# Onion Encryption Techniques for Cloud Database Security (OnioDB)

**Yogita Manish Datwani**[1], **Prof. Bela Shrimali**[2]
[1, 2] Department of Computer Engineering
[1, 2] LDRP-ITR, Gandhinagar,Gujarat,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 without decrypting it. A novel architecture for adaptive encryption of public cloud databases that offers an interesting alternative to the tradeoff between the required data confidentiality level and the flexibility of the cloud database structures at design time. This paper proposes a novel architecture for adaptive encryption of public cloud databases that offers a proxy-free alternative to the system. The project demonstrates the feasibility and performance of the proposed solution through a software prototype. The proposed architecture manages five types of information: plain data represent the tenant information; encrypted data are the encrypted version of the plain data, and are stored in the cloud database; plain metadata represent the additional information that is necessary to execute SQL operations on encrypted data; encrypted metadata are the encrypted version of the plain metadata, and are stored in the cloud database; master key is the encryption key of the encrypted metadata, and is known by legitimate clients.*

***Keywords***- Cloud Database, Adaptive Encryption, SQL operations, Metadata, Distributed SQL operation, adaptive encryption, onion encryption.

## I. INTRODUCTION

In a cloud context, where critical information is placed in infrastructures of untrusted third parties, ensuring data confidentiality is of paramount importance. 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 while guaranteeing confidentiality in the database as a service (DBaaS) paradigm 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 : to allow multiple and independent 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, elasticity, 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. 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 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 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 the performance of concurrent read and write operations not modifying the SecureDBaaS database.

## II. MOTIVATION

Sustaining critical data in the hands of a cloud provider is often a challenging one that is certainly associated with secrecy. There exist quite a few encryption techniques now available that provides the particular guarantee of safety measures and availability for data. To provide each of the security features to data there were different types of architectures; algorithms are increasingly being developed and are in naive stage. Feasibility, functionality and performance can be calculated using typical tested like emulab and cloud providers like amazon EC2, windows azure, Xeround , Rackspace etc.,

## III. LITERATURE SURVEY

Data storage in a cloud is where the user stores his data through a CSP (cloud service provider) into a set of cloud servers, which are running parallarly and in a distributed manner. The idleness of the data can be employed with technique of erasure-correcting code to further accept faults or server crash as user's data grows in size and importance . Thereafter, for application purposes, the user interacts with the cloud servers via CSP to access or recover his data[5] It is of vital importance to guarantee users that their data are being correctly stored and maintains, as users no longer have their data locally, That is, users should be prepared with security means so that they can make continuous correctness declaration of their data stored in Cloud Servers even without the existence of local copies[14].

D.Pratiba in 2002 proposed that Data mining in Cloud Computing is the process of extracting structured information from unstructured or semi-structured web data sources. The Data mining in Cloud Computing allows information from unstructured or semi- management of software and data storage, with the guarantee of efficient sharing of resources for their users.[16]

ZengDadan in 2003 proposed that an perfect computing platform aims at adapting to different kinds of computing applications. Different applications may prefer different storage systems for their specific needs. A distributed computing platform with broad usability can shorten the developing process distributed applications. If the application requirement changes then different storage systems can be used without the need for additional development or significant changes to the system, making the distributed application more flexible and efficient.[9]

Yu et al In 2005 proposed that Attribute-based encryption is by nature a tool for access control, and thus it is not surprising to see that the primitive is adapted to the setting of cloud computing, where fine grained access control is preferred so as to make the cloud a scalable and convenient platform for data sharing.[15]

Mao Tan in 2006 proposed that Distributed heterogeneous data exchange is an important research topic in the area of data sharing and integration, a significant difficulty in process of data exchange is to determine the problem caused by the heterogeneity of data the architecture of hardware or operating system may be different, which leads to differences in the structure of data storage, the acceptable technologies of data processing can also be different [1]

