Article can be accessed online at http://www.publishingindia.com

In-Line Auditing and Real-Time Lineage Summaries to Maintain Ownership of Information Stored in Cloud Servers

Hiroshi Fujinoki*

Abstract

We propose new security architecture to enhance direct control to the information stored in cloud servers. It splits the cloud stack to two layers and having the security control for the owner of the information between them. By executing security-critical operations at the inline owner agent, the owner of the data logically preserves the essential security control to its data physically stored in a private cloud. The shadow auditor monitors the integrity of information stored in a cloud server to detect unauthorized modification of the information even by the administrators in the clouds while real-time lineage summaries provide cloud users timely feedback on the quality of data without disturbing their workflow. Our performance evaluations showed that real-time lineage summaries are effective for feed-backing quality of information for systems that have frequent references to the information. The shadow auditor was also workload scalable, while the major bottleneck was securing communication between the split cloud and the owner agent. The proposed security architecture will be a solution to make secure transition to clouds while the advantages of clouds are maintained.

Keywords: Cloud Security, Data Lineage, Information Assurance, Information Quality, Private Clouds

1. Introduction

The concept of cloud computing began to attract attention from IT industry in the late 90's. It integrates existing concepts for enhancing usability and efficiency of computer systems, such as utility computing, on-demand computing, and ubiquitous computing. Since then, the trade-off between its cost effectiveness, mainly due to economies of scale, and its security risks, has always been one of the top issues IT system managers must consider before their transition to clouds. Its major security risks include possible information leak, loss of the information stored in a cloud, and unauthorized modification of information by malicious users (Srinivasamurthy, 2010), (Zhang, 2010). Despite such risks, cloud computing has been widely adopted and it has been one of the hottest topics in IT innovations in the last decade.

While the private industry section took a lead in adoption of cloud computing, that trend is propagating to non-profit and government sections, mainly because of the recent heavy pressure to reducing the cost of their information systems. NCES, GES, Air Force's JBI and JBMC2, Navy's FORCEnet, and Army's FCS are examples of such projects (Paul, 2005). On the other hand, the security risks pose serious threats to some of such organizations.

A solution to reduce the security risks is private clouds, where a cloud infrastructure is accessible only to the users in the organization that owns the cloud (Mell, 2011). Such private clouds are owned by a single organization, where multiple subordinate units in the organization can share the same private cloud infrastructure to exploit the economy of scales. Private clouds reduce security risks by limiting accesses to both the hardware and software resources only by a well-defined and controlled group of users.

Many of government organizations have subordinate units, each of which owns and maintains its own IT system. For such organizations, the possible reduction in the cost of their IT systems using private clouds will be significant while sharing the hardware and software resources by trusted users significantly enhances security.

* Associate Professor, Southern Illinois University Edwardsville, United States. E-mail: hfujino@siue.edu

Another advantage in using a private cloud is information sharing, which effectively eliminates redundant information. In legacy information systems, each organization unit builds its own information system. In such systems, redundant information has been a serious problem since duplicated information in multiple units causes inflation to the storage capacity requirement. Whenever a unit "imports" information from another unit, it results in duplicated copy at the importing unit. Moreover, if the information duplicated to multiple units is frequently updated, the integrity of the information will be a serious issue.

Although private clouds significantly reduce the risk of information leak while they realize the benefits of cloud computing, private clouds still have some weaknesses. Our informal interviews with the IT officers in a government organization revealed that there are three major concerns that prevent their transition to clouds:

(a) Loss of direct control to information: Adoption of a cloud entails separation of the organization units who own information from those who maintain it. This implies that cloud users will lose their direct control to the information they own. This becomes a serious problem if the users are not provided sufficient feedback from the cloud provider regarding how their information is maintained and shared with other parties in a cloud server. Loss of direct control to the information is a serious concern especially when crimes by insiders in cloud providers tend to increase (NIST, 2011).

(b) Lack of mechanism for information fusion: For many large-scale organizations, each organization unit often has its own formats of the data to represent information, which are not compatible with those in other units. In legacy systems, data is transformed to the formats of the importing units semi-manually using one-way format conversion tools before the imported data is used. If the imported data must be shared by other units after it is modified by a unit that imported the data, another round of format conversion is necessary, which significantly decreases the organization's productivity. If information stored in a private cloud should be shared to eliminate redundant copies of the data, a new mechanism to dynamically absorb the incompatible data formats is necessary.

(c) Lack of security protection against unknown bugs: Any unexpected program bugs in virtualized cloud servers may leak highly confidential information to unauthorized users. As described by Ristenpart (Ristenpart 2009), processes in an existing virtual environment in a cloud could leak confidential information to other processes running in the same physical computer. As a result, confidentiality of any information processed in such clouds will never be guaranteed.

