Provable Dynamic Data Possession by Datalog Rules

Jing Lu
School of Optical Electrical and Computing Engineering
University of Shanghai for Science and Technology, Shanghai, China, 200093
Email: jing.lu@usst.edu.cn

Ting Dou
School of Optical Electrical and Computing Engineering
University of Shanghai for Science and Technology, Shanghai, China, 200093
Email: ting.dou@foxmail.com

***Abstract*—In recent years, the concept of data outsourcing has become quite popular. With the appearance of Cloud Computing, there is a new paradigm: Cloud Storage. Cloud Storage means that the data owner moves its data to a Cloud Storage Server that is supposed to faithfully store the data and make it available to the data owner and perhaps the data requestors. This paper discusses how to guarantee the Provable Data Possession when data dynamics are supported. Different integrity granularities are supported including Size Integrity, Block-Number Integrity and the Block-neighborhood integrity which is realized by rules defined in Datalog through neighborhood relationship between data blocks. All integrity checking runs as a service in a Third Party Auditor. The whole architecture is implemented above Hadoop, which gives the first experience of a trustable cloud data service platform.**[1]

***Index Terms*—Cloud Computing; Provable Data Possession; Third Party Auditor; Data Dynamics; Datalog.**

I. INTRODUCTION

Data outsourcing, together with Cloud Data Storage, is becoming more and more popular and at the same time prompts a number of interesting security issues. These issues have been extensively investigated in the past few years [1] [2] [3]. However, Provable Data Possession (PDP) is a topic that has only recently been noticed by the research field [4][5]. The main issue is how to efficiently and securely verify that a Cloud Storage Server is faithfully storing the client's outsourced data that is normally very large.

A typical network architecture for Cloud Data Storage is illustrated in Fig. 1. Three different network entities can be identified as follows [4] [5]:

- Client: an entity, which has large data files to be stored in the cloud and relies on the cloud for data maintenance and computation, can be either individual consumers or organizations;
- Cloud Storage Server (CSS): an entity, which is managed by Cloud Service Provider (CSP), has significant storage space and computation resource to maintain clients' data;
- Third Party Auditor (TPA): a TPA, which has expertise and capabilities that clients do not have, is trusted to assess and expose risk of cloud storage services on behalf of the clients upon request.

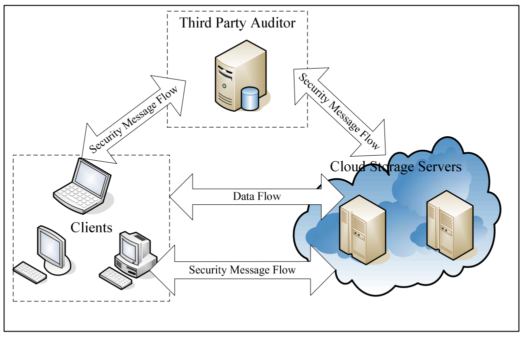

Figure 1. Cloud Data Storage Architecture

The Cloud Storage Server is assumed to be untrusted both in security and reliability. It might maliciously or accidentally erase hosted data; change hosted data, make the query slow, etc. The problem becomes even more serious if the client has a small computing device with limited resources. Prior work has addressed this problem using either public key cryptography or requiring the client to outsource its data in an encrypted form [6] [7]. However, storing data in an encrypted form limited the query flexibility [8][9]. Fuzzy queries cannot be executed above encrypted data [10][11]. Meanwhile, encryption and decryption need extra computing cost [12] [13][14].

Furthermore, most of the previous work focuses on data query, which is static. In Cloud Computing, the remotely stored electronic data might not only be retrieved but also modified by the clients or some other

[1]This work is supported by Innovation Program of Shanghai Municipal Education Commission (12YZ095), and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry.

data users, e.g., through block insertion, deletion and modification. Unfortunately, the state-of-the-art in the context of remote data storage mainly focuses on static data files and the importance of this dynamic data updates has received limited attention in the data possession applications so far [6] [7] [8] [9].

