A Review: A New Approach to spread, Concurrent, and Independent Access to Encrypted Cloud Databases

Madhuri R. Rajput[1], Ankush Narkhede [2]

[1] *Post Graduate Student, Department of CE, Padm.Dr. V.B.K.C.O.E., Malkapur, S.G.B.A. University, Maharashtra, India*

[2] *Asst. Professor, Department of CSE, Padm. Dr. V.B.K.C.O.E., Malkapur, S.G.B.A. University, Maharashtra, India*

[1]madhurirajput495@gmail.com
[2]ankushnarkhede1989@gmail.com

Abstract* — The data under cloud provider is guarantee of security and availability for remaining data in use or not in use. There are numerous services, while private data solution for the database as ideal services is still underdeveloped. They advise novel architecture and design that combine cloud database services with private data and chance to implement simultaneous operation on encrypted report. It is the solution for supporting geographically circulated user to link directly to an encrypted cloud database and to implement simultaneous and independent operation as well as those improves the database structure. The present architecture has the advantages of removing middle proxies that limit the elasticity, availability, and scalability properties that are belonging naturally in cloud based solution. The effectiveness of present architecture is valuated through theoretical study and vast experimental conclusion based on prototype execution subject to the TPCC standard basis for different number of user and network latency.

Keywords-DBaaS, SQL, cloud, database

I. INTRODUCTION

In a cloud situation, where significant data under infrastructures of untrusted third parties, ensure information privacy is of paramount value [2], [3]. This necessity imposes apparent information management choices: original basic information has to be reachable only by trusted parties that do not contain cloud providers, mediators, and Internet; in any untrusted environment, information has to be encrypted. Agreeable these goals have different levels of complication depending on the type of cloud service. There are numerous solutions ensuring privacy for the storage as a service pattern (e.g., [4], [5], [6]), while guaranteeing privacy in the database as a service (DBaaS) model [7] is at rest an open research region. In this situation, we suggest safe DBaaS as the first resolution that allows cloud tenants to get complete benefit of DBaaS qualities, such as availability, reliability, and elastic scalability, not including exposing unencrypted data to the cloud supplier.

The structural design was aggravated by a threefold aim: to permit many, free, and geologically distributed customers to perform parallel operations on encrypted information, containing SQL statements that modify the database formation; to conserve information privacy and reliability at the customer and cloud stage; to remove any middle server among the cloud customer and the cloud supplier. The chance of combining availability, elasticity, and scalability of a classic cloud DBaaS with data privacy is established through a prototype of safe Dabs that supports the execution of parallel and independent operations to the remote encrypted record from many geographically spread customers as in any unencrypted Dbase setup. To get these goals, protected DBase integrates presented cryptographic schemes, separation mechanisms, and novel strategy for management of encrypted metadata on the untrested cloud database. In this context, we cannot change completely homomorphism encryption schemes [8] because of their unnecessary computational difficulty. The safe DBase design is adapted to cloud platforms and does not establish any mediator proxy or adviser server between the customer and the cloud supplier. Removing any trusted middle server allows safe DBase to get the same availability, reliability, and elasticity stages of a cloud DBase. Other proposals (e.g., [9], [10],[11], [12]) based on middle server(s) were considered impossible for a cloud-based solution as any proxy represents a single point of breakdown and a system block that limit the key profit (e.g., scalability, availability, and elasticity) of a record service deployed on a cloud proposal. Unlike safe DBase, architectures relying on a trusted middle proxy do not support the most characteristic cloud state where geographically discrete customers can parallel concern read/write operations and data construction modifications to a cloud database. A huge set of experiments based on actual cloud platforms express that safe DBase is instantly valid to any DBMS because it requires no modification to the cloud record services. added studies where the planned design is focus to the TPC-C standard for special numbers of clients and network latencies show that the presentation of parallel read and write operations not modifying the safe DBase database construction is equivalent to that of unencrypted cloud record. Workloads as well as modifications to the record construction are also supported by safe DBase, but at the price of overheads that seem adequate to get the desired level of data privacy. The motivation of these outcomes is that network latencies, which are typical of cloud scenarios, tend to cover the presentation costs of data encryption on response time.