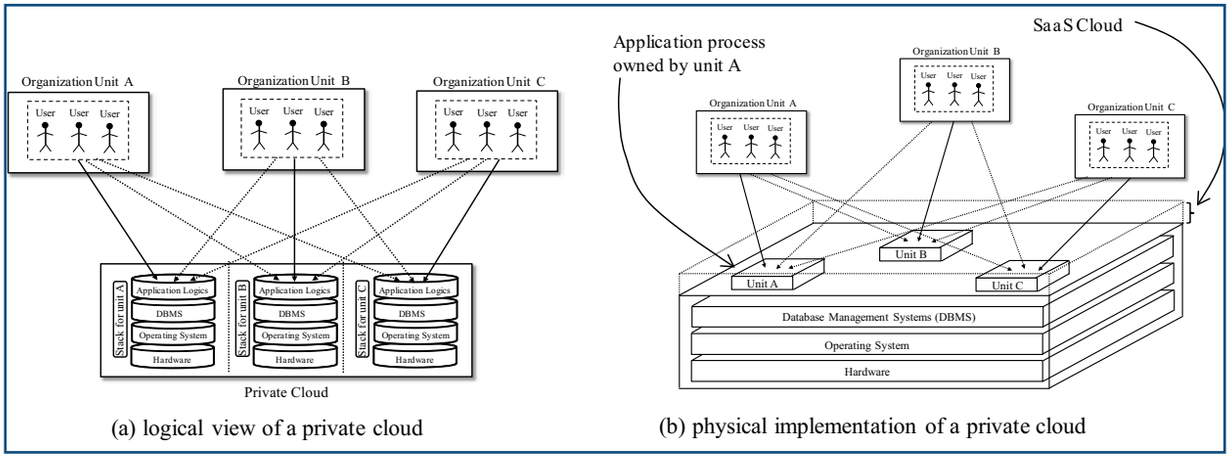
To cope with the loss of direct control to the information owned by each unit, we designed Split Cloud. The core of Split Cloud consists of the inline shadow auditor and real-time lineage summaries. Shadow auditor automatically attests the integrity of the data stored in a private cloud, while the real-time lineage summaries provide feedback on data quality to human users in real-time with its overhead less than a level of delay that does not disturb workflow of the users. The lack of mechanism for information fusion is addressed by Split Cloud by having a layer that identifies the owner unit of data being used and transparently performs transformations of the data formats "on line".

An effective way to prevent the problem of unknown bugs is not to use virtualization. However, preventing the problem for virtualized cloud environments is quite hard mainly because it is impossible to predict all the possible bugs in the implementation of virtual environment. A possible solution is encryption of information before it is stored in the cloud as Sahai (Sahai, 2008) and Bain (Bain, 2011) suggested, but the operations the cloud side DBMS can perform to the encrypted information are significantly limited. Because of the reasons, we took a reactive approach to this problem, by detecting illegal updates by such attacks as soon as they happen.

As a model for our proposed security architecture, we assume a large government organization, such as US DoD, which consists of Army, Marine, and Navy. The information they produce and use is stored in a private cloud that is conditionally shared by the three units. The service model we assume is SaaS, in which the three units have accesses to their information management systems (e.g., DBMS) provided and set up by the central data center who runs the private cloud for all of the three units.

Figure 1. shows (a) the logical view and (b) physical implementation of the private cloud assumed in this

Figure 1. (a) Logical View and (b) Physical Implementation of a Private Cloud



work. Each of the three units has their own stack in a private cloud (Figure 1. (a)). It is the system each unit logically owns in the private cloud (indicated by a solid line). Each unit may use the information owned by other units (indicated by a dashed line). In (b), the underlying system hosts DBMS, operating systems, and underlying hardware, which serve to all the three units.

The rest of this paper is organized as follow. Section II describes the existing work related to the proposed security architecture. The existing work is described and analyzed especially by identifying what are still missing in the existing work. Section III describes Split Cloud, focusing on how the missing security protections in the existing solutions are addressed by Split Cloud. Section IV presents the performance evaluation of Split Cloud from the viewpoint of its overhead, followed by Section V, which describes the conclusions and future work.

2. Existing Work

Data lineage is a technique that has been popularly used for assuring information quality in databases ((Groth, 2004), (Tsai, 2007), (Moitra, 2009)). Data lineages are essentially high-level access logs, which are stored as "metadata" for each tuple in a table. Data lineage keeps track of the origin and access history of each tuple stored in database tables by recording how data is created and modified in the past. Each such activity is recorded as a "lineage record".

Although it is quite similar to access logs, data lineage is rather high level metadata that represents quality in information. There are two concerns about the effectiveness in data lineage. The first concern is that data lineage is vulnerable to insiders' malicious accesses since data lineage is performed and stored in the server side as described in section 3.2 (Problem 1).

The second problem is the usability in data lineage. For tuples that have been updated frequently, there will be a large number of lineage records associated to such tuples. Data lineage will be effective in providing detailed feedback about the quality of the information, especially when users have some specific doubts about the accuracy of data, but may not be effective in providing the users with quick but intuitive feedback mainly due to the volume of the lineage records associated to each tuple especially when a large number of tuples are displayed in users' local display (Problem 2).

Another solution is data auditing. In data auditing, an auditor process periodically contacts the database server to compare the data stored in a database server and its local copy (Ateniese, 2007). Like the one proposed by Wang (Wang, 2010), most of the existing data auditing executes auditing processes at an off-line (not on the path between clients and the DBMS in a cloud server). However, off-line auditors have the following four issues (Problem 3). First, since they are "off-line", capturing creations of new tuples is not easy. Second, leaving an auditor to a third party essentially causes the same security problems we have for clouds. Third, off-line auditors can not efficiently attest the integrity of each tuple in large tables. Finally, hiding the network address of auditors from the cloud is not easy, since auditors have to contact the clouds to attest integrity of information.

