Data Integrity and Confidentiality in Outsourced Database

Sonal Balpande
Ramrao Adik Institute of Technology
Navi Mumbai

Rajashree Shegde
Ramrao Adik Institute of Tech.
Navi Mumbai

Lata Ragha
Ramrao Adik Institute of Tech.
Navi Mumbai

ABSTRACT
An increasing number of enterprises outsource their IT functions or business processes to third-parties who offer these services with a lower cost due to the economy of scale. Quality of service has become a major concern in outsourcing. When database owners outsourced their data to service providers, which might be untrusted or compromised, two issues of data security emerge, data confidentiality and data integrity. Most of the previous research focuses on only one issue and the solution to integrate two approaches is expensive. Furthermore, no solution is raised on various character queries. We propose an approach which keeps data confidentiality and integrity. The proposed approach is based on bin checksum, which can be used to check correctness, completeness and freshness of multiple tuples at one time.

General Terms
Outsourced Database, Database Security

Keywords
Data Integrity, Data confidentiality, Outsourced database

1. INTRODUCTION
Database outsourcing, usually referred to as Database As a Service [1] the external service provider provides mechanisms for clients to access the outsourced databases. A major advantage of database outsourcing is related to the high costs of in-house versus outsourced hosting. Outsourcing provides significant cost savings and promises higher availability and more effective disaster protection than in-house operations. On the other hand, database outsourcing poses a major security problem, due to the fact that the external service provider, which is relied upon for ensuring high availability of the outsourced database (i.e., it is trustworthy), cannot always be trusted with the confidentiality of database content.

There are three main entities in the Outsourced Database (ODB) model: A User poses the query to the client. A Server is hosted by the service provider who stores the encrypted database. Client also maintains metadata for translating user queries to the appropriate representation on the server and performs post processing on server query result. It is assumed that the data owner may up-date the database periodically or occasionally, and that the data management and retrieval happens only at the servers.

Existing proposals in the data outsourcing area typically support data confidentiality or data integrity and it support limited types
of queries. Further no scheme support authentication on character data query. In order to solve above problems we propose the scheme which keep data confidentiality and mean while guarantee data integrity on numeric and character query**.**

The remainder of the paper is organized as follows section 2 gives an overview of the existing proposals. Section 3 presents the System Model for enforcing Data confidentiality and integrity in the outsourced database. Authentication of range aggregation and character based queries explained in section 4. Finally, section 5 concludes the paper.

2. LITERATURE SURVEY
Query Authentication should verify data correctness, completeness and freshness. Three signature-based approaches were proposed: tuple level signature, aggregated signature (AS) [2] and signature chaining [3]. In tuple level signature, each tuple is assigned with a signature and verified based on this signature. In AS, when t signatures $s_1; : : : ; s_t$ on t messages $m_1; : : : ;m_t$ signed by the same signer need to be verified all at once, certain signature schemes allow for more efficient communication and verification than t individual signatures.

In signature chaining, a signature is signed on three consecutive tuples, i.e. si =sign (ti−1|ti |ti+1). Note that $t_1. . . t_N$ are sorted on some query predicate. Two special records t_0 and t_{N+1} are added for the signature of tuple t_1 and t_N .

In authenticated data structures, tuples are organized into a tree such that one signature to the root node can guarantee the data integrity of other nodes in the tree. MHT is a main-memory binary index tree, where each leaf node contains the hash of a tuple, and each internal node contains the hash of the concatenation of its two children. Based on MHT, several disk-based dynamic variants have been proposed [4],[5].Since one MHT is built on one attribute, to support authentication of multi-dimensional range queries, multiple MHTs are required to be constructed.

Table 1: Overview of existing Model

	Confidentiality	Integrity	Queries		
			Range	Aggreg ation	Char acter
AS	_	√	√	_	_
MHT	_	√	√	_	_
SAE	_	√	√	_	_
BBA	√	√	√	√	_
CMCIS	√	_	_	_	√
DIC	√	√	√	√	√

SAE [6] separates authentication from query execution by exploiting trustworthy organizations with expertise on security issue. Such an organization is referred as a *trusted entity* (*TE*). Although a *TE* possesses up-to-date resources and know-how on security standards, cryptographic libraries, etc., it does not necessarily have the infrastructure to manage large databases and high query loads. Therefore, SAE assigns to the *TE* only the authentication process, which involves little computational effort compared to the actual query processing performed at the *service provider*. Clients issue queries directly to the *SP*, which sends back only the results.

