International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 4, Issue 5, May 2017)

Cloud Database Services with Data Confidentiality and Possibility of Executing Concurrent Operations on Encrypted Data

Mr. J. Sagar Babu, Asst. Prof
Dept. of CSE,
Princeton College of Engineering & Tech, Hyderabad, India

Mr. Yakhoob, Asst. Prof
Dept. of CSE,
Princeton College of Engineering & Tech, Hyderabad, India

Mr. E. Laxman, Asst. Prof
Dept. of CSE,
Princeton College of Engineering & Tech, Hyderabad, India

***Abstract*— Placing critical data in the hands of a cloud provider should come with the guarantee of security and availability for data at rest, in motion, and in use. Several alternatives exist for storage services, while data confidentiality solutions for the database as a service paradigm are still immature. We propose a novel architecture that integrates cloud database services with data confidentiality and the possibility of executing concurrent operations on encrypted data. This is the first solution supporting geographically distributed clients to connect directly to an encrypted cloud database, and to execute concurrent and independent operations including those modifying the database structure. The proposed architecture has the further advantage of eliminating intermediate proxies that limit the elasticity, availability, and scalability properties that are intrinsic in cloud-based solutions. The efficacy of the proposed architecture is evaluated through theoretical analyses and extensive experimental results based on a prototype implementation subject to the TPC-C standard benchmark for different numbers of clients and network latencies.**

***Index Terms*— security, confidentiality, SecureDBaaS, database**

I. INTRODUCTION

IN a cloud context, where critical information is placed in infrastructures of untrusted third parties, ensuring data confidentiality is of paramount importance [1], [2]. This requirement imposes clear data management choices: original plain data must be accessible only by trusted parties that do not include cloud providers, intermediaries, and Internet; in any untrusted context, data must be encrypted. Satisfying these goals has different levels of complexity depending on the type of cloud service. There are several solutions ensuring confidentiality for the storage as a service paradigm (e.g., [3], [4], [5]), while guaranteeing confidentiality in the database as a service (DBaaS) paradigm [6] is still an open research area. In this context, we propose SecureDBaaS as the first solution that allows cloud tenants to take full advantage of DBaaS qualities, such as availability, reliability, and elastic scalability, without exposing unencrypted data to the cloud provider.

The architecture design was motivated by a threefold goal: to allow multiple, independent, and geographically distributed clients to execute concurrent operations on encrypted data, including SQL statements that modify the database structure; to preserve data confidentiality and consistency at the client and cloud level; to eliminate any intermediate server between the cloud client and the cloud provider. The possibility of combining availability, elasti-city, and scalability of a typical cloud DBaaS with data confidentiality is demonstrated through a prototype of SecureDBaaS that supports the execution of concurrent and independent operations to the remote encrypted database from many geographically distributed clients as in any unencrypted DBaaS setup. To achieve these goals, SecureDBaaS integrates existing cryptographic schemes, isolation mechanisms, and novel strategies for management of encrypted metadata on the untrusted cloud database. This paper contains a theoretical discussion about solutions for data consistency issues due to concurrent and independent client accesses to encrypted data. In this context, we cannot apply fully homomorphic encryption schemes [7] because of their excessive computational complexity.

The SecureDBaaS architecture is tailored to cloud platforms and does not introduce any intermediary proxy or broker server between the client and the cloud provider. Eliminating any trusted intermediate server allows SecureDBaaS to achieve the same availability, reliability, and elasticity levels of a cloud DBaaS. Other proposals (e.g., [8], [9], [10], [11]) based on intermediate server(s) were considered impracticable for a cloud-based solution because any proxy represents a single point of failure and a system bottleneck that limits the main benefits (e.g., scalability, availability, and elasticity) of a database service deployed on a cloud platform. Unlike SecureDBaaS, architectures relying on a trusted intermediate proxy do not support the most typical cloud scenario where geographically dispersed clients can concurrently issue read/write operations and data structure modifications to a cloud database.