3. Description of the Split Cloud

3.1. Structure of the security architecture

We designed new security architecture to cope with the three problems identified in the previous section. This section describes the designs of the proposed security architecture. The SaaS layer and DBMS layer in Figure 1. (b) are split to have two additional layers between them: the data integration and the owner agent layers (Figure 2.). The owner agent is a proxy that represents an owner unit. It is running in a separate host computer owned by each organization unit who owns a system stack in Figure 1. (a). Each owner agent consists of two processes; the security gateway and the shadow auditor. The cloud server and the owner agent are connected by a secure network connection. The frontend (SaaS and data integration layers) and the backend (DBMS, OS, and hardware) can be hosted in the same host computer. Each functional component in Split Cloud is described as follow:

Applications in SaaS layer: It is assumed that the application processes running in SaaS layer are those that interface to the underlying database system. Applications in SaaS layer accept database queries from human users at remote client host computers, forward the queries to the data integration layer, and transmit the results of queries to the human users when they receive the results of the queries from the data integration layer.

Data integration: This layer accepts database queries from the applications running in SaaS layer. Then, it identifies the owner unit of the information each database query needs to process and forwards the queries to the owner agent through a secure connection.

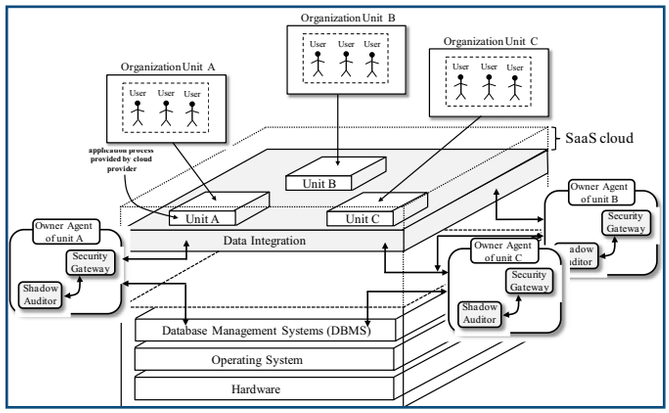
Owner agent: The owner agent is running in a separate host computer owned and maintained by an organization unit. It consists of the security gateway and the shadow auditor. Each organization unit has its own owner agent.

Security gateway: The security gateway performs five tasks. They are (1) authenticating users, (2) duplicating database queries to the shadow auditor, (3) synthesizing SELECT and UPDATE queries for the shadow auditor, (4) forwarding database queries to the DBMS in the cloud server, and (5) performing network address translation (NAT) for queries from the shadow auditor. The security gateway has the definition of each table (the primary key and the name and the data type of all fields in a table) its organization unit owns in the cloud server.

Shadow auditor: The shadow auditor maintains digests of the database tables in the cloud server that are owned by each organization unit. The digests are maintained in the form of tables, each of which consists of the primary key, hash digest of each tuple, and the real-time lineage summaries for each tuple in a database table stored in the cloud server. When the security gateway sends a duplicate of a database query to the shadow auditor, the shadow auditor calculates the hash digest of the latest values of data in a whole tuple and its real-time lineage summaries. The shadow auditor uses the hash digests to periodically and randomly attest the integrity of the contents and the lineage summaries for each tuple stored in the cloud server.

The major advantages in having the security gateway and the auditor "in line" are: the auditor can capture even the query to insert a new tuple to a table, the auditor is hidden from the cloud using NAT at the security gateway, the security at the cloud server can be simplified since only one connection can access the information stored in the cloud for each organization unit, and having the security gateway "in line" and having the auditor behind the security gateway bring the advantage of making the owner agent as the single security checkpoint under the direct control of the owner unit.

Figure 2. The Organization of Split Cloud



3.2. Real-Time Lineage Summaries

We propose real-time lineage summaries to provide the feedback on the quality of data to users in real-time. Real-time lineage summaries are digests of data lineage. Data lineage attaches records as a linked list to each tuple in a database table as metadata to track its history, such as when the data was first entered to a table, who first entered it, when it was modified, and by whom. Each time such an activity is made to a tuple, a lineage record is added to the linked-list (Figure 3. (a). As a result, data lineage provides information about quality of data, which can be used for both proactive and reactive feedback on the quality of information stored in database systems.

Data lineage generates a large volume of lineage records for each tuple in database tables. Although data lineage will provide essential information about the quality of data, its volume can prohibit human data operators from frequently using the lineage information. By saying "human data operators", we mean the people who use data in database tables for their frequent, but critical, decision makings. For example, they can be the dispatchers of military transportation vehicles to move troops and ammunition to a battle zone, which risk the lives of the soldiers if such missions are not carried out on time. When the operators have to make a large number of such critical decisions frequently, a large volume of raw lineage information can significantly slow down their decision-makings.