Moreover, the direct extension of the current Provable Data Possession (PDP) schemes to support data dynamics may lead to security loopholes [15][16][17]. Although there are many difficulties faced by researchers, it is well believed that supporting dynamic data operation can be of vital importance to the practical application of Cloud Data Storage. In view of the key role of public verifiability and the supporting of data dynamics for cloud data storage, we present a service, which runs in the Third Party Auditor (TPA) [18] and which write the data integrity rules in Datalog. In case clients do not necessarily have the time, feasibility or resources to monitor their data, they can delegate the monitoring task to a trusted TPA. For application purposes, the clients may interact with the cloud servers via Cloud Storage Server (CSS) to access or retrieve their pre-stored data [19][20]. The rules can be deduced to produce new rules.

This paper makes the following contributions:

- We introduce a method to partition cloud data into blocks and then tag them according to a coordinate system;
- We insert random numbers between blocks using watermarking technology to strengthen the data security;
- We define different integrity granularity levels and thus speed up the integrity checking procedure;
- We introduce how to describe data integrity rules in declarative Datalog rules through the meta-data and logs stored in TPA;
- We propose a provable data integrity service running in TPA, where the declarative rules are stored and the PDP is issued;
- We describe the whole procedure of Provable Data Possession service. The whole integrity service is implemented above Hadoop.

The remainder of this paper is organized as follows: Section 2 gives the definition of data dynamics. Section 3 is about the architecture of the remote data integrity checking service running in TPA. How to partition and tag blocks and how to generate new neighborhood relationships between data blocks after modification are also introduced in Section 3. The implementation and evaluation of this approach is detailed in Section 4. Section 5 gives the related work and finally Section 6 draws the conclusion.

II. Supported Data Dynamics

In practical scenarios, the clients may frequently perform block-level operations on the data files. The most general forms of these operations we consider in this paper are insertion, deletion and modification, which we call Data Dynamics [16][21].

A. *Data Insertion*

A general form of data insertion refers to inserting new blocks after some specified positions in the data file F.

Among the Data Insertion we define another special operation, which is Data Appending. Data Appending means the user insert new blocks at the end of the whole file.

B. *Data Deletion*

Data Deletion is just the opposite operation of Data Insertion. For single block deletion, it refers to deleting the specified block and moving all the latter blocks one block forward.

For continuous block deletion, it refers to deleting the speficied blocks and moving all the latter blocks forwards. Suppose the user wants to delete some blocks continuously, he must specified the beginning block number *i* and the last block number *j*. Therefore, the latter blocks will be moved *j-i+1* blocks forward.

C. *Data Modification*

Data Modification is one of the most frequently used operations in cloud data storage. A basic data modification operation refers to the replacement of specified blocks with new ones.

III. Dynamic Data Operation with Integrity Assurance

A. *Architecture*

Fig. 2 shows the processing procedure of integrity checking service running in TPA.

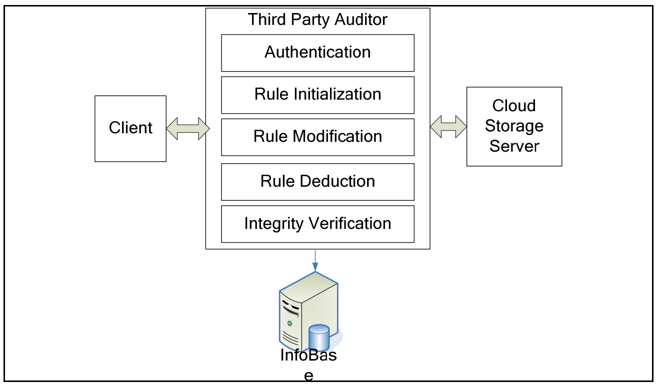

Figure 2.Verification Service in TPA for Remote Integrity Checking

Our approach runs as a service in TPA. First we begin with partitioning and tagging data blocks. Then random numbers are generated and inserted between data blocks using watermarking technology. Then the Meta-data of the outsourced files is stored in the TPA. When the data modification happens, the updating log will be stored in the TPA too. TPA will generate the integrity rules in Datalog using the neighborhood relationship between data blocks. Also, when the data modification happens, the TPA will generate the newest integrity rules according to the updating log. The TPA will verify the Provable Data Possession periodically using the Meta-data and Datalog rules. There is a InfoBase running in TPA which is shown in Fig. 3. Details will be discussed in this section.