II. LITERATURE REVIEW

Safe DBaaS supplier many unique features that distinguish it from earlier work in the field of safety for secluded database services. It guarantees information privacy by allowing a cloud record server to perform parallel SQL operation (not only read/write, but also development to the record configuration) above encrypted information. It provides the same availability, elasticity, and scalability of the unique cloud DBaaS because it does not need any middle server. Reply times are affected by cryptographic expenses that for most SQL operations are covered by network latencies. Many customers, may be geographically spread, can access parallel and alone a cloud database service. It does not need a trusted agent or a trusted proxy because tenant information and metadata stored by the cloud database are constantly encrypted. It is compatible with the nearly all popular relational record servers, and it is appropriate to dissimilar DBMS implementations because all adopted solutions are record doubter. Cryptographic file systems and safe storage solutions represent the previous works in this field. In such a way, they avoid one cloud supplier to read its part of information, but data can be reconstructed by colluding cloud supplier. A step ahead is planned in [16] that make it achievable to perform range queries on information and to be robust against collusive supplier. Safe DBaaS differs from these solutions as it does not need the use of many cloud suppliers, and makes use of SQL-aware encryption algorithms to maintain the execution of most general SQL operations on encrypted information. Safe DBaaS relates more closely to works using encryption to guard data managed by entrusted record. In such a case, a main topic to address is that cryptographic techniques cannot be naı¨vely apply to standard DBaaS because DBMS can only perform SQL operations above plaintext information. Some DBMS engines offer the option of encrypting information at the file system stage through the so-called Transparent. Information Encryption feature. This quality makes it possible to construct a trusted DBMS above entrusted storage. However, the DBMS is trusted and decrypts information ahead of their use. Hence, this approach is not valid to the DBaaS background considered by safe DBaas, because we guess that the cloud supplier is untrusted.Other solutions, such as [19], allow the effecting of operations above encrypted information. These approaches protect information privacy in scenarios where the DBMS is not trusted; however, they need a modified DBMS engine and are not compatible with DBMS software used by cloud suppliers. On the other hand, safe DBaaS is compatible with standard DBMS engines, and allows tenants to construct safe cloud record by leveraging cloud DBaaS services previously obtainable. For this reason, safe DBaaS is more correlated to [10] and [9] that protect data privacy in entrusted DBMSs through encryption techniques, permit the implementation of SQL operations above encrypted information, and are compatible with general DBMS engines. However, the construction of these solutions is based on an middle and trusted proxy that mediates any interface between each customer and the entrusted DBMS server. The approach planned in [9]by the authors of the DBaaS model [8] works by encrypting block of information instead of each information point. Whenever a information article that belongs to a block is necessary, the trusted proxy require to improve the complete block, to decrypt it, and to clean out unnecessary information that belong to the same building block. Since a consequence, this design choice requires heavy modifications of the unique SQL operations formed by each customer, thus causing important overheads on both the DBMS server and the trusted proxy. Another works [11], [12] introduce optimization and simplification that expand the subset of SQL operators maintained by [9], but they share the same proxy-based design and its essential issues. On the other hand, safe DBaaS permits the implementation of operations above encrypted information through SQL-aware encryption algorithms. This method, primarily proposed in Crypt DB [8], makes it achievable to perform operations above encrypted information that are analogous to operations above plaintext information In several cases, the doubt plan executed by the DBMS for encrypted and plaintext information is the equal. The trust on a trusted proxy that characterize [11] and[10] facilitates the execution of a safe DBaaS, and is proper to multitier web function, which are their major focus. Though, it causes numerous drawbacks. As the proxy is trusted, its applications cannot be outsourced to an entrusted cloud supplier. so, the proxy is meant to be implemented and managed by the cloud occupant. Availability, scalability, and elasticity of the entire safe DBaaS service are then surrounded by availability, scalability, and elasticity of the trusted proxy, that becomes a particular spot of breakdown and a organization blockage. as high availability, scalability, and elasticity are between the primary reasons that lead to the implementation of cloud services, this restriction hinders the applicability of [9] and [8] to the cloud database situation. Safe DBaaS solves this difficulty by letting customers attach directly to the cloud DBaaS, with no require of other middle part and without initiating latest bottlenecks and single points of breakdown. A proxy-based design requiring that any customer operation should pass throughout one middle server is not appropriate to cloud-based scenarios, in which several customers, normally spread between dissimilar locations, require parallel access to information stored in the similar DBMS. On the other hand, safe DBaaS supports spread customers issuing independent and parallel SQL operations to the similar database and possibly to the similar facts. Safe DBaaS extends our primary studies viewing that information stability can be sure for a few operations by leveraging concurrency separation mechanisms implemented in DBMS engines, and identifying the least separation level essential for those statements. Moreover, we now consider theoretically and experimentally a whole set of SQL operations represented by the TPC-C standard benchmark, in addition to several customers and dissimilar customer-cloud network latencies that were never evaluated in the text.