Bucket based authentication [7] is based on bucket checksum, which can be used for the authentication of multiple tuples.

CMCIS [8] (Character Mapping Cipher Index Scheme), allows various key word and fuzzy queries. But does not provide integrity of data.

Table 1 gives an overview of existing model

To address the problems of existing approaches, we propose a new authentication approach:

1. It provides an integrated solution for data confidentiality and data integrity. Data integrity is checked at bin level instead of at tuple level. One bin has a unique identifier and consists of multiple tuples.
2. It provides efficient authentication for range queries and aggregation queries, including MIN, MAX, and COUNT queries.
3. It provide efficient authentication for character queries.
4. It provides the authentication of data freshness efficiently in dynamic scenarios, *incremental hash* [9] is used.

3. SYSTEM MODEL

3.1 System architecture & query processing

The system architecture has three entities are involved: user, trusted client and untrusted service provider.

The basic architecture and the control flow of a Database Service modeled using an encrypted database is shown in Fig. 1.

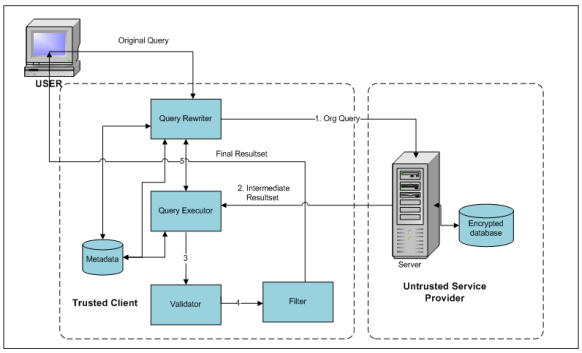

Figure 1.System architecture and query processing.

The system has three entities are involved: user, trusted client and untrusted service provider. A user poses the query to the client. A server is hosted by the service provider who stores the encrypted database. A client stores the data at the server. Client also maintains metadata for translating user queries to the appropriate representation on the server, and performs post-processing on server query results

The query is processed between trusted client and untrusted service provider interactively as follows:

Query Rewriter translates 'original query' into 'server query' according to the metadata at client, and sends the translated query to service provider for execution. Service provider executes 'server query' on encrypted data and returns the intermediate result set to *Query Executor* at client. *Query Executor* decrypts intermediate result set and sends the result set to *Validator*, which verifies data integrity and sends the result to *Filter*. *Filter* filters out unqualified tuples and attributes.

3.2 Confidentiality

A database can be encrypted according to different strategies. In principle, both symmetric and asymmetric encryption can be used at different granularity levels. Symmetric encryption, being cheaper than asymmetric encryption, is usually adopted. The granularity level at which database encryption is performed can depend on the data that need to be accessed. For balancing client workload and query execution efficiency, database is encrypted at tuple level. In cipher text table, attribute ETuple represents encrypted tuple. For tuple *ti* ETuple is calculated as follows: *Etuple(ti)=E(ti.TID||ti .BID||ti*.CBID||*ti .Name*|. . . |*ti .Salary y)k* ; where *E()k* is a symmetric encryption function with key *k*, and '|' is string concatenation.

While database encryption provides an adequate level of protection for data, it makes impossible for the server to directly execute the users' queries on the encrypted database. To allow the server to select a set of tuples to be returned in response to a query, BID (Bucket Identifier) and CBID (character bucket identifier) associated with the encrypted table.TID denotes tuple identifier. Bucket-based index was proposed by Hacigumus [10] and extensively studied in [11],[12],[13],[14]. Basically, a bucket is a *d*-dimensional rectangle built on numeric data with the following elements: (1) Bucket Identifier, denoted as BID; (2) the number of tuples in bucket, denoted as bucket size; (3) lower bound (left bottom corner) and higher bound (right top corner), denoted as *(L,H)*'

Table 2: Plaintext Table

TID	BID	CBID	NAME	AGE	CITY	SALARY
1	1	9090	Ash	28	Abad	2200
2	1	8187	Sonali	30	Mumbai	2500
3	1	9090	Anu	33	Abad	2800
4	2	8492	Pooja	27	Chennai	3500
5	3	9090	Aryan	45	Abad	4300

Table 3 Cipher text Table

BID	CBID	ETUPLE
1	9090	&65%*!@76&W#$UN)
1	8187	jy*bg&*!ecbrjydgtvr*%^ K*
1	9090	k@fid&*!@&khdyt*#%vr
2	8492	kh8krrh*!ehbrwj*jyu*!gyr
3	9090	y*nhk&*!yie7g^tkfd6*%^ &@

Table 4 Bucket Schema

BID	l_salary	h_salary
1	2000	3000
2	3000	4000
3	4000	5000
4	5000	6000

Cipher text table and bucket schema are constructed and illustrated in Tables 3 and 4, respectively. Character Identifier Table 6 store character identification index CID for character. In plaintext table, attribute BID is introduced to distinguish tuples of different buckets and CBID is formed by concating first character identification number of attribute Name and City respectively.