To solve the problems in the data lineage, real-time lineage summaries are proposed. Real-time lineage summaries are digests of the data lineage for each tuple (Figure 3. (b). They are high-level abstraction of lineage records, focusing on some essential factors about the data stored in a tuple in database tables to provide human data operators a quick view of the quality of data displayed next to the data in real-time. The term, "real-time" means two things. The first is as soon as the data in a database table is updated. Second, each time a human data operator displays the data in his local monitor, lineage summaries are displayed.

We implemented four different real-time lineage summaries. They were for user trust level, data freshness, access location trust level, and the integration of the three factors. The real-time lineage summaries quantify and visualize the reliability level using gradational transition of green (highest reliability) to red (lowest reliability).

User trust levels quantify the trust level of the users who created and updated a tuple in a table. The three classes of the trust levels we implemented were: H (high), M (medium), and L (low). Level H assumed highly trusted IT staffs (e.g., the users in the organization unit who owns the information). Level M is for mid-level staffs (e.g., the users in other units in the same organization) and level L for the staffs from third-party organizations (e.g., external contractors). Data freshness level represents how recently and how often a tuple has been updated. Access location trust level represents where data updates were issued. For large organizations, it is common that database operations are made from foreign branches, each of which has a significantly different security risk level. Figure 4. shows a snapshot of the real-time lineage summary in data freshness. Human data operators can switch to other factors of lineage summaries instantly using a pull-down menu.

Figure 3. The Server-Side Data Structure in Split Cloud

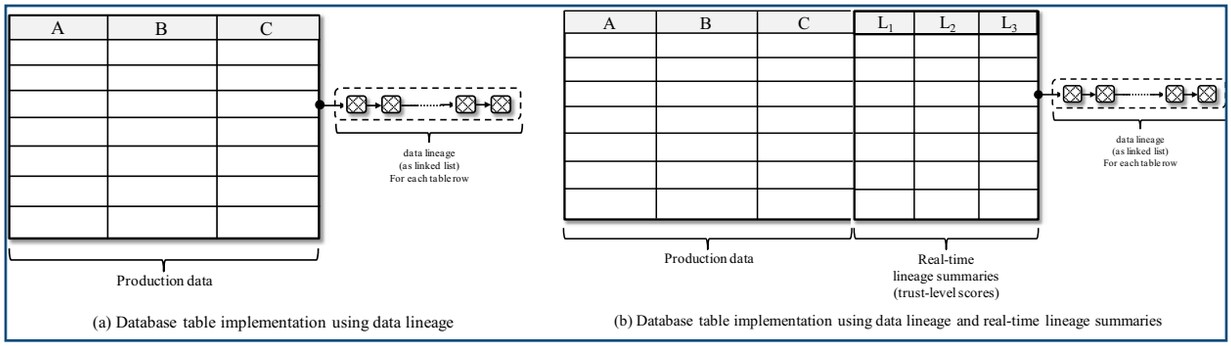
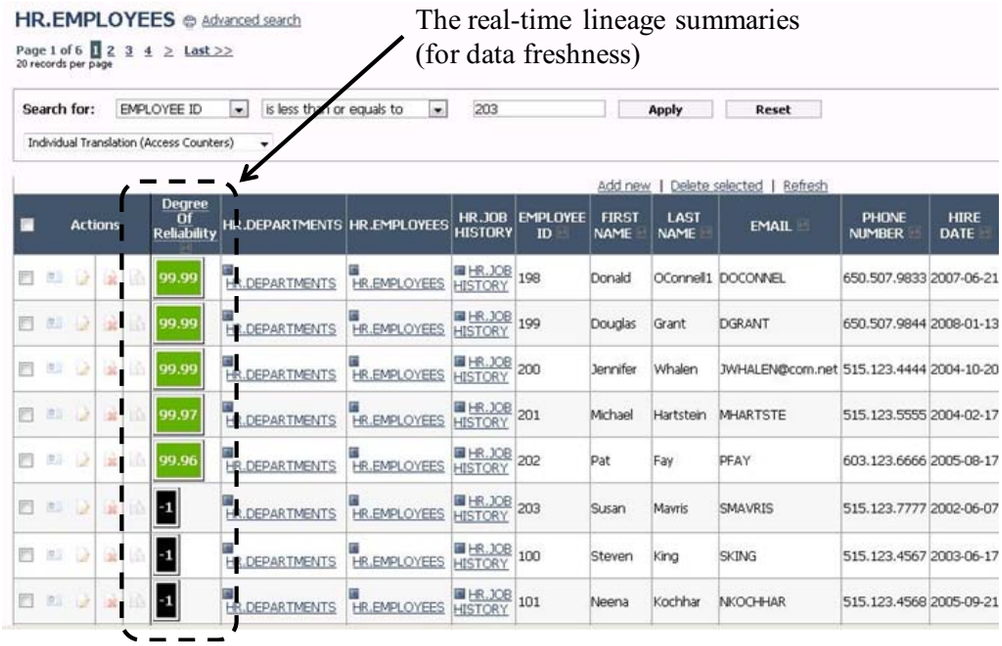


Figure 4. Snapshot of the Real-Time Lineage Summaries (Lineage Summaries on Data Freshness)



3.3. In-Line Shadow Auditor

The shadow auditor calculates the hash digests of the contents and stores the copy of the real-time lineage summaries of each tuple in the database tables owned by the organization unit. The shadow auditor stores the hash digests and lineage summaries for each tuple in the table format, called "digest table", using the same primary key in the table stored in the cloud server. When any update query is intercepted by the security gateway, the shadow auditor updates the hash digest and real-time lineage summaries. For any read query, the shadow auditor tests the integrity background. In addition, the shadow auditor can periodically and randomly issue read queries to tuples in tables stored in the cloud server to constantly attest the integrity of the data and the real-time lineage summaries.