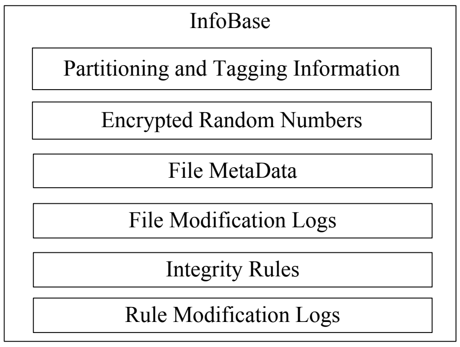

Figure 3. Rule Base Architecture

B. Partitioning and Tagging Data Blocks

In Cloud Data Storage, the outsourced data is normally very large. A coordinate system is designed here to partition data. In our coordinate system, every data block is partitioned into smaller blocks. Each block is tagged with a horizontal coordinate and a vertical coordinate. Both record the relative position of each block. Therefore, each block has its neighbors including: left neighbor, right neighbor, upper neighbor and down neighbor. We use logical (datalog) statements to capture these different neighbor relationships of information that are typically considered in isolation. Knowledge can be communicated, replicated, queried, updated, and monitored.

Suppose there are m blocks in horizontal coordinate and n blocks in vertical coordinate. We define a two-dimensional array to record the position of each block: *BLOCK [i][j] (0<i<=m; 0<j<=n)*.

The procedure to partition and tag the blocks is described in the following algorithm:

Algorithm01: Partitioning and tagging the blocks

```
Scanning the outsourced file;
Generate the horizontal and vertical coordinate;
Tagging the blocks with horizontal and vertical
coordinate;
```

The algorithm to generate neighbor relationship is shown in the following algorithm:

Algorithm02: Generating Neighborhood Relationships

```
For (i=0; i<=m; i++)
  For (j=0; j<=n; j++)
HorizontalNeighbor(BLOCK[i][j],BLOCK[i+1][j]);
VerticalNeighbor(BLOCK[i][j],BLOCK[i][j+1]);
```

The tagging of each block and the generated neighbor relationship are stored in InfoBase. This procedure is done in Rule Definition module.

The data is in an integrity state if and only of all the data is stored as if it is still in the client's side. That means, there are no macilious destroyments of the data blocks. Also, integrity required that the Cloud Data Storage does not change the relative position of each block. If the TPA finds from the sampled data blocks that the neighborhood relationship is destroyed, e.g., there are missing neighbors, it means that the file is no longer in an integrity state.

C. Inserting Random Numbers between Neighbors

In order to strengthen the security, we generate the random numbers and insert them between the blocks using the watermarking technology. The randon function is kept secret to Cloud Storage Provider and all the numbers are stored in TPA. When we want to check the data integrity of the file, we will sample the blocks from the file and the random numbers will be fetched out, recovered and compared with the random numbers stored in TPA. If the block is destroyed the random numbers will be destroyed too and thus the illegal action will be detected.

Compared with the size of the outsourced file, the random numbers only need limited space to be stored.

D. Meta-Data and Updating Log Stored in TPA

The Third Party Auditor is responsible for the storage of Meta-data of the outsourced files. In our approach, the integrity is checked according to the Meta-data. Meta-data is "data about data" [22]. We use the Meta-data to describe the outsourced file and it records the information about the size of the file, the number of blocks,the block neighborhood relationship and the inserted random numbers. When legal Data Insertion, Data Deletion or Data Update happens, the corresponding Meta-data will be updated in the Third Party Auditor so that the Meta-data is always correct. All the Data Insertion, Data Deletion or Data Update behaviors will be logged and the log will be kept also in Third Party Auditor.