A large set of experiments based on real cloud platforms demonstrate that SecureDBaaS is immediately applicable to any DBMS because it requires no modification to the cloud database services. Other studies where the proposed architecture is subject to the TPC-C standard benchmark for different numbers of clients and network latencies show that

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 4, Issue 5, May 2017)

the performance of concurrent read and write operations not modifying the SecureDBaaS database structure is comparable to that of unencrypted cloud database. Workloads including modifications to the data-base structure are also supported by SecureDBaaS, but at the price of overheads that seem acceptable to achieve the desired level of data confidentiality. The motivation of these results is that network latencies, which are typical of cloud scenarios, tend to mask the performance costs of data encryption on response time. The overall conclusions of this paper are important because for the first time they demonstrate the applicability of encryption to cloud database services in terms of feasibility and performance.

The remaining part of this paper is structured as follows: Section 2 compares our proposal to existing solutions related to confidentiality in cloud database services. Sections 3 and 4 describe the overall architecture and how it supports its main operations, respectively. Section 5 reports some experimental evaluation achieved through the implemented prototype. Section 6 outlines the main results. Space limitation requires us to postpone the assumed security model in Appendix A, which can be found on the Computer Society Digital Library athttp://doi. ieeecomputersociety.org/10.1109/TPDS.2013.154, to de-scribe our solutions to concurrency and data consistency problems in Appendix B, available in the online supple-mental material, to detail the prototype architecture in Appendix C, available in the online supplemental material.

II. RELATED WORK

SecureDBaaS provides several original features that differentiate it from previous work in the field of security for remote database services.

- It guarantees data confidentiality by allowing a cloud database server to execute concurrent SQL operations (not only read/write, but also modifications to the database structure) over encrypted data.
- It provides the same availability, elasticity, and scalability of the original cloud DBaaS because it does not require any intermediate server. Response times are affected by cryptographic overheads that for most SQL operations are masked by network latencies.
- Multiple clients, possibly geographically distributed, can access concurrently and independently a cloud database service.
- It does not require a trusted broker or a trusted proxy because tenant data and metadata stored by the cloud database are always encrypted.
- It is compatible with the most popular relational database servers, and it is applicable to different DBMS implementations because all adopted solutions are database agnostic.

Cryptographic file systems and secure storage solutions represent the earliest works in this field. We do not detail the several papers and products (e.g., Sporc [3], Sundr [4], Depot [5]) because they do not support computations on encrypted data.

Different approaches guarantee some confidentiality (e.g., [12], [13]) by distributing data among different providers and by taking advantage of secret sharing [14].

In such a way, they prevent one cloud provider to read its portion of data, but information can be reconstructed by colluding cloud providers. A step forward is proposed in [15], that makes it possible to execute range queries on data and to be robust against collusive providers. SecureDBaaS differs from these solutions as it does not require the use of multiple cloud providers, and makes use of SQL-aware encryption algorithms to support the execution of most common SQL operations on encrypted data.

SecureDBaaS relates more closely to works using encryption to protect data managed by untrusted databases. In such a 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.

III. ARCHITECTURE DESIGN

SecureDBaaS is designed to allow multiple and independent clients to connect directly to the untrusted cloud DBaaS without any intermediate server. Fig. 1 describes the overall architecture. We assume that a tenant organization acquires a cloud database service from an untrusted DBaaS provider. The tenant then deploys one or more machines (Client 1 through N) and installs a SecureDBaaS client on each of them. This client allows a user to connect to the cloud DBaaS to administer it, to read and write data, and even to create and modify the database tables after creation.