III.RELATED WORK

1. *ARCHITECTURE DESIGN*

Safe DBaaS is designed to permit several and independent customers to join directly to the entrusted cloud DBaaS without any middle server. Fig. a. explains the overall design. We assume that a tenant organization obtain a cloud database service from an entrusted DBaaS supplier. The tenant then install one or more machines (Client 1 through N) and installs a safe DBaaS customer on each of them. This customer permit a client to join to the cloud DBaaS to manage it, to read and write information, and even to generate and modify the database tables after formation. We imagine the same safety model that is generally accept by the literature in this field (e.g., [9], [10]), where occupant consumers are trusted, the network is entrusted, and the cloud supplier is honest-but-curious, so as to , cloud service operations are accomplished properly, but tenant data confidentiality is at danger. For these cause, tenant information, data construction, and metadata should be encrypted previous to exit from the customer. A thorough presentation of the safety model accepted and obtainable in the online supplemental matter.

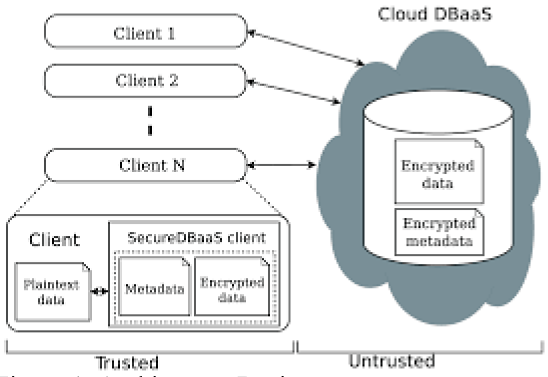

Figure 1. Architecture Design

The information handled by safe DBaaS contains plaintext information, encrypted information, metadata, and encrypted metadata. Plaintext information consists of data that a tenant wants to store and process slightly in the cloud DBaaS. To avoid an entrusted cloud supplier from violating privacy of tenant information stored in plain form, safe DBaaS accept several cryptographic methods to change plaintext information into encrypted tenant information and encrypted tenant information construction because even the names of the tables and of their columns should be encrypted. Safe DBaaS customers create also a set of metadata containing of data essential to encrypt and decrypt information as well as administration information. Still metadata are encrypted and store up in the cloud DBaaS. Safe DBaaS go away from existing architectures that store up only tenant information in the cloud database, and keep metadata in the client machine [10] or crack metadata in the cloud database and a trusted proxy [9]. While considering scenarios where several customers can access the equal database parallel, these earlier results is quite wasteful. For example, saving metadata on the customers would need onerous methods for metadata organization, and the practical impossibility of allowing several customers to accept cloud database services alone. Results based on a trusted proxy are extra feasible, but they initiate a system blockage that decreases accessibility, elasticity, and scalability of cloud record services. Safe DBaaS suggest a dissimilar approach where all information and metadata are save in the cloud database. Protected DBaaS customers can recover the essential metadata from the entrusted database during SQL statements, so that several cases of the safe DBaaS customer can access to the entrusted cloud database alone with the assurance of the similar availability and scalability properties of characteristic cloud DBaaS. Encryption plans for tenant information and innovative results for metadata management and storage are explain in the following two sections.