There are altogether two possibilities which will cause the data modification: (1) Data destroyed by the Cloud Service Provider; (2) Data modified by the user. In these two possibilities, the first one is illegal and must be detected. The second one is legal but must be recorded too. In order to record the legal data modification we log all the data modification and the log is stored in TPA for auditing.

Considering support of frequent data modification we use a link data structure to store each log of the modification behavior. Each outsourced file has a corresponding Log Link which is composed of many Log Records. Each Log Record expresses one modification behaviour and including the following information:

(1) ***File_id***: the identification of the outsourced files. Each file has a unique id in TPA;
(2) ***Log_id:*** the identification of the log. Each modification has a unique Log_id in TPA.
(3) ***Modification_type***: there are altogether five types: Insertion, Appending, One-block-deletion, Continuous-block-deletion, Update;
(4) ***Affected_block_head:*** the first affected block;
(5) ***Affected_block_tail***: the last affected block;
(6) ***Time:*** when the file is modified;
(7) ***Last_log_id***: point to the most recent log record which has the same File_id.

When the outsourced file is updated frequently, if we want to check the integrity of the file, we first find the Log Link of this file, then we retrieve the whole link and generate the affected blocks so as to calculate what the data blocks should be now. Using the Log Link we don't have to search in the whole log space but we only search the log related to the outsourced file.

E. Integrity Checking Granularity

In order to save integrity checking time, we define different checking granularities including Size Integrity, Block Number Integrity and the Neighborhood Integrity.

The most coarse-grained integrity is the ***Size Integrity***. The TPA will ask the CSP to provide the size of the outsourced file. If the size matches the value stored in the TPA, we say that the outsourced file is in Size Integrity.

The second level integrity is the ***Block Number Integrity***. The TPA will ask the CSP to provide the number of the blocks in which the outsourced file is partitioned. The returned result first accepts the even-odd check. If it could not pass the check, that means the block number is wrong, then the block number integrity is destroyed. If it could pass the check, then it will accept the concrete number check.

The most fine-grained integrity is the ***Neighborhood Integrity***. Each block has its original neighborhood according to our partitioning and tagging algorithm. After the file is outsourced, the neighborhood relationship should not be changed. Only if the legal Data Insertion, Data Deletion or Data Update happens, the neighborhood relationship will be changed. The neighborhood relationship information is also stored in TPA as meta-data. The TPA will ask the CSP to provide the sampled continuous blocks. The TPA will generate neighborhood relationship of the sampled blocks and the result will be compared with the neighborhood relationship stored in TPA. Only if all the sampled block neighborhood relationships are consistent, we consider the blocks outsourced in CSP are in Neighborhood Integrity.

F. Verification Process

Integrity Description Library Establishment: The first step is to establish the integrity description library. The integrity description is given using the characteristic vectors. Each outsourced file is described in the following vectors: the file size, the size of each block in which the file is partitioned, and the number of the blocks. All the description information is encrypted and stored in TPA where the library is stored and maintained.

Periodical Verification Process:

The TPA will periodically ask the CSP to provide the file size, the number of blocks, and the sampled blocks. With the information, the TPA will compare with the Meta-data and Updating Log in the following steps:

Step 1: Size match. If the file size is equal to the file size in TPA then the file is in Size Integrity;

Step 2: Even-odd checking of block numbers. This procedure is designed to speed up the block number check. If the number cannot pass the even-odd check, the block number is wrong. The block number integrity cannot be ensured.

Step 3: Block number checking. After the block number passes the even-odd check, the number will accept the concrete number check. Again, the number is compared with the block number stored in TPA. If they are the same, the file is in block number integrity.

Step 4: Neighborhood checking. The TPA will sample some blocks from the file and check the neighborhood relationship. This will be detailed in the following sections.

Step 5: Random number checking. After the file passes the Neighborhood checking, the watermarked random number will be fetched out, recovered and compared with the number in TPA. Only if it is consistent with the original numbers, we consider the data blocks are in a consistent state.