3.3 Authentication of range and character query

For 1-D range query an equi-width bucket is built on attribute salary and illustrated as attribute (*L*,*U*) of Table 4.Given table employee(TID, Name, Age, City, Salary) illustrated in Table I, an original query is issued as follows:

SELECT * FROM EMPLOYEE WHERE SALARY BETWEEN 2500 AND 3500.
Based on the Table 4 bucket schema, the original query is translated into the following server query:
SELECT * FROM EMPLOYEE WHERE BID IN (1,2).
For 2-d range query, considering data distribution characteristics in the multi-dimensional space, such as sparsity and clustering, a multi-dimensional partition is considered. An example of multidimensional bucket schema is given below.

Table 5 Two dimension bucket checksum

BID	l_age	h_age	l_salary	h_salary
1	27	33	2200	2800
2	25	38	3300	4000
3	44	50	4100	5000

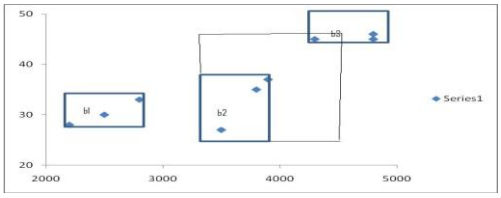

Figure 2.Multi dimension Partition

Given Table 2, a multi-dimensional range query is issued as follows:
SELECT * FROM EMPLOYEE
WHERE SALARY BETWEEN 3300 AND 4500 AND AGE BETWEEN 25 AND 45.

In Fig 2,query is depicted as a thin line rectangle and buckets are depicted as three real line rectangles. The multi-dimensional bucket schema is presented in Table 5, where attribute lower bound and upper bound of buckets is stored. Bucket 2 is fully covered and bucket3 is partially covered. It is required that all the overlapped buckets should be returned. So original SQL is transformed using Table 5 is as follows:
SELECT * FROM EMPLOYEE WHERE BID in (1, 2);
To authenticate Character keyword queries. First determine all the characters composed attribute. Then form character bucket identifier (CBID) by concating the character identifier shown in table 6 of first alphabet of character attribute. Let relation R={A_1,$A_{2.....}$} where A_1 and A_2 are character attributes. If A_1={a_1,$a_{2.....}$} andA_2={b_1,b_2....} then CBID is formed by concating CID(first character of a_1)||CID(first character of b_1).
For Character data query
SELECT * FROM EMPLOYEE WHERE CITY LIKE 'a%' AND SALARY > 4000

Original SQL is transformed using Table 5 and Table 6 is:
SELECT * FROM EMPLOYEE WHERE CBID LIKE '_ _90%' AND BID =3
Here CID for alphabet "a" is selected from character identifier Table 6.

Table 6 Character Identifier

Alphabet	CID	Alphabet	CID	Alphabet	CID
A	90	J	73	S	81
B	91	K	89	T	80
C	92	L	88	U	79
D	93	M	87	V	78
E	94	N	86	W	77
F	95	O	85	X	76
G	96	P	84	Y	75
H	97	Q	83	Z	74
I	98	R	82		

For query,
SELECT * FROM EMPLOYEE WHERE NAME='SONALI' AND CITY LIKE 'M%'
Is transformed as
SELECT * FROM EMPLOYEE WHERE CBID LIKE '8187'

3.4 Authentication of Aggregation Query

To improve authentication performance of aggregation queries, several rules are used, under which a certain percentage of buckets need not be authenticated. Buckets are divided into four classes: uncovered, inner partially covered, outer partially covered and fully covered.

For MIN query is assume the set of overlapped buckets with the query is: {b1. . .bk}. In step 1, scan all overlapped bucket for lowest bound. Such that if bucket is fully covered consider its lower bound and if bucket is inner partial covered consider its higher bound. Get the lowest of them. In step 2, ignored buckets whose lower bound is greater than the lowest bound found in step one.