2. Data Management

We imagine that tenant information is stored in a relational database. We have to save the privacy of the stored information and still of the database construction because table and column names can yield data regarding saved facts. We differentiate the strategies for encrypting the database structures and the tenant information. Encrypted tenant information are stored through safe tables into the cloud database. To permit visible execution of SQL statements, every plaintext table is changed into a protected table because the cloud database is entrusted. The name of a secure table is produced by encrypting the name of the equivalent to plaintext table. Table names are encrypted by means of the similar encryption algorithm and an encryption key that is well-known to all the safe DBaaS customers. Hence, the encrypted name can be calculated from the plaintext name. On the other hand, column names of protected tables are randomly created by protected DBaaS; hence, still if dissimilar plaintext tables have columns withThe similar identity, the names of the columns of the parallel safe tables are dissimilar. This design choice recover privacy by avoiding an adversarial cloud database from imagines relations within dissimilar protected tables through the recognition of columns having the same encrypted name. Protected DBaaS permits tenants to leverage the computational authority of entrusted cloud databases by creating it probable to perform SQL statements remotely and above encrypted tenant information, while distant processing of encrypted information is possible to the extent permitted by the encryption policy. To this use, protected DBaaS expand the idea of data type, that is related with every column of a Traditional database by beginning the protected kind. By selecting a protected kind for every column of a protected table, a tenant able to define fine-grained encryption rules, thus reaching the preferred trade-off among facts privacy and remote processing capacity. A protected kind is composed of three fields: data kind, encryption kind, and field confidentiality. The mixture of the encryption kind and of the field privacy factors defines the encryption rule of the related column. The data category explain the kind of the plaintext data (e.g. int , varchar). The encryption kind recognizes the encryption algorithm which is used to cipher every the facts of a column. It is select within the algorithms hold by the protected DBaaS execution. As in [8], protected DBaaS leverages many SQL-aware encryption algorithms that permit the implementation of statements above encrypted facts. It is significant to examine that every algorithm supports only a subset of SQL operators. When Secure DBaaS creates an encrypted table, the data type of every column of the encrypted table is discovered by the encryption algorithm used to encode tenant information. Two encryption algorithms are defined matched if they produce encrypted information that need the similar column data kind. Since a default performance, safe DBaaS uses a dissimilar encryption input for every column; so, equivalent values stored in dissimilar columns are changed into dissimilar encrypted representation. This architecture selection assurance the maximum confidentiality stage, since it avoids an adversarial cloud supplier to recognize facts that are continual in. encryption key. safe DBaaS suggest three field confidentiality aspect:

a. Column (COL) is the defaulting privacy stage that must be use when SQL statements work on one column; the charge of this column are encrypted throughout a accidentally created encryption key that is not utilize by any another column.

b. Multicolumn (MCOL) must be utilized for columns referenced by link operators, foreign keys, and another operations containing two columns; the two columns are encrypted throughout the same key.

c. Database (DBC) is suggested when operations contain several columns; in this case, it is suitable to utilize the particular encryption key that is created and completely shared with every columns of the database distinguish by the similar protected kind.The option of the field confidentiality stages creates it achievable to perform SQL statements over encrypted information while permitting a tenant to reduce key distribution.

3. *Metadata Management*

Metadata created by protected DBaaS include all the data that is essential to handle SQL statements above the encrypted database in a mode visible to the consumer. Metadata management policies represent an original plan because protected DBaaS is the first design storing all metadata in the entrusted cloud database collectively with encrypted tenant information. Protected DBaaS utilized two types of metadata.

a. Database metadata are correlated to the entire database. There is only one occasion of this metadata kind for every database.

b. Table metadata are related with secure table. Every table metadata include all information that is essential to encrypt and decrypt facts of the related protected table. Database metadata include the encryption key that are utilize for the protected kind containing the field isolation set-to database. The organization of a table metadata is shown in Figure. 2. as below

Figure 2.Structure of metadata

Table metadata include the name of the correlated protected table and the unencrypted name of the correlated plaintext table. Furthermore, table metadata contain column metadata for all column of the correlated protected table. Every column metadata include the following data.

a. Plain name: - the name of the parallel column other plaintext table.

b. Coded name: - the name of the column of the protected table. This is the single information that links a column to the parallel plaintext column since column names of protected tables are accidentally created.

c. Secure type: - the secure kind of the column, as defined in some above parts. This permits a protected DBaaS consumer to be informed regarding the data type and the encryption strategies connected with a column.

d. Encryption key: -the key utilized to encrypt and decrypt the every information saved in the column.