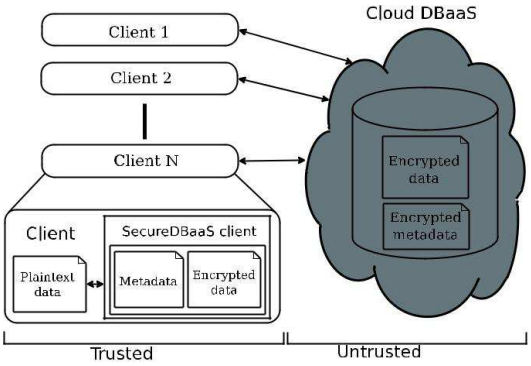

Fig.1. SecureDBaaS architecture

We assume the same security model that is commonly adopted by the literature in this field (e.g., [8], [9]), where tenant users are trusted, the network is untrusted, and the cloud provider is honest-but-curious, that is, cloud service operations are executed correctly, but tenant information confidentiality is at risk. For these reasons, tenant data, data

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 4, Issue 5, May 2017)

structures, and metadata must be encrypted before exiting from the client. A thorough presentation of the security model adopted in this paper is in Appendix A, available in the online supplemental material.

The information managed by SecureDBaaS includes plaintext data, encrypted data, metadata, and encrypted metadata. Plaintext data consist of information that a tenant wants to store and process remotely in the cloud DBaaS. To prevent an untrusted cloud provider from violating confidentiality of tenant data stored in plain form, SecureDBaaS adopts multiple cryptographic techniques to transform plaintext data into encrypted tenant data and encrypted tenant data structures because even the names of the tables and of their columns must be encrypted. SecureDBaaS clients produce also a set of metadata consisting of information required to encrypt and decrypt data as well as other administration information. Even metadata are encrypted and stored in the cloud DBaaS.

SecureDBaaS moves away from existing architectures that store just tenant data in the cloud database, and save metadata in the client machine [9] or split metadata between the cloud database and a trusted proxy [8]. When considering scenarios where multiple clients can access the same database concurrently, these previous solutions are quite inefficient. For example, saving metadata on the clients would require onerous mechanisms for metadata synchronization, and the practical impossibility of allowing multiple clients to access cloud database services independently. Solutions based on a trusted proxy are more feasible, but they introduce a system bottleneck that reduces availability, elasticity, and scalability of cloud database services.

SecureDBaaS proposes a different approach where all data and metadata are stored in the cloud database. SecureDBaaS clients can retrieve the necessary metadata from the untrusted database through SQL statements, so that multiple instances of the SecureDBaaS client can access to the untrusted cloud database independently with the guarantee of the same availability and scalability properties of typical cloud DBaaS. Encryption strategies for tenant data and innovative solutions for metadata management and storage are described in the following two sections.

IV. OPERATIONS

In this section, we outline the setup setting operations carried out by a database administrator (DBA), and we describe the execution of SQL operations on encrypted data in two scenarios: a naıve context characterized by a single client, and realistic contexts where the database services are accessed by concurrent clients.

- **Setup Phase**

We describe how to initialize a SecureDBaaS architecture from a cloud database service acquired by a tenant from a cloud provider. We assume that the DBA creates the metadata storage table that at the beginning contains just the database metadata, and not the table metadata. The DBA populates the database metadata through the SecureDBaaS client by using randomly generated encryption keys for any combinations of data types and encryption types, and stores them in the metadata storage table after encryption through the master key. Then, the DBA distributes the master key to the legitimate users. User access control policies are administrated by the DBA through some standard data control language as in any unencrypted database.

In the following steps, the DBA creates the tables of the encrypted database. It must consider the three field confidentiality attributes (COL, MCOL, and DBC) introduced at the end of the Section 3. Let us describe this phase by referring to a simple but representative example shown in Fig. 4, where we have three secure tables named ST1, ST2, and ST3. Each table STi (i ¼ 1; 2; 3) includes an encrypted table Ti that contains encrypted tenant data, and a table metadata Mi. (Although, in reality, the names of the columns of the secure tables are randomly generated; for the sake of simplicity, this figure refers to them through C1-CN.)

For example, if the database has to support a joint statement among the values of T1.C2 and T2.C1, the DBA must use the MCOL field confidentiality for T2.C1 that references T1.C2 (solid arrow). In such a way, SecureDBaaS can retrieve the encryption key specified in the column metadata of T1.C2 from the metadata table M1 and can use the same key for T2.C1. The solid arrow from M2 to M1 denotes that they explicitly share the encryption algorithm and the key.