Shucheng Yu in 2007 proposed that the data owner and cloud servers are very likely to be in two different domains. On one hand, cloud servers are not allowed to access the outsourced data content for data confidentiality; on the other hand, the data resources are not physically under the full control of the owner. For the purpose of helping the data owner as like fine-grained access control of data stored on untrusted cloud servers, a feasible solution would be encrypting data through certain cryptographic primitive(s), and disclose decryption keys only to the authorized users. Unauthorized users, including cloud servers, are not able to decrypt because they do not have the data decryption keys.[18]

Sundareswaran et al. in 2008 proposed that also bundles the data with an access policy. Additionally, a record file is also bundled with the data. Any operation the user carries out will be appended to the log file, and this record file will be from time to time sent to the Cloud. A data owner can then access the record files to check whether data is being used appropriately this prevents from man-in-the-middle attacks [19]

PengzhiXu in 2009 proposed that, the distributed file systems make use of storage resource of product distributed file systems make use of storage supply of service machines to provide a large extensible cost capacity, low storage and high parallel transfer rate. However, our campus cloud is designed as access tools and a set of cloud middleware building upon the storage resources provided by these distributed file systems. Therefore, both works are critical to the personal storage and communities data sharing.[15]. KuiRen in 2010 proposed that, Address this open issue and propose a secure and scalable fine-grained data access control scheme for cloud computing. Our proposed scheme is partially based on our observation that, in practical application scenarios each data file can be related with a set of attributes which are meaningful in the perspective of their interest. The access structure of each user can thus be defined as a unique logical expression over these attributes to reflect the scope of data files that the user is allowed to access. As the logical expression can represent any desired data file set, finegrainedness of data access control is achieved.[12]

Yanjiang Yang in 2011 proposed on attribute-based encryption (or predicate encryption) and proxy encryption, two cryptographic primitives underlying our construction, as well as cryptographic access control mechanisms for cloud computing.[11]

Ateniese et al. in 2012 proposed that at the time of encryptions we secure distributed storage system. Specifically, the data owner encrypts blocks of content with unique and symmetric content keys, which are further encrypted under a master public key. For access control, the server uses proxy cryptography to directly reencrypt the appropriate contents key(s) from the master public key to a granted user’s public key.[17]

Luca Ferretti in 2013 proposed that Different approaches guaranteed some confidentiality by distributing data among different providers and by taking advantage of secret sharing . In such a way, they prevent one cloud provider to read its portion of data, but information can be reconstructed by colluding cloud providers.[8]. Chen et al. in 2014 proposed that bundle the data with an access policy and sending this bundle to authorized use untrusted applications this proposed Architecture called Data Safe. This Data Safe Mechanism allows only to the authorized user to set out the policy to the data so that it can accessible only from the trusted party[1].

## IV. EXISTING&PROPOSED SYSTEMS

A) **Existing System:** 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 Secure Database as a Service paradigm (SecureDBaaS) 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 in applicable to the database paradigm. Other encryption schemes, which allow the execution of SQL operations over encrypted data, either suffer from performance limits or they require the choice of which encryption scheme must be adopted for each database column and SQL operations.

B) **Proposed System:** It guarantees data confidentiality by permitting a cloud information server to execute coinciding SQL operations over encrypted data using adaptive encryption schemes. It provides identical accessibility, security, elasticity and measurability of the first cloud DBaaS as a result of it doesn't need any intermediate proxy. Multiple consumers, probably regionally distributed, will access the same time and severally a cloud information service. It does not need a trusty broker or a trusty server as a result of tenant data and information keep by the cloud information area unit forever encrypted.

## V. RELATED WORK

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. 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. The tenant has two alternatives for any SQL operation : downloading the entire database, decrypting it, executing the query and, if the operation modifies the databases, encrypting and uploading the new data; decrypting temporarily the cloud database, executing the query, and re-encrypting it. The former solution is affected by huge communication and computation overheads, and costs that would make the cloud database services quite inconvenient; the latter solution does not guarantee data confidentiality because the cloud provider obtains decryption keys. The right alternative is to execute SQL operations directly on the cloud database, but avoiding that the provider obtains the decryption key. This proposal is based on data aggregation techniques associate plaintext metadata to sets of encrypted data to allow data retrieval. However, plaintext metadata may leak sensitive information and data aggregation introduces unnecessary network overheads. The use of fully homomorphic encryption would guarantee the execution of any operation over encrypted cloud data, but existing implementations are affected by huge computational costs to the extent that they would make impractical the execution time of SQL operations over a cloud database. Other encryption algorithms characterized by acceptable computational complexity support a subset of SQL operator.