G. Integrity Rules described in Datalog

Datalog is a truly declarative logic programming language that syntactically is a subset of Prolog. It is often used as a query language for deductive databases: it has a cleaner syntax and is more expressive than SQL. In recent years, Datalog has found new application in data integration, information extraction, networking, program analysis, security, and cloud computing.

Datalog is introduced by showing a sequence of interactions with a Datalog interpreter. The interpreter is started with the datalog command.

H. Integrity Assurance

The Third Party Auditor periodically checks the data blocks stored in the Cloud Data Storage Server, download them and take some sample blocks to test the integrity. Algorithm 02 is used to generate the neighbor relationship of sampled data blocks. Then we will use the following query in Datalog to find the neighbor relationship fact:

```
> HorizontalNeighbor(A, B)?

HorizontalNeighbor(block[12][3],block[13][3]).
HorizontalNeighbor(block[11][3],block[12][3]).
```

If the one of the Horizontal Neighbors is missed, we can judge that the Cloud Data Storage Server deletes this missing block. The Vertical Neighbors can also be checked using the same method.

I. Data Insertion

When there is a data block insertion, we will not change the tagging of each block after the inserted one. We only record the neighbor relationship. For example, BLOCK[100][100] is inserted after BLOCK[12][3]. Then the following neighbor relationship fact will be generated:

```
HorizontalNeighbor(block[12][3],block[100][100]
).
HorizontalNeighbor
(block[13][3],block[100][100]).
VerticalNeighbor(block[13][2],block[100][100]).
VerticalNeighbor(block[13][4],block[100][100]).
```

The corresponding rest blocks should also be adjusted and the new neighbor relationship will also be generated. The modification of rules is stored in Rule History.

J. *Data Deletion*

When there is a data block deletion, we will not change the tagging of each block after the deleted one. We only record the neighbor relationship modification. For example, BLOCK[2][3] is deleted. Then the neighbor relationship of BLOCK[2][3] will be deleted. We will use the follow sentence to find all the neighbor relationship of BLOCK[2][3]:

```
> HorizontalNeighbor(BLOCK[2][3], B)?
HorizontalNeighbor(BLOCK[2][3], BLOCK[1][3]).
HorizontalNeighbor(BLOCK[2][3], BLOCK[3][3]).

> VerticalNeighbor(BLOCK[2][3], B)?
VerticalNeighbor(BLOCK[2][3],BLOCK[2][2]).
VerticalNeighbor(BLOCK[2][3],BLOCK[2][4]).
```

Then these four relationship facts will be deleted and the following will be added to InfoBase.

```
HorizontalNeighbor(block[2][2],block[2][4]).
VerticalNeighbor(block[1][3],block[2][4]).
VerticalNeighbor(block[3][3],block[2][4]).
```

The corresponding rest blocks should also be adjusted and the new neighbor relationship will also be generated. The modification of rules is stored in Rule History.

K. *Data Modification*

Data modification include two procedures: to delete the block, and to insert a new block in the position. The processing of the rule modification can be divided into two parts: deletion and insertion.

IV. Implementation and Evaluation

We establish the Cloud Computing environment above Hadoop and simulate the Cloud Data Storage Server. We use IRIS Reasoner [23] (Integrated Rule Inference System), which is an extensible reasoning engine for expressive rule-based languages to evaluate queries over a knowledge base. The neighborhood relationship between different data blocks is stored in the knowledge base and in our system in the InfoBase. The facts describe the neighborhood relationship of the data blocks. The rules describe when the data blocks are changed, how to adjust the neighborhood relationship. The combination of facts, rules and queries is known as a logic program and forms the input to a reasoning task.

By using IRIS reasoning system, we build up the InfoBase and by using the API we set up the interface to initialize the neighborhood relationship, the integrity checking interface to find the neighborhood relationship of the sampled data blocks and compare whether there are missing blocks.

