, A., N. Chenette and A. O'Neill, 2011. "Order-preserving encryption revisited: Improved security analysis and alternative solutions," in Proc. 31st Annu. Int. Conf. Adv. Cryptology, pp: 578–595.

Buyya, R., C.S. Yeo, S. Venugopal, J. Broberg and I. Brandic, 2009. "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generation Comput. Syst., 25(6): 599–616.

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

Ferretti, L., F. Pierazzi, M. Colajanni and M. Marchetti, 2013. "Security and confidentiality solutions for public cloud database services," presented at the 7th Int. Conf. Emerging Security Information, Systems and Technology, Barcelona, Spain.

Ferretti, L., M. Colajanni and M. Marchetti, 2014. "Distributed, concur- rent, and independent access to encrypted cloud databases," IEEE Trans. Parallel Distrib. Syst., 25(2): 437–446.

Gentry, C., 2009. "Fully homomorphic encryption using ideal lattices," in Proc. 41st ACM Symp. Theory Comput, pp: 169–178.

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

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

Paillier, P., 1999. "Public-key cryptosystems based on composite degree residuosity classes," in Proc. 17th Int. Conf. Theory Appl. Crypto- graphic Tech, pp: 223–238.

Popa, R.A., C.M.S. Redfield, N. Zeldovich and H. Balakrishnan, 2011. "CryptDB: Protecting confidentiality with encrypted query proccessing," in Proc. 23rd ACM Symp. Operating Systems Principles, pp: 85–100.

Song, D., D. Wagner and A. Perrig, 2000. "Practical techniques for searches on encrypted data," in Proc. IEEE Symp. Security Privacy, pp: 44–55.

Truong, H.L. and S. Dustdar, 2010. "Composable cost estimation and monitoring for computational applications in cloud computing environments," ProcediaComput. Sci., 1(1): 2175–2184.

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