## VI. SYSTEM OVERVIEW

The system mainly focuses on following

1. Cloud database
2. Metadata Management
3. Encryption algorithm

1. **Cloud database:** We assume that tenant data are saved in a relational database. We have to preserve the confidentiality of the stored data and even of the database structure because table and column names may yield information about saved data. We distinguish the strategies for encrypting the database structures and the tenant data.

2. **Metadata Management:** Metadata generated by SecureDBaaS 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 SecureDBaaS is the first architecture storing all metadata in the untrusted cloud database together with the encrypted tenant data.

3. **Encryption Algorithm:** Choosing the encryption algorithms used to encrypt and decrypt all the data stored in the database table

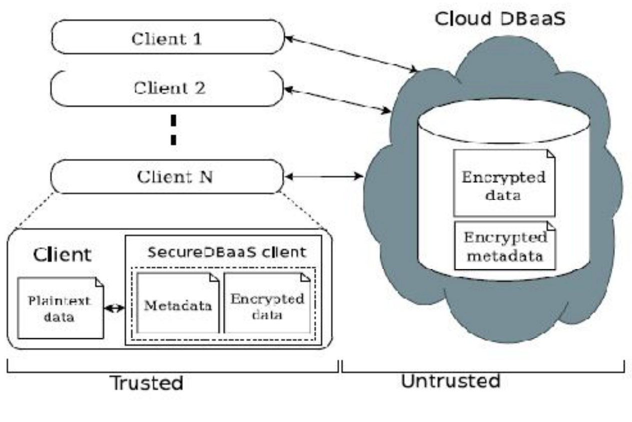


Fig. describes the overall architecture. We assume that a tenant organization acquires a cloud database service from an un-trusted DBaaS provider. The tenant then deploys one or more machines (Client 1 through N) and installs a SecureDBaaS client on each of them. This client allows a user to connect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

## VII. ADAPTIVE ENCRYPTION SCHEME

Consider SQL-aware encryption algorithms that guarantee data confidentiality and allow the cloud database engine to execute SQL operations over encrypted data. As each algorithm supports a specific subset of SQL operators, we refer to the following encryption schemes.

**Random (Rand):** It is the most secure encryption because it does not reveal any information about the original plain value. It does not support any SQL operator, and it is used only for data retrieval.

**Deterministic (Det):** It deterministically encrypts data, so that equality of plaintext data is preserved. It supports the equality operator.

**Order Preserving Encryption (Ope):** It preserves in the encrypted values the numerical order of the original unencrypted data. It supports the comparison SQL operators (=,<,<=,>,>=).

**Homomorphic Sum (Sum):** It is homomorphic with respect to the sum operation, so that the multiplication of encrypted integers is equal to the sum of plaintext integers. It supports the sum operator between integer values.

**Search:** It supports equality check on full strings (i.e., the LIKE operator).

**Plain:** It does not encrypt data, but it is useful to support all SQL operators on non confidential data. If each column of the database was encrypted with only one algorithm, then the database administrator would have to decide at design time which operations must be supported on each database column. However, this solution is impractical for scenarios in which the database workload changes over time.

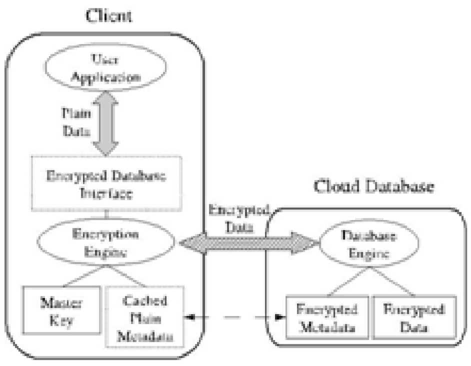
The proposed system in supports adaptive encryption for public cloud database services, where distributed and concurrent clients can issue direct SQL operations. By avoiding an architecture based on intermediate servers between the clients and the cloud database, the proposed solution guarantees the same level of scalability and availability of the cloud service.