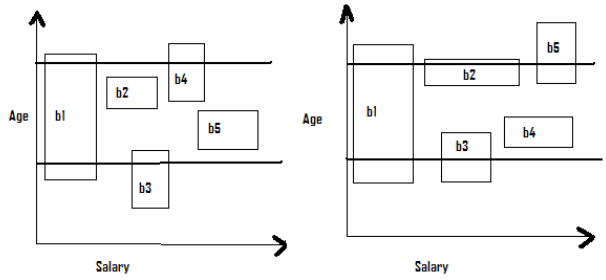

Figure 3 Aggregation (a)) candidate buckets: {*b*1,*b*2}; and (b) candidate buckets: {*b*1,*b*2,*b*3}.

Given MIN query against Table I:
SELECT MIN (salary) FROM EMPLOYEE WHERE AGE BETWEEN 25 AND 35,

The rule is shown in fig.3 (a). Here L_A (b2) is the lowest bound .So there must be tuple on left bottom which is less than any tuple on attribute salary in bucket b3,b4,b5 and can be ignored safely.b1 is outer bucket and lower bound of bucket is less than b2,so b1 is considered. So the candidate bucket set is {b1, b2}. The rule for MAX query is similar. In Figure 3(b), lowest bound of overlapped bucket is H_A (b3).Here b1 and b2 has lower bound less than the lowest bound found in step 1. So the candidate bucket set is {*b*1, *b2*, *b3*}.

4. QUERY AUTHENTICATION

Query authentication should verify three dimensions of the result set: correctness, completeness and freshness. Correctness means the result set originates from the data owner and has not been tampered with. Completeness means that no tuple has been deleted from the result set. Freshness means that the result set incorporates the latest updates from the data owner.

Query authentication is performed by *Validator* .In Figure 1, the input to *Validator* is a set of decrypted bin. To validate the data integrity of bin, *Validator* checks the following two aspects: inter-bin integrity and intra-bin integrity.

Inter-bin integrity checks whether all bins covered by the query are returned. Intra-bin integrity checks the following aspects: each bin is covered by the query; each tuple in the bin is correct; no invalid tuples are inserted or valid tuples are deleted from the bin. To verify intra-bin integrity, a new concept called *bin checksum* is introduced.

4.1 Checksum

Unique combination of Bucket identification (BID) and Character Bucket identifier (CBID) forms Bins (each row in Bin Checksum table is a unique Bin).

Table 7 Bin Checksum

BID	CBID	CHECKSUM	Count
1	9090	xybh*!ehbrwj*jyu*!yyr	2
1	8187	Rk!bg&*!ecbrjydgtr*th	1
2	8492	jk&*!@#k4JKs*w%gh	1
3	9090	bg&*!ecbrjydgtr*th*+j	1

Bin checksum is a digest of the Bin, which has the following characteristics:

1. Unique, which means each bin is assigned with a different bin checksum;
2. Sensitive to updates, which means any modifications to tuples in bin will generate a different bin checksum;
3. Efficient in calculation and storage.

The uniqueness of bin checksum is utilized to prevent substitution attack (one bucket is Substituted by another bucket). The sensitive to updates characteristic guarantees data freshness in dynamic scenarios. Assume that there are k tuples $t1. . . tk$ in bin bs . To calculate bin checksum is presented in Equation (1), where tuples are ordered by their primary key to guarantee the uniqueness of bin checksum:

$$\text{Checksum}\ (bs) = H(t1|. . . |tk) \quad \text{----------(1)}$$

Here, H is a collision-free hash function, which is utilized to achieve the uniqueness of checksum. Besides, hash is space compact. For example, MD5, a widely used hash function, takes variable length input and gets fixed length output, 16 bytes.

An efficient solution based on incremental hash. Incremental hash [9] was proposed to maintain hash incrementally instead of recalculating from scratch. The definition is as follows:

Let $x = x1 . . . xn$ be a sequence of blocks, H a collision-free hash function. If block x_i is modified to x_i^1 , the new hash of x can be recalculated as

$H(x) \otimes H(x_i)^{-1} \otimes H(xi^1)$, where $\otimes$ is an commutative, associative and invertible operation. Then we call H an incremental hash.

Assuming bin bi has k tuples set $\{T1, . . . , Tk\}$, the calculation of checksum_{incre} is as follows:

$$\text{Checksum}_{incre}\ (bi) = \prod_{j=1}^{k} (\text{H (Tj) mod q})$$

In Equation, q is a large prime with at least 512 bits in length. H is a full domain collision free hash function.