4. OPERATIONS

In this section, we summarize the arrangement setting operations holding out by a database administrator (DBA), and we explain the effecting of SQL actions on encrypted is as.

a. Setup Phase:-We explain how to initialize a protected DBaaS design from a cloud database service obtained by a tenant from a cloud supplier. We imagine that the DBA generates the metadata storage table that at the starting includes only the database metadata, . The DBA populate the database metadata during the protected DBaaS customer by utilizing randomly created encryption keys for any mixture of data types and encryption types, and saves them in the metadata storage table behind encryption during the master key. Then, the DBA spreads the master key to the valid consumer. Consumer access manage strategies are administrated by the DBA throughout a few standard facts control language as in some unencrypted database. In the next steps, the DBA generates the tables of the encrypted database. It should be consider the three field privacy aspects (COL, MCOL, and DBC) initiates in the above section. Let us explain this stage by referring to a easy but representative example shown in Figure. 3, there are three safe tables as ST1, ST2, and ST3. Every table STi (i ¼ 1; 2; 3) contains an encrypted table Ti that includes encrypted tenant information, and a table metadata Mi. (though, in certainty, the names of the columns of the protected tables are at random created; for the sake of simplicity, this figure use them throughC1-CN.)

For example, if the database has to maintain a attach statement within the value of T1.C2 and T2.C1, the DBA should use the MCOL field privacy for T2.C1 that references T1.C2 (solid arrow). In such a way, safe DBaaS can recover the encryption key particular in the column metadata of T1.C2 from the metadata table M1 and can utilize the similar key for T2.C1. The solid arrow from M2 to M1 indicates that they openly distribute the encryption algorithm and the key. As operations (e.g., arithmetical, order relationship) involve more than two columns, it is suitable to accept the DBC field privacy. This has a double advantage: we can utilize the particular encryption key that is created and completely shared within every the columns of the database feature by the similar protected kind; we boundary possible consistency topics in some situations characterized by parallel consumers

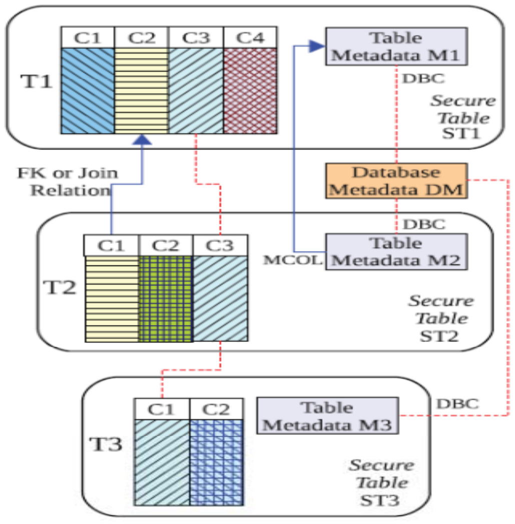

Figure 3.Management of Encryption key

For example, the columns T1.C3, T2.C3, and T3.C1 in Figure.3 distribute the similar protected kind. So, they indication the database metadata, as shown by the dashed line, and utilize the encryption key related with their information and encryption kinds. When they have the similar information and encryption kinds, T1.C3, T2.C3, and T3.C1 can utilize the similar encryption key still if no direct reference survives among them. The database metadata previously include the encryption key K related with the information and the encryption kinds of the three columns, since the encryption keys for every combinations of information and encryption kind are produced in the initialization stage. Hence, K is utilize as the encryption input of the T1.C3, T2.C3, and T3.C1 columns and copied in M1, M2, and M3.