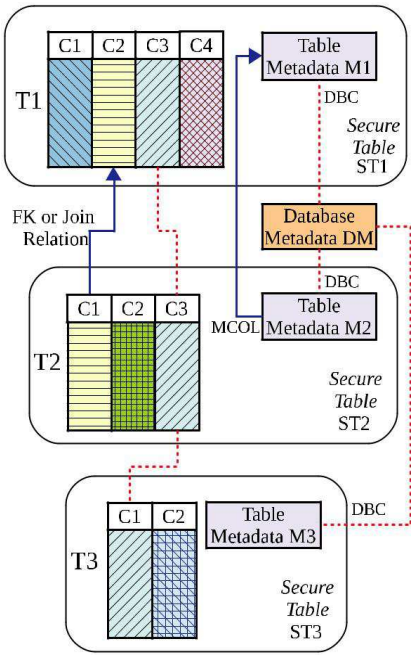

Fig.2. Management of the encryption keys according to the field confidentiality parameter.

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 4, Issue 5, May 2017)

When operations (e.g., algebraic, order comparison) involve more than two columns, it is convenient to adopt the DBC field confidentiality. This has a twofold advantage: we can use the special encryption key that is generated and implicitly shared among all the columns of the database characterized by the same secure type; we limit possible consistency issues in some scenarios characterized by concurrent clients (see Appendix B, available in the online supplemental material). For example, the columns T1.C3, T2.C3, and T3.C1 in Fig. 4 share the same secure type. Hence, they reference the database metadata, as represented by the dashed line, and use the encryption key associated with their data and encryption types. As they have the same data and encryption types, T1.C3, T2.C3, and T3.C1 can use the same encryption key even if no direct reference exists between them. The database metadata already contain the encryption key K associated with the data and the encryption types of the three columns, because the encryption keys for all combinations of data and encryption types are created in the initialization phase. Hence, K is used as the encryption key of the T1.C3, T2.C3, and T3.C1 columns and copied in M1, M2, and M3.

- **Sequential SQL Operations**

We describe the SQL operations in SecureDBaaS by considering an initial simple scenario in which we assume that the cloud database is accessed by one client. Our goal here is to highlight the main processing steps; hence, we do not take into account performance optimizations and concurrency issues that will be discussed in Section 4.3 and Appendix B, available in the online supplemental material.

The first connection of the client with the cloud DBaaS is for authentication purposes. SecureDBaaS relies on standard authentication and authorization mechanisms pro-vided by the original DBMS server. After the authentication, a user interacts with the cloud database through the SecureDBaaS client. SecureDBaaS analyzes the original operation to identify which tables are involved and to retrieve their metadata from the cloud database. The metadata are decrypted through the master key and their information is used to translate the original plain SQL into a query that operates on the encrypted database.

Translated operations contain neither plaintext database (table and column names) nor plaintext tenant data. Nevertheless, they are valid SQL operations that the SecureDBaaS client can issue to the cloud database. Translated operations are then executed by the cloud database over the encrypted tenant data. As there is a one-to-one correspondence between plaintext tables and encrypted tables, it is possible to prevent a trusted database user from accessing or modifying some tenant data by granting limited privileges on some tables. User privileges can be managed directly by the untrusted and encrypted cloud database. The results of the translated query that includes encrypted tenant data and metadata are received by the SecureDBaaS client, decrypted, and delivered to the user. The complexity of the translation process depends on the type of SQL statement.