The whole system gives us the first experience that the clients are freed from the heavy work to check the integrity of data blocks. The integrity checking is running as a service in TPA so that it is easy to maintain the InfoBase and the reasoning procedure.

V. Related Work

Recently, much of research interest has been focused in the context of remotely stored data verification.

[4] defines the "Provable Data Possession" (PDP) model for ensuring possession of files on untrusted storages. It utilizes RSA-based homomorphic tags to audit the outsourced data, thus can provide public verifiability. Unfortunately, dynamic data storage is not considered. If the scheme from static data storage is extended directly to dynamic case, there are many design and security problems. In [15] [24], they propose a dynamic version of the prior PDP scheme. However, the number of queries is limited and fully dynamic data operations is not supported, i.e., block insertions cannot be supported. In our work, we support fully data dynamics including block insertion, block deletion and block modification. Our scheme fits not only for static data verification but also for dynamic data verification.

[16] was the first to explore constructions for dynamic provable data possession. They extend the PDP model in [4] to support provable updates to stored data files using rank-based authenticated skip lists. This scheme is essentially a fully dynamic version of the PDP solution. In particular, to support updates, especially for block insertion, they try to eliminate the index information in the "tag" computation in [4]. To achieve this, before the verification procedure, they employ authenticated skip list data structure to authenticate the tag information of challenged or updated blocks first. However, using extra "tags" in the outsourced data decreases the space efficiency. To some degree, the efficiency of query is also decreased. In our approach, all the neighborhood relationships are stored in TPA, which does not affect the storage efficiency of the outsourced data and the queries over it.

[24][25] considers dynamic data storage in distributed scenario, and the proposed challenge-response protocol can both determine the data correctness and locate possible errors. Similar to [15], they only consider partial support for dynamic data operation while our approach support full support to data dynamics.

[26][27] presents a cooperate PDP scheme based on Homomorphic Verifiable Response and Hash Index Hierachy based on multi-prover zero-knowledge proof system. This system supports Multi-cloud environment and introduces low computation and communication overheads in comparison with non-cooperative schemes. Still, this job only supports data statics while ours supports both data statics and data dynamics.

VI. Conclusion

In this paper, we discussed how to prove Data Possession by a Third Party Auditor when the outsourced data supports dynamics. We developed two algorithms to partition data into blocks and each block is given a vertical and horizontal coordinate. The neighborhood relationship is generated and stored in the InfoBase. When the data dynamics happen, the InfoBase is adjusted responsively. The TPA will check the outsourced data periodically through the meta-data, the updating log and the neighborhood relationship. The whole integrity checking service is implemented by many user-interfaces. The service gives the data owner a flexible and economical way to ensure the outsourced data integrity.

REFERENCES

[1] F. Sebe, J. Domingo-Ferrer, A. Martinez-Balleste, Y. Deswarte, and J.-J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," IEEE Transactions on Knowledge and Data Engineering, Vol.20, No.8, pp. 1034-1038, 2008.

[2] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generation Computer Systems, vol. 25, no. 6, pp. 599-616, 2009.

[3] J. Ju, J. Wu, J. Fu, Z. Lin, and J. Zhang, "A Survey on Cloud Storage", Journal of Computers, Vol 6, No 8, pp.1764-1771, 2011.

[4] J. Wu, Q. Shen, T. Wang, J. Zhu, and J. Zhang, "Recent Advances in Cloud Security", Journal of Computers, Vol 6, No 10, pp. 2156-2163, 2011.

[5] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: Multiple-Replica Provable Data Possession," in ICDCS '08: Proceedings of the 2008 The 28th International Conference on Distributed Computing Systems, (Washington, DC, USA), pp. 411–420, IEEE Computer Society, 2008.

[6] Davida GI, Wells DL, Kam JB. "A database encryption system with subkeys", ACM Transactions on Database Systems, vol.6, no.2, pp. 312-328, 1981.

[7] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data", In: Proceedings of 2000 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, pp. 44-55, 2000.