b. Sequential SQL Operations:-We explain the SQL operations in protected DBaaS by think an early easy scenario in which we imagine that the cloud database is accessed by one consumer. Our aim here is to underline the core processing steps; so, we do not take into account presentation optimizations and concurrency issues that will be discussed in below the initial correlation of the consumer with the cloud DBaaS is for confirmation purposes. Protected DBaaS relies on standard confirmation and permission mechanisms supply by the original DBMS server. After the confirmation, a consumer cooperates with the cloud database during the protected DBaaS consumer. Protected DBaaS evaluates the unique operation to recognize which tables are involved and to improve their metadata from the cloud database. The metadata are decrypted throughout the master key. Converted operations include neither plaintext database nor tenant information. However, they are legal SQL operations that the protected DBaaS customer can issue to the cloud database. Converted operations are then performed by the cloud database above the encrypted tenant data. Since there is a one to-one correspondence in plaintext tables and encrypted tables, it is possible to avoid a trusted database consumer from accessing or modifying a few tenant facts by granting restricted rights on a few tables. Consumer rights can be handled by the entrusted and encrypted cloud database. The outcomes of the translated doubt that contains encrypted tenant facts and metadata are gain by the protected DBaaS consumer,decrypted, and send to the consumer.

c. Concurrent SQL Operations

The support to parallel implementation of SQL statements issued by several independent consumers is one of the most important profits of protected DBaaS with respect to state-of-the-art results. Our design should guarantee consistency within encrypted tenant facts and encrypted metadata since corrupted or outdated metadata would avoid consumers from decipher encrypted tenant data resulting in permanent information losses. Now, we mention the significance of distinguishing two classes of statements that are maintained by protected DBaaS: SQL operations not reasoning modifications to the database configuration, such as read, write, and update; operations containing alterations of the database configuration throughout formation, elimination, and modification of database tables. In situations characterized by a static database configuration, protected DBaaS permit customers to issue parallel SQL commands to the encrypted cloud database exclusive of begining any fresh consistency topics with respect to unencrypted databases. Once metadata retrieval, a plaintext SQL command is changed into one SQL command working on encrypted tenant data.

IV. CONCLUSIONS

We study an innovative design that assurances confidentiality of data stored in public cloud databases. Dissimilar state-of-the-art approaches, our result does not depend on a middle proxy that we consider a only one point of breakdown and a bottleneck restrictive availability and scalability of characteristic cloud database services. A huge part of the study contains results to support parallel SQL operation on encrypted data copied by heterogeneous and probably geographically dispersed customers. The study design does not need changes to the cloud database, and it is instantly applicable to existing cloud DBaaS. There are no theoretical with practical borders to extend the outcomes to another platform and to contain new encryption algorithms. It is worth examineing that practical solutions based on the TPC-C standard benchmark indicate that the presentation collision of data encryption on response time become unimportant. In particular, simultaneous read and write processes that do not change the construction of the encrypted database cause slightly overhead. Dynamic situations characterized by simultaneous modifications of the database design are supported, but at the cost of high computational. These presentation producees open the space to future development we are investigating.

REFERENCES

[1] Luca Ferretti, Michele Colajanni, and Mirco Marchetti "Distributed, Concurrent, and Independent Access to Encrypted Cloud Databases" IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 25, NO. 2, FEBRUARY 2014

[2] M. Armbrust et al., "A View of Cloud Computing," Comm. of the ACM, vol. 53, no. 4, pp. 50-58, 2010.

[3] W. Jansen and T. Grance, "Guidelines on Security and Privacy in Public Cloud Computing," Technical Report Special Publication 800-144, NIST, 2011.

[4] A.J. Feldman, W.P. Zeller, M.J. Freedman, and E.W. Felten, "SPORC: Group Collaboration Using Untrusted Cloud Resources,"Proc. Ninth USENIX Conf. Operating Systems Design and mplementation, Oct. 2010.

[5] J. Li, M. Krohn, D. Mazie`res, and D. Shasha, "Secure Untrusted Data Repository (SUNDR)," Proc. Sixth USENIX Conf. Opearting Systems Design and Implementation, Oct. 2004.