The proposed architecture manages five Types of information:

1. **Plain data** represent the tenant information;

2. **Encrypted data** are the encrypted version of the plain data, and are stored in the cloud database;
3. **Plain metadata** represent the additional information that is necessary to execute SQL operations on encrypted data;
4. **Encrypted metadata** are the encrypted version of the plain metadata, and are stored in the cloud database.
5. **Master key** is the encryption key of the encrypted metadata, and is known by legitimate clients. 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. As in related literature, the proposed architecture guarantees data confidentiality in a security model in which: the network is untrusted, tenant users are trusted, that is, they do not reveal information about plain data, plain metadata, and the master key; the cloud provider administrators are defined semi-honest or honest-but-curious, that is, they do not modify tenant's data and results of SQL operations, but they may access tenant's information stored in the cloud database. The remaining part of this section describes the adaptive encryption schemes, the encrypted metadata stored in the cloud database, and the main operations for the management of the encrypted cloud database. Metadata concept used is also considered in the proposed work. After generating the multi-user key its distribution is also done in the algorithm. A cost model can also be derived which will tell the mathematical representation of the algorithm performance. Java platform will be used for the implementation of the algorithm and MYSQL server is used as the back-end. Linux operating system is considered good in the security point of view thus except for windows OS Linux operating system will be used. We are interested in the database as a service paradigm that poses several research challenges in terms of security and cost evaluation from tenant's point of view. Most results concerning encryption for cloud database 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.



## VIII. MODULE DESCRIPTION

**REGISTRATION PHASE WITH ACCESS CONTROL MECHANISM:** Cloud Owner and its remote user would be registered with monotonic access control mechanism. A monotonic access structure is a structure where: given a universal set P, if a subset S! Of P satisfies the access structure, all subsets S! of P which contain S! Satisfy the access structure.

**SECURE DATABASE AS A SERVICE(S-DBAAS):** SecureDBaaS supports the execution of concurrent and independent operations to the remote encrypted database from many geographically distributed clients as in any unencrypted DBaaS setup. It allows cloud tenants to take full advantage of DBaaS qualities, such as availability, security, reliability, elasticity and scalability without exposing unencrypted data to the cloud provider SecureDBaaS adopts multiple cryptographic techniques and isolation mechanism to transform plain text data into encrypted tenant data and encrypted tenant data structures.

**MANAGEMENT OF DATA AND METADATA** Encrypted tenant data are stored through secure tables into the cloud database. To allow easily seen execution of SQL statements, each plain text data is transformed into a secure table because the cloud database is untrusted. Metadata generated by SecureDBaaS contain all the information necessary to manage SQL statements over the encrypted database in a way transparent to the user. There are two types of such a Database metadata are related to the whole database. Table metadata contains all information that is necessary to encrypt and decrypt data of associated secure table.

**CONFIDENTIAL CONCURRENT ACCESS TO DBAAS (CCAD) CONCURRENT SQL OPERATIONS** Support to the execution of SQL statements issued by multiple freelance (and presumably geographically distributed) consumers is one in every of the foremost necessary edges of SecureDBaaS with reference to progressive solutions. Our design should guarantee consistency among encrypted tenant knowledge and encrypted information as a result of corrupted or obsolete data would stop purchasers from decipherment encrypted tenant knowledge leading to permanent knowledge losses. An intensive analysis of the potential problems and solutions associated with synchronic SQL operations on encrypted tenant knowledge and data is contained in Appendix B, out there within the on-line supplemental material. Here, we have a tendency to comment the importance of characteristic two categories of statements that area unit supported by SecureDBaaS: SQL operations not inflicting modifications to the information structure, like browse, write, and update; operations involving alterations of the information structure through creation, removal and modification of information tables.Here, we've got an inclination to remark the importance of distinctive two classes of statements that unit supported by SecureDBaaS: SQL operations not inflicting modifications to the data structure, like scan, write, and update; operations involving alterations of the data structure through creation, removal, and modification of knowledge tables (data definition layer operators). In eventualities characterised by a static information structure, SecureDBaaS permits purchasers to issue synchronize SQL commands to the encrypted cloud information while not introducing any new consistency problems with relevance unencrypted databases. When information retrieval, a plain text SQL command is translated into one SQL command operative on encrypted tenant knowledge. As information does not need modification, a client can browse them once and cache them for additional uses in turn so rising performance. SecureDBaaS is that the first design that permit to synchronize and consistent accesses even once there are operations that may modify the information structure. In such cases, we've got to ensure the consistency of information through isolation levels that we tend to demonstrate will work for many victimization eventualities.