- **Concurrent SQL Operations**

The support to concurrent execution of SQL statements issued by multiple independent (and possibly geographically distributed) clients is one of the most important benefits of SecureDBaaS with respect to state-of-the-art solutions. Our architecture must guarantee consistency among encrypted tenant data and encrypted metadata because corrupted or out-of-date metadata would prevent clients from decoding encrypted tenant data resulting in permanent data losses. A thorough analysis of the possible issues and solutions related to concurrent SQL operations on encrypted tenant data and metadata is contained in Appendix B, available in the online supplemental material. Here, we remark the importance of distinguishing two classes of statements that are supported by SecureDBaaS: SQL operations not causing modifications to the database structure, such as read, write, and update; operations involving alterations of the database structure through creation, removal, and modification of database tables (data definition layer operators).

In scenarios characterized by a static database structure, SecureDBaaS allows clients to issue concurrent SQL commands to the encrypted cloud database without introducing any new consistency issues with respect to unencrypted databases. After metadata retrieval, a plaintext SQL command is translated into one SQL command operating on encrypted tenant data. As metadata do not change, a client can read them once and cache them for further uses, thus improving performance.

SecureDBaaS is the first architecture that allows concurrent and consistent accesses even when there are operations that can modify the database structure. In such cases, we have to guarantee the consistency of data and metadata through isolation levels, such as the snapshot isolation [21], that we demonstrate can work for most usage scenarios.

V. EXPERIMENTAL RESULTS

We demonstrate the applicability of SecureDBaaS to different cloud DBaaS solutions by implementing and handling encrypted database operations on emulated and real cloud infrastructures. The present version of the SecureDBaaS prototype supports PostgreSQL, MySql, and SQL Server relational databases. As a first result, we can observe that porting SecureDBaaS to different DBMS required minor changes related to the database connector, and minimal modifications of the codebase. We refer to Appendix C, available in the online supplemental material, for an in-depth description of the prototype implementation.

Other tests are oriented to verify the functionality of SecureDBaaS on different cloud database providers.

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 4, Issue 5, May 2017)

Experiments are carried out in Xeround [22], Postgres Plus Cloud Database [23], Windows SQL Azure [24], and also on an IaaS provider, such as Amazon EC2 [25], that requires a manual setup of the database. The first group of cloud providers offer ready-to-use solutions to tenants, but they do not allow a full access to the database system. For example, Xeround provides a standard MySql interface and proprietary APIs that simplify scalability and availability of the cloud database, but do not allow a direct access to the machine. This prevents the installation of additional software, the use of tools, and any customization. On the positive side, SecureDBaaS using just standard SQL commands can encrypt tenant data on any cloud database service. Some advanced computation on encrypted data may require the installation of custom libraries on the cloud infrastructure. This is the case of Postgres plus Cloud that provides SSH access to enrich the database with additional functions.

The next sets of experiments evaluate the performance and the overheads of our prototype. We use the Emulab [26] testbed that provides us a controlled environment with several machines, ensuring repeatability of the experiments for the variety of scenarios to consider in terms of workload models, number of clients, and network latencies.

VI. CONCLUSIONS

We propose an innovative architecture that guarantees confidentiality of data stored in public cloud databases. Unlike state-of-the-art approaches, our solution does not rely on an intermediate proxy that we consider a single point of failure and a bottleneck limiting availability and scalability of typical cloud database services. A large part of the research includes solutions to support concurrent SQL operations (including statements modifying the database structure) on encrypted data issued by heterogenous and possibly geographically dispersed clients. The proposed architecture does not require modifications to the cloud database, and it is immediately applicable to existing cloud DBaaS, such as the experimented PostgreSQL plus Cloud Database [23], Windows Azure [24], and Xeround [22]. There are no theoretical and practical limits to extend our solution to other platforms and to include new encryption algorithms.

It is worth observing that experimental results based on the TPC-C standard benchmark show that the performance impact of data encryption on response time becomes negligible because it is masked by network latencies that are typical of cloud scenarios. In particular, concurrent read and write operations that do not modify the structure of the encrypted database cause negligible overhead. Dynamic scenarios characterized by (possibly) concurrent modifications of the database structure are supported, but at the price of high computational costs. These performance results open the space to future improvements that we are investigating.