[6] P. Mahajan, S. Setty, S. Lee, A. Clement, L. Alvisi, M. Dahlin, and M. Walfish, "Depot: Cloud Storage with Minimal Trust," ACM Trans. Computer Systems, vol. 29, no. 4, article 12, 2011.

[7] H. Hacigu¨mu¨ s,, B. Iyer, and S. Mehrotra, "Providing Database a Service," Proc. 18th IEEE Int'l Conf. Data Eng., Feb. 2002.

[8] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," Proc. 41st Ann. ACM Symp. Theory of Computing, May 2009.

[9] R.A. Popa, C.M.S. Redfield, N. Zeldovich, and H. Balakrishnan, CryptDB: Protecting Confidentiality with Encrypted Query Processing," Proc. 23rd ACM Symp. Operating Systems Principles, Oct. 2011.

[10] H. Hacigu¨mu¨ s,, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over Encrypted Data in the Database-Service-Provider Model," Proc. ACM SIGMOD Int'l Conf. Management Data, June 2002.

[11] J. Li and E. Omiecinski, "Efficiency and Security Trade-Off in Supporting Range Queries on Encrypted Databases," Proc. 19th Ann. IFIP WG 11.3 Working Conf. Data and Applications Security, Aug. 2005.

[12] E. Mykletun and G. Tsudik, "Aggregation Queries in the Database-as-a-Service Model," Proc. 20th Ann. IFIP WG 11.3 Working Conf. Data and Applications Security, July/Aug. 2006.

[13] D. Agrawal, A.E. Abbadi, F. Emekci, and A. Metwally, "Database Management as a Service: Challenges and Opportunities," Proc. 25th IEEE Int'l Conf. Data Eng., Mar.-Apr. 2009.

[14] V. Ganapathy, D. Thomas, T. Feder, H. Garcia-Molina, and R Motwani, "Distributing Data for Secure Database Services," Proc. Fourth ACM Int'l Workshop Privacy and Anonymity in th Information Soc., Mar. 2011.

[15] A. Shamir, "How to Share a Secret," Comm. of the ACM, vol. 22, no. 11, pp. 612-613, 1979.

[16] M. Hadavi, E. Damiani, R. Jalili, S. Cimato, and Z. Ganjei, "AS5 A Secure Searchable Secret Sharing Scheme for Privacy Preserving Database Outsourcing," Proc. Fifth Int'l Workshop Autonomou and Spontaneous Security, Sept. 2013. "Oracle Advanced Security," Oracle Corporation, http://www. oracle.com/technetwork/database/options/advance security, Apr. 2013..

[17] L. Ferretti, M. Colajanni, and M. Marchetti, "Supporting Security and Consistency for Cloud Database," Proc. Fourth Int'l Symp.Cyberspace Safety and Security, Dec. 2012.

[18] H. Berenson, P. Bernstein, J. Gray, J. Melton, E. O'Neil, and P. O'Neil, "A Critique of Ansi Sql Isolation Levels," Proc. ACM SIGMOD, June 1995.

[19] "Transaction Processing Performance Council," TPC-http:// www.tpc.org, Apr. 2013.

[20] "Xeround: The Cloud Database," Xeround, http://xeround.com,

[21] "Postgres Plus Cloud Database," EnterpriseDB, http:// enterprisedb.com/cloud-database, Apr. 2013.

[22] "Windows Azure," Microsoft corporation, http://www. windowsazure.com, Apr. 2013.

[23] "Amazon Elastic Compute Cloud (Amazon Ec2)," Amazon Web Services (AWS), http://aws.amazon.com/ec2, Apr. 2013.

[24] A. Fekete, D. Liarokapis, E. O'Neil, P. O'Neil, and D. Shasha" "Making Snapshot Isolation Serializable," ACM Trans. Database Systems, vol. 30, no. 2, pp. 492-528, 2005.

[25] A. Boldyreva, N. Chenette, and A. O'Neill, "Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions," Proc. 31st Ann. Conf. Advances in Cryptology (CRYPTO '11), Aug. 2011.

[26] "IP Latency Statistics," Verizon, http://www.verizonbusiness com/about/network/latency, Apr. 2013