**SEQUENTIAL SQL OPERATIONS** We describe the SQL operations in SecureDBaaS by considering associate degree initial easy situation within which we tend to assume that the cloud information is accessed by one consumer. Our goal is to focus on the most process steps therefore; we do not take into consideration performance optimizations and concurrency problems which will be mentioned but there within the on-line supplemental material. The first affiliation of the consumer with the cloud DBaaS is for authentication functions. SecureDBaaS depends on common place authentication and authorization mechanisms provided by the initial software system server. When the authentication, a user interacts with the cloud information through the SecureDBaaS consumer. SecureDBaaS analyzes the initial operation to spot that tables measure concerned and to retrieve their information from the cloud. The information are decrypted through the key and their data is employed to translate the initial plain SQL into a question that operates on the encrypted information. Instance operations contain neither plain text information (table and column names) nor plain text tenant data still, valid SQL operations that the SecureDBaaS consumer will issue to the cloud information. Translated operations are executed by the cloud information over the encrypted tenant knowledge. There is a one to one correspondence between plain text tables and encrypted tables, it's potential to stop a trust worthy information user from accessing or modifying some tenant knowledge by granting restricted privileges on some tables. User benefit is managed directly by the untrusted and encrypted cloud information. The results of the instance question that focuses encrypted tenant data and information are received by the SecureDBaaS.

consumer, decrypted and delivered to the user. The quality of the interpretation method depends on the kind of SQL statement. In situations characterised by a static information structure, SecureDBaaS permits clients to issue coinciding SQL commands to the encrypted cloud information while not introducing any new consistency problems with relevance unencrypted databases. When metadata retrieval, a plain text SQL command is translated into one SQL command in operation on encrypted tenant knowledge. Data does not need modification, a consumer will browse them once and cache them for additional uses, so rising performance.

SecureDBaaS is that the first design that permits coincident and consistent accesses even once there are operations which will modify the information structure. In such cases, we have got to ensure the consistency of knowledge and information through isolation levels, like the snapshot isolation, that we tend to demonstrate will work for many usage situations. Characterized by a similar secure sort we tend to limit potential consistency problems in some eventualities characterised by synchronous clients. As a example, the column share a similar secure sort. Hence reference the information, as diagrammatical by the dotted line, and use the encoding key related to their knowledge and encoding sorts. As they need a similar data and encoding sorts, will use a similar encoding key though no direct reference exists between them. The information already contain the encryption K related to the information and therefore the encoding forms of the 3 columns, as a result of the cryptography keys for all mixtures of information and encoding sorts are created within the data formatting part.

Hence, K is employed because the encoding key of columns and derived in M1, M2, and M3.

## IX. CONCLUSION

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 Adaptive 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 SecureDBaaS, 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) concurrentmodifications 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.

## BIBLIOGRAPHY

[1] M. Armbrust et al., “A View of Cloud Computing,” Comm. of theACM, vol. 53, no. 4, pp. 50-58, 2010.

[2] W. Jansen and T. Grance, “Guidelines on Security and Privacy inPublic Cloud Computing,” Technical Report Special Publication800-144, NIST, 2011.

[3] 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 Implementation, Oct. 2010.