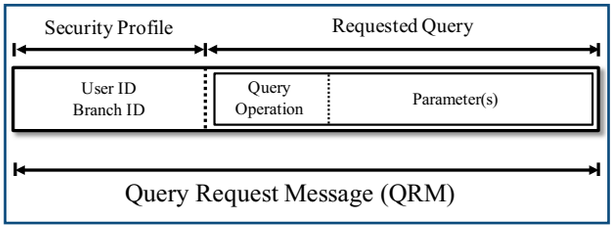
Whenever the shadow auditor issues such read queries to the cloud server, the security gateway replaces the source network address of such queries by others to hide the identity of the shadow auditor using NAT. As soon as the shadow auditor receives the response to such read queries (the contents of the tuple and its real-time lineage summaries), it calculates the hash digest of them and compared them to those locally stored in the shadow auditor.

3.4. The Data Structure and Procedure To Implement Split Cloud

The database tables in the cloud server are designed with the real-time lineage summaries. The real-time lineage summaries for user trust level, data freshness, access location trust level, and the integration of the three factors are stored for each tuple in a table. New lineage summaries are inserted to a table when a new tuple is inserted to a table. They are updated each time a tuple is updated.

Every database query from users to DBMS in the cloud server must go through the security gateway of the organization unit that owns the information. Every query is formatted in Query Request Message (QRM). QRM consists of the security profile and the requested query fields (Figure 5.). The security profile contains the user ID (the identity of the user who issued the query) and the branch ID (the location of the host computer that issued the query). Requested query consists of the query operation and its parameters. The query operation indicates the type of database query, such as SELECT, INSERT, DELTE, and UPDATE. The parameter field consists of the necessary parameter(s) to perform the query operation, such as the table name(s), and all the necessary other parameters needed for the query.

Figure 5. Structure of QRM



When each query goes through the security gateway, the gateway intercepts it and a duplicate of the query is forwarded to the shadow auditor if necessary. Database queries are categorized as either active or inactive queries. Inactive queries are those that do not add any change to the contents of a tuple (e.g., SELECT queries), while active queries are those that insert a new tuple to a table, delete an existing tuple from a table, or update the contents of an existing tuple. The update queries are further classified to replace or delta-update queries. Replace queries overwrite all the fields in a tuple by new values without using the existing values. Delta updates modify the contents of a tuple based on its existing values (e.g., "+$100 to the current balance") or overwrite some, but not all, fields in a tuple. Based on the type of each query, the security gateway performs the following procedures:

Inactive queries: When the security gateway intercepts an inactive query, it simply forwards the requested query in QRM to the DBMS in the cloud server. The DBMS in the cloud responds to the query by transmitting the requested information and its real-time lineage summaries without adding any change in the real-time lineage summary associated to the tuple referenced by this query. The security gateway forwards a duplicate of the response to its shadow auditor. The shadow auditor compares the integrity in the contents by comparing its hash digest and lineage summaries. If any mismatch is detected, a warning message is issued to the user.

Insert queries: When the security gateway intercepts a database query that inserts a new tuple to a table, the security gateway duplicates the QRM to the shadow auditor before it forwards the query to the DBMS in the clouds. When the shadow auditor receives the duplicate of the query, it performs the following activities:

(1) Extracts the primary key, the trust level of the user who issued the query and the location of the host computer that issued the query

(2) Calculates the hash digest of the content of the new tuple using the query parameters in QRM

(3) Calculates the real-time lineage summaries

(4) Stores the hash digest and the real-time lineage summaries of the new tuple in the local digest table using the primary key

Delete queries: If a DELETE query specifies the tuple(s) to be deleted using only its primary key (Figure 6. (a)), the security gateway duplicates the DELETE query to the shadow auditor before it forwards the DELETE query to the DBMS in the cloud server. The shadow auditor deletes the information about the tuple (the hash digest and the real-time lineage summaries) from its local digest table. Otherwise (if a DELETE query specifies the table tuple(s) to be deleted using any key other than the primary key – as shown by Figure 6. (b)), the security gateway first constructs the SELECT query (can be more than one SELECT query) to retrieve the tuple(s) to be deleted using the conditions in its WHERE clause. The security gateway sends the SELECT query to the DBMS in the cloud server to extract the primary key of the tuple(s) to be deleted. The security gateway constructs DELETE query using the primary key(s) for the shadow auditor. The DELETE query is then forwarded to the shadow auditor.