REFERENCES

[1] M. Armbrust et al.,"A View of Cloud Computing," Comm. of t he ACM, vol. 53, no. 4, pp. 50-58, 2010.
[2] W. Jansen and T. Grance, "Guidelines on Security and Privacy in Public Cloud Computing,"Technical Report Special Publication 800-144, NIST, 2011.
[3] A.J. Feldman, W.P. Zeller, M.J. Freedman, and E.W. Felten, "SPORC: Group Collaboration Using Untrusted Cloud Re-sources," Proc. Ninth USENIX Conf. Operating Systems Design and Implementation, Oct. 2010.
[4] 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.
[5] 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.
[6] H. Hacigu¨mu¨s¸, B. Iyer, and S. Mehrotra, "Providing Database as a Service," Proc. 18th IEEE Int'l Conf. Data Eng., Feb. 2002.
[7] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," Proc. 41st Ann. ACM Symp. Theory of Computing, May 2009.
[8] 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.
[9] 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.
[10] 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.
[11] 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.
[12] 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.
[13] 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 the Information Soc., Mar. 2011.
[14] A. Shamir, "How to Share a Secret," Comm. of the ACM, vol. 22, no. 11, pp. 612-613, 1979.
[15] 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 Autonomous and Spontaneous Security, Sept. 2013.
[16] "Oracle Advanced Security," Oracle Corporation, http://www.oracle.com/technetwork/database/options/advanced-security, Apr. 2013.
[17] G. Cattaneo, L. Catuogno, A.D. Sorbo, and P. Persiano, "The Design and Implementation of a Transparent Cryptographic File System For Unix," Proc. FREENIX Track: 2001 USENIX Ann. Technical Conf., Apr. 2001.
[18] E. Damiani, S.D.C. Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati, "Balancing Confidentiality and Efficiency in Untrusted Relational Dbmss," Proc. Tenth ACM Conf. Computer and Comm. Security, Oct. 2003.
[19] 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.
[20] "Transaction Processing Performance Council," TPC-C, http:// www.tpc.org, Apr. 2013.
[21] 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.
[22] "Xeround: The Cloud Database," Xeround, http://xeround.com, Apr. 2013.
[23] "Postgres Plus Cloud Database," EnterpriseDB, http:// enterprisedb.com/cloud-database, Apr. 2013.

[24] "Windows Azure," Microsoft corporation, http://www. windowsazure.com, Apr. 2013.
[25] "Amazon Elastic Compute Cloud (Amazon Ec2)," Amazon Web Services (AWS), http://aws.amazon.com/ec2, Apr. 2013.
[26] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An Integrated Experimental Environment for Distributed Systems and Net-works," Proc. Fifth USENIX Conf. Operating Systems Design and Implementation, Dec. 2002.
[27] 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.
[28] 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.
[29] "IPLatencyStatistics,"Verizon,http://www.verizonbusiness. com/about/network/latency, Apr. 2013.

ABOUT THE AUTHORS

J. Sagar babu, Asst. Professor

Received B.Tech degree in Computer Science and Engineering from the University of JNTU Hyderabad and M.Tech degree in Computer Science and Engineering from the JNTU-Hyderabad. He is currently working as an Asst. Professor in CSE Department at Princeton College of Engineering & Technology, Hyderabad. Up to now he was attended several National and International Conferences, Workshops.

K. Yakhoob, Asst. Professor

Received B.Tech degree in Information Technology from the University of JNTU Hyderabad and M.Tech degree in Computer Science and Engineering from the JNTU-Hyderabad. He is currently working as an Asst. Professor in CSE Department at Princeton College of Engineering & Technology, Hyderabad. Up to now he was attended several National and International Conferences, Workshops.

E. Laxman, Asst. Professor

Received B.Tech degree in Computer Science and Engineering from the University of JNTU Hyderabad and M.Tech degree in Computer Science and Engineering from the JNTU-Hyderabad. He is currently working as an Asst. Professor in CSE Department at Princeton College of Engineering & Technology, Hyderabad. Up to now he was attended several National and International Conferences, Workshops.