[4] J. Li, M. Krohn, D. Mazie`res, and D. Shasha, “Secure UntrustedData Repository (SUNDR),” Proc. Sixth USENIX Conf. OpeartingSystems Design and Implementation, Oct. 2004.

[5] P. Mahajan, S. Setty, S. Lee, A. Clement, L. Alvisi, M. Dahlin, andM. Walfish, “Depot: Cloud Storage with Minimal Trust,” ACMTrans. Computer Systems, vol. 29, no. 4, article 12, 2011.

[6] H. Hacigu¨mu¨ s¸, B. Iyer, and S. Mehrotra, “Providing Database as aService,” 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 QueryProcessing,” Proc. 23rd ACM Symp. Operating Systems Principles,Oct. 2011.

[9] H. Hacigu¨mu¨ s¸, B. Iyer, C. Li, and S. Mehrotra, “ExecutingSQL over Encrypted Data in the Database-Service-ProviderModel,” Proc. ACM SIGMOD Int’l Conf. Management Data, June 2002.

[10] J. Li and E. Omiecinski, “Efficiency and Security Trade-Off inSupporting Range Queries on Encrypted Databases,” Proc. 19thAnn. IFIP WG 11.3 Working Conf. Data and Applications Security,Aug. 2005.

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

[12] D. Agrawal, A.E. Abbadi, F. Emekci, and A. Metwally, “DatabaseManagement 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 InformationSoc., Mar. 2011. FERRETTI ET AL.: DISTRIBUTED, CONCURRENT, AND INDEPENDENT ACCESS TO ENCRYPTED CLOUD DATABASES 445 Fig. 9. TPC-C performance (latency equal to 40 ms). Fig. 10. TPC-C performance (latency equal to 80 ms).

[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: ASecure Searchable Secret Sharing Scheme for Privacy PreservingDatabase Outsourcing,” Proc. Fifth Int’l Workshop Autonomous andSpontaneous Security, Sept. 2013.

[16] G. Cattaneo, L. Catuogno, A.D. Sorbo, and P. Persiano, “TheDesign and Implementation of a Transparent Cryptographic FileSystem For Unix,” Proc. FREENIX Track: 2001 USENIX Ann.Technical Conf., Apr. 2001.

[17] E. Damiani, S.D.C. Vimercati, S. Jajodia,S. Paraboschi, and P.Samarati, “Balancing Confidentiality and Efficiency in UntrustedRelational Dbmss,” Proc. Tenth ACM Conf. Computer and Comm.Security, Oct. 2003.

[18] L. Ferretti, M. Colajanni, and M. Marchetti, “Supporting Securityand Consistency for Cloud Database,” Proc. Fourth Int’l Symp.Cyberspace Safety and Security, Dec. 2012.

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

[20] H. Berenson, P. Bernstein, J. Gray, J. Melton, E. O’Neil, and P.O’Neil, “A Critique of AnsiSql Isolation Levels,” Proc. ACMSIGMOD, June 1995.

[21] “Xeround: The Cloud Database,” Xeround, http://xeround.com,Apr. 2013.

[22] “Postgres Plus Cloud Database,” EnterpriseDB, http://enterprisedb.com/cloud-database, Apr. 2013.

[23] “Windows Azure,” Microsoft corporation, http://www.windowsazure.com, Apr. 2013.

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

[25] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, “An IntegratedExperimental Environment for Distributed Systems and Networks,”Proc. Fifth USENIX Conf. Operating Systems Design andImplementation, Dec. 2002.

[26] A. Fekete, D. Liarokapis, E. O’Neil, P. O’Neil, and D. Shasha,“Making Snapshot Isolation Serializable,” ACM Trans. DatabaseSystems, vol. 30, no. 2, pp. 492-528, 2005.

[27] A. Boldyreva, N. Chenette, and A. O’Neill, “Order-PreservingEncryption Revisited: Improved Security Analysis and Alternative Solutions,” Proc. 31st Ann. Conf. Advances in Cryptology (CRYPTO’11), Aug. 2011.

[28] “IP Latency Statistics,” Verizon, http://www.verizonbusiness.com /about/network/latency, Apr. 2013.