Figure 6. DELETE Query in Which WHERE Clause Contains Only The Primary Key (A) And in Which WHERE Clause Contains A Non-Primary Key (B)

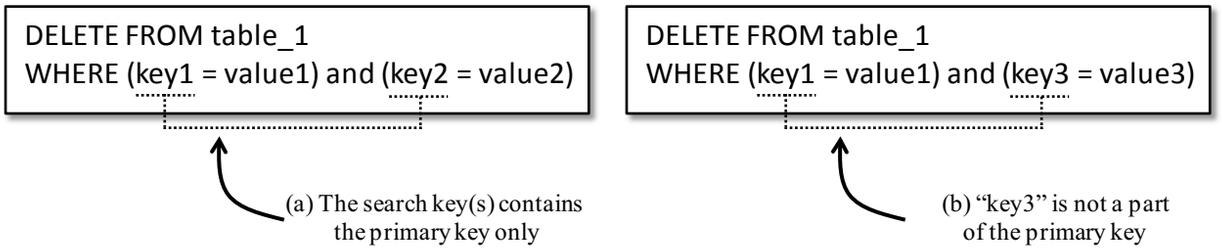
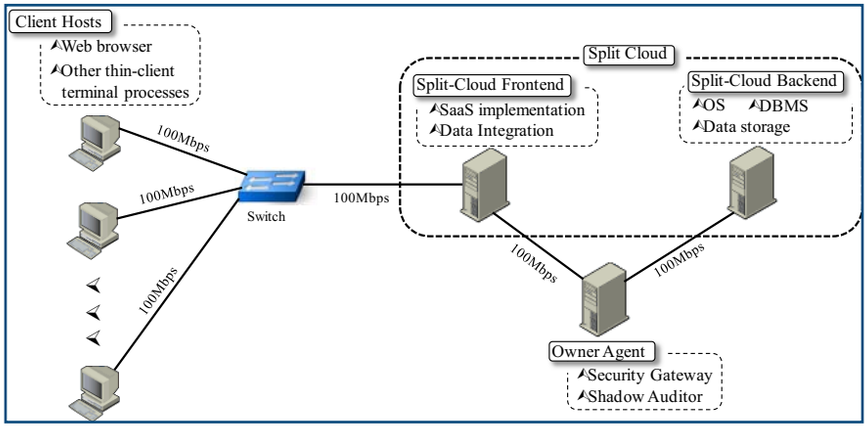


Figure 7. Organization of the Experiment Test Bed



Replace queries: If a replace query satisfies both of the following two conditions, the replace query is processed in the same way as an insert query:

(1) Replace query replaces the whole tuple (i.e., updates the value in every field in the tuple) without using the existing value

(2) WHERE clause contains only the primary key(s)

If one of the above two conditions is not satisfied, the security gateway constructs a SELECT query to extract the contents and/or the primary key of the tuples from the table(s) in the cloud server. The hash digest of the contents and the real-time lineage summaries of the table tuple(s) in the shadow auditor are updated using the response for the SELECT query from the DBMS in the cloud.

Delta-update queries: Delta-update queries are processed using four phases. In the first phase, the security gateway constructs the SELECT query to obtain the tuple(s) whose values should be "delta-updated" and sends the SELECT query to the DBMS in the cloud server. In the second phase, when the tuple(s) is obtained from the DBMS, the security gateway applies the "delta update" to the tuple(s) from the DBMS to calculate the updated value(s) for each tuple. In the third phase, the security gateway constructs a replace query (i.e., UPDATE query) to update the contents of the tuple(s). Finally, the security gateway sends the UPDATE query to both the DBMS in the cloud server and the local shadow auditor. The shadow auditor updates the hash digest and the real-time lineage summaries using the UPDATE query from the security gateway.

For delete, replace, or delta-update queries that involve more than one table, the security gateway processes such queries by constructing SELECT and UPDATE queries for individual table in the cloud server using the definition of each table held at the security gateway. This process translates such queries (those that involve multiple tables) to those that use single table, which the shadow auditor uses to update the hash digests and the real-time lineage summaries.

3.5. Solutions for the three problems in the existing solutions in Section 2

Problem 1: The inline shadow auditor monitors the integrity in the production data, the lineage data, and lineage summaries stored in the cloud using the local hash digests and lineage summaries stored in the shadow auditor. Using this method, the shadow auditor can detect illegal updates even by the cloud administrators.

Problem 2: Real-time lineage summaries provide feedback on the quality of information without downloading the whole liked list of lineage records for each tuple, which significantly slows down the decision makings by the users. Instead, lineage summaries, which are the digest of the lineage records, are used to provide light-weight feedback on the quality of information that helps the users to detect obsolete, inconsistent and even fake data in database tables while the users are using the database tables.

Problem 3: Since the shadow auditor exists logically "inline" on the communication path, it can efficiently attest the integrity in the information, as well as the shadow auditor can easily and reliably capture even the query to insert a new tuple to a table, while it is under the control of the owner unit.

4. Performance Evaluation

4.1. Test Bed for Performance Evaluation

Performance of the proposed security architecture was evaluated by experiments using a prototype test bed. An isolated network test bed for our experiments was set up as shown in Figure 7. Split Cloud was implemented using two servers. One of the servers hosted the DBMS, OS, and the storage dvices, while the other hosted the frontend of Split Cloud that implemented the SaaS and the data integration layers. We implemented the owner agent using a host computer where the security gateway and the shadow auditor were implemented and executed. Client host computers were connected to the frontend of the Split Cloud through a switch. Each component was connected by a 100Mbps Ethernet link. For each of the owner agent, the cloud server, and client hosts, we used a PC running a 3.4GHz Intel D processor with 2.0GB of memory.