Example: At time $t0$, bucket bi has 5 tuples with T set: $\{T1, T2, T3, T4, T5\}$.

Incremental bucket checksum is calculated as

$$f0 = \prod_{j=1}^{5} (H(Tj) \bmod q).$$

At time $t1$, tuple T1 is updated to $T1^1$ then the bucket checksum is recalculated as

$$f1 = (f0 * H(T1)^{-1} * H(T1^1)) \bmod q.$$

At time $t2$, a tuple with $T30$ is inserted, the bucket checksum is recalculated as

$$f2 = (f1 * H(T30)) \bmod q.$$

At time $t3$, the tuple with $T3$ is deleted from the bucket, the bucket checksum is recalculated as

$$f3 = (f2 * H(T3)^{-1}) \bmod q.$$

Checksum_{incre} *is* much more efficient than checksum in maintenance cost Checksum_{incre} incurs no network overhead, whereas checksum incurs heavy network overhead.

5. CONCLUSION

There are two major issues of data security in outsourced databases: Data confidentiality and data integrity. Existing proposals in the data outsourcing area support data confidentiality or data integrity on limited types of queries. And, all the schemes are according to numeric data. We proposed an integrated solution to keep both Data confidentiality and integrity. Specifically, we proposed a confidentiality-preserving authentication approach which works on both numeric and character data. Compared with previous authentication approaches, our approach is efficient (provides authentication at bucket level) and effective (provides authentication for range, character and aggregation queries).

6. REFERENCES

[1] H. Hacigumus, B. R. Iyer, and S. Mehrotra, "Providing database as a service," In Proceedings of International Conference on Data Engineering (ICDE), 2002.

[2] E. Mykletun, M. Narasimha, G. Tsudik, "Authentication and Integrity in Outsourced Databases," ACM Transactions on Storage, 2(2):107–138, 2006.

[3] Pang H, Jain A, Ramamritham K, Tan K., "Verifying completeness of relational query results in data publishing," SIGMOD, Baltimore, MD, U.S.A., June 2005

[4] Devanbu, M. Gertz, C. Martel, S.G. Stubblebine, "Authentic Third-party Data Publication," In Proc. of the 14th Annual IFIP WG 11.3 Working Conference on Database Security, School, the Netherlands, 2000.

[5] Li F, Hadjieleftheriou M, Kollios G, Reyzin L. Dynamic authenticated index structures for Outsourced databases. SIGMOD, Chicago, IL, U.S.A., June 2006

[6] Stavros Papadopoulos, Dimitris Papadias, Weiwei Cheng, Kian-Lee Tan, "Separating Authentication from Query Execution in Outsourced Databases," 2009.

[7] Jieping Wang, Xiaoyong Du, Jiaheng Lu,.and Wei Lu1., " Bucket-based authentication for outsourced databases," Concurrency and Computation: Practical and Experience. 2010.

[8] Ning Wang, Wei Zhao, Ying Wang, Guohua Liu., "Character Mapping Cipher Index Scheme for Character Data in Outsourced Databases," International Conference on Computational Intelligence and Security, 2009

[9] Bellare M, Micciancio D., "A new paradigm for collision-free hashing: Incrementality reduced cost," Eurocrypt (Lecture Notes in Computer Science, vol. 1233). Springer: Berlin, 11–15 May 1997; 163–192

[10] Hacigumus H, Iyer B, Li C, Mehrotra S., "Executing SQL over encrypted data in the Database service provider modelling," ACM SIGMOD. Madison: Academic, 2002

[11] Damiani E, Vimercati SDC, Jajodia S, Paraboschi S, Samarati P., "Balancing Confidentiality and efficiency in untrusted relational DBMSs," *ACM Conference on Computer and Communications Security*, Washington, DC, U.S.A., October 2003.

[12] Jieping Wang, Xiaoyong Du., "Cluster Based Partition for Multi-dimensional Range Query in DAS Model," Eighth IEEE/ACIS International Conference on Computer and Information Science 2009

[13] Wang J, Du X. LOB: Bucket based index for range queries. WAIM, Zhangjiajie, China, 27–29 August 2008.

[14] LeFevre K, DeWitt D, Ramakrishnan R. Mondrian, "Multidimensional k-anonymity," *ICDE*, Atlanta, GA, U.S.A., 3–7 April 2006.