[8] H. Hacigümüs, B. Iyer, S. Mehrotra S, and C. Li, "Executing SQL over encrypted data in the database service provider model", In: Proceedings of the ACM SIGMOD Conference, ACM Press, pp. 216-227, 2002.

[9] G. Özsoyoglu, D. Singer, and S. Chung, "Anti-Tamper databases: Querying encrypted databases", In Proceedings of the 17th Annual IFIP WG 11.3 Working Conferene on Database Applications and Security, pp. 133-146, 2003..

[10] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data", In Proceedings of the ACM SIGMOD Conference, ACM Press, pp. 563-574, 2004..

[11] M. Kantarcioglu, and C. Clifton, "Security issues in querying encrypted data", In Proceedings of the IFIP Conference on Database and Applications Security, LNCS 3654, pp. 325-337, Heidelberg, Berlin: Springer-Verlag, 2005.

[12] J. Li, and E. Omiecinski, "Efficiency, security trade-off in supporting range queries on encrypted databases", In Proceedings of the IFIP Conference on Database and Applications Security, LNCS 3654, pp. 69-83, Heidelberg, Berlin: Springer-Verlag, 2005.

[13] S. Chun, and G. Ozsoygolu, "Anti-Tamper databases: Processing aggregate queries over encrypted databases", In Proceedings of the 22nd International Conference on Data Engineering Workshops, LNCS 4127, pp.89-103, Heidelberg, Berlin: Springer-Verlag, 2006.

[14] T. Ge, and S. Zdonik, "Answering aggregation queries in a secure system model", In Proceedings of the 33rd International Conference on Very Large Data Bases, ACM Press, pp. 519-530, 2007.

[15] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in SecureComm '08: Proceedings of the 4th international conference on Security and privacy in communication netowrks, (New York, NY, USA), pp. 1–10, ACM, 2008.

[16] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in CCS '09: Proceedings of the 16th ACM conference on Computer and communications security, (New York, NY, USA), pp. 213–222, ACM, 2009.

[17] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in 14th European Symposium on Research in Computer Security, pp. 355–370, Springer Berlin / Heidelberg, September 2009.

[18] P. Devanbu, M. Gertz, C. Martel, and S. Stubblebine, "Authentic third-party data publication," in IFIP DB, Sec'03, also in Journal of Computer Security, Vol. 11, No. 3, pp. 291-314, 2003, 2003.

[19] K. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability: Theory and implementation," Cryptology ePrint Archive, Report 2008/175, 2008.

[20] E.-C. Chang and J. Xu, "Remote integrity check with dishonest storage server," in Proc. of ESORICS'08. Berlin, Heidelberg: Springer-Verlag, pp. 223–237, 2008.

[21] A. Oprea, M. K. Reiter, and K. Yang, "Space-efficient block storage integrity," in Proc. of NDSS'05, 2005.

[22] Y. Xiao, G. Xu, Y. Liu, and B. Wang, "A Metadata-driven Cloud Computing Application Virtualization Model", Journal of Computers, Vol 8, No 6, pp. 1571-1579, 2013.

[23] http://www.iris-reasoner.org

[24] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in CCS '07: Proceedings of the 14th ACM conference on Computer and communications security, (New York, NY, USA), pp. 598–609, ACM, 2007.

[25] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song, "Remote data checking using provable data possession", ACM Transactions on Information and System Security, Volume 14 Issue 1, pp. 1-34, 2011.

[26] Y. Zhu, H. Wang, Z. Hu, G. Ahn, H. Hu, and S. Yau, "Efficient provable data possession for hybrid clouds", In Proceedings of the 17th ACM conference on Computer and communications security (CCS '10). ACM, New York, NY, USA, pp. 756-758, 2010.

[27] Y. Zhu, H. Hu, G. Ahn, and M. Yu, "Cooperative Provable Data Possession for Integrity Verification in Multicloud Storage", IEEE Transactions on Parallel and Distributed Systems, Volume 23, Issue 12,pp.2231-2244, 2012.