4.2. Experiment Designs and Definitions

Response time was measured at each cloud user's host computer under different circumstances. Response time was defined as the time difference between a transmission of a database query and receiving the first response to the query at each client host computer. The three overhead factors we tested were (1) implementing real-time lineage summaries, (2) securing connections between the cloud server and the owner agent using 3DES encryption and SSL connections, and (3) executing the shadow auditor. Using the three factors, we designed seven different experiments as defined below.

Experiment #1 (real-time lineage summary experiments for inactive queries): The effect of real-time lineage summaries to response time was evaluated for inactive queries (SELECT queries). In this experiment, shadow auditor was not used. Either 3DES encryption or SSL secure connection was not used to measure the net overhead of real-time lineage summaries. The response time for SELECT queries was measured while the workload (number of SELECT queries per minute) was increased from 5,000 to 45,000 queries. The response to a SELECT query contained the whole contents of a tuple with the real-time lineage summaries when the real-time lineage summary was used, while only the contents of a tuple was transmitted for "without lineage summary". To avoid the effect of caching, each SELECT query was designed to read only a tuple without repeating a query to the same tuple during each experiment.

Experiment #2 (3DES encryption experiments for inactive queries): To secure the communication between the security gateway in an owner agent and the DBMS in the cloud server, proper encryption would be necessary in Split Cloud. The overhead of encrypting the payload data at the security gateway was evaluated. The response time for each SELECT query was measured while the workload was increased from 5,000 to 45,000. The average response time was calculated for the case the security gateway performed 3DES encryption to the payload data before the data was sent to the DBMS. It was compared to the response time of the SELECT queries without the encryption. The other details were the same as for Experiment #1.

Experiment #3 (SSL experiment for inactive queries): Data encryption protects payload data only from eavesdropping. To provide protection for two-party authentication and data integrity, as well as eavesdropping, more than encryption is necessary. To evaluate the performance of Split Cloud when SSL is used, the response time of user queries was measured when SSL was used between the cloud server and the owner agent. The other details were same as Experiment #2.

Experiment #4 (real-time lineage summary experiments for update queries): This experiment was same as Experiment #1 except that the queries were all replace (UPDATE) queries. The workload was increased from 5,000 to 17,000 queries. Response time was measured for UPDATE queries when real-time lineage summary was used and not used. The UPDATE queries used the primary key only to identify each table tuple.

Experiment #5 (3DES encryption experiments for update queries): This experiment was same as Experiment #2 except that the queries were all replace (UPDATE) queries. All the other details were same as Experiment #4.

Experiment #6 (SSL experiment for update operations): This experiment was same as Experiment #3 except that the queries were all replace (UPDATE) queries. All the other details were same as Experiment #4.

Experiment #7 (shadow auditor experiments): The

Figure 8. Increase in the Response Time When Real-Time Lineage Summaries Were Performed for SELECT (A) And UPDATE (B) Queries

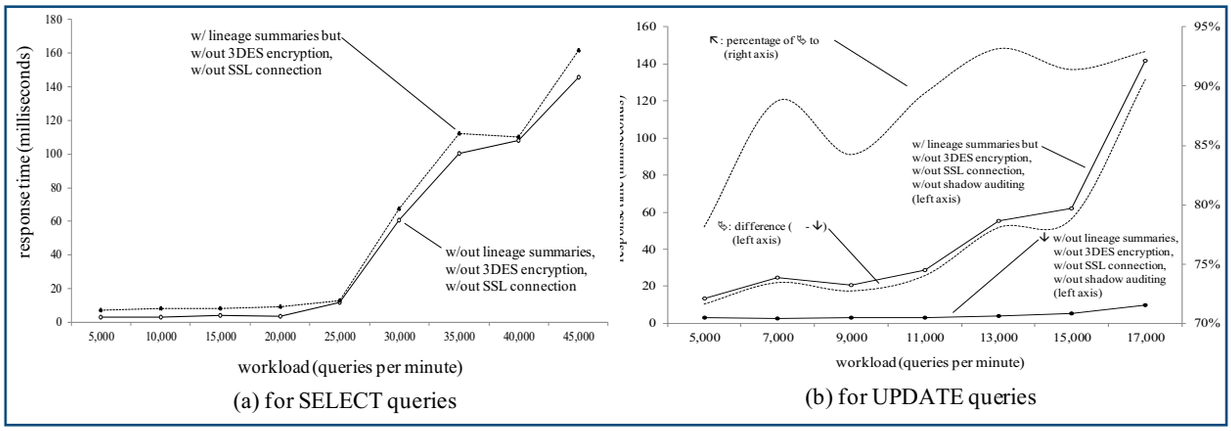
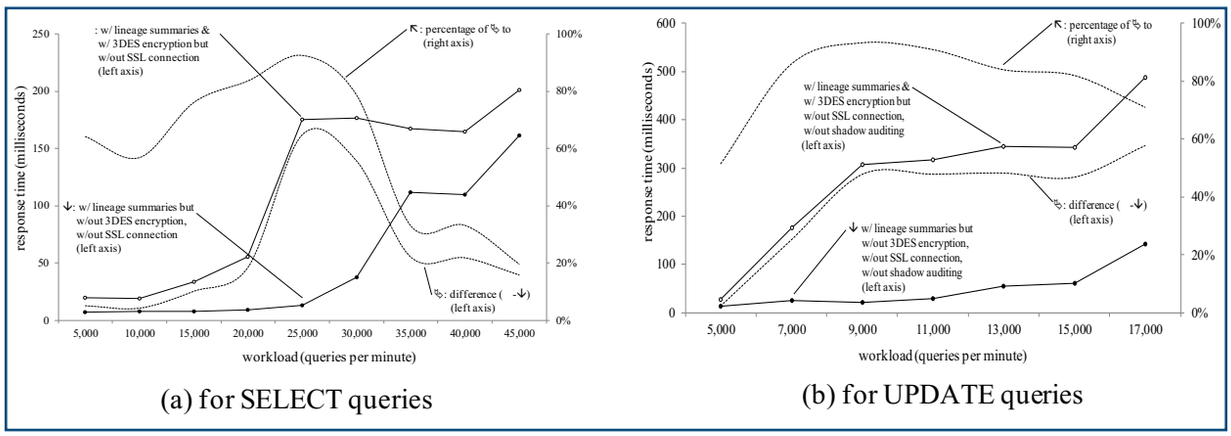


Figure 9. Increase in the Response Time When 3DES Encryption Was Performed At the Security Gateway for SELECT (A) And UPDATE (B) Queries



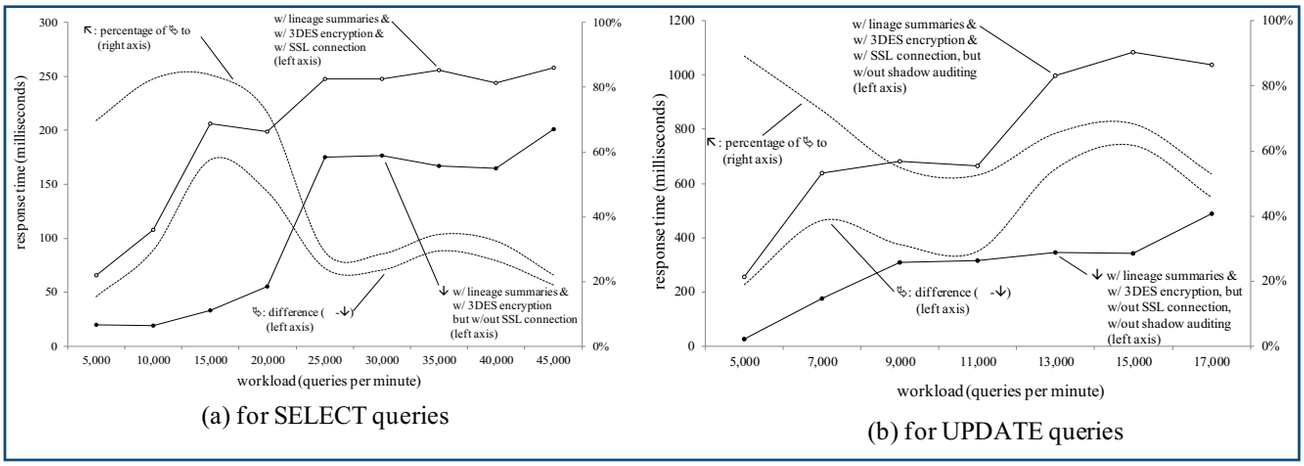
overhead of implementing the shadow auditor was evaluated. The average response time was measured for replace queries when the shadow auditor was used and it was compared with that of the replace queries without the shadow auditor. All the other details were same as Experiment #4.

4.3. Observed Outcomes

Figure 8. shows the results from Experiment #1 and #4 ("lineage summary experiments"). The results in Figure 8. (a) show that there was no major impact from the real-time lineage summaries to SELECT queries. The ratio of the increase in the response time was up to 11.6% for more than 25,000 queries per minute. The overhead for UPDATE queries showed that real-time lineage summaries increased the response time by 10.5 and 25.8ms for low (5,000 queries) and medium (11,000 queries) workload respectively (Figure 8. (b)). For high workload (more than 11,000 queries), the increase was 51.7 and 131.6ms for 13,000 and 17,000 queries (graph in Figure 8. (b)). For more than 11,000 queries, the overhead for implementing real-time lineage summary consisted more than 90% of the response time (graph).

The majority of the overhead for UPDATE queries must have been from adding a lineage record to the linked list and updating the four real-time lineage summaries for each tuple, assuming that the read and write accesses in the disk drive took the same amount of time. Despite the increase in the response time for UPDATE queries, the response time was still lower than 500ms. It is half of the one-second threshold, which is the limit for users' flow of thoughts to stay uninterrupted, while less than 100ms delay is perceived as "instant" by most human users

Figure 10. Increase in the Response Time When SSL Connection was Established for SELECT (A) And UPDATE (B) Queries



(Card, 1991). By having the threshold at the half of the one-second limit, we assume 500ms threshold will safely avoid interrupting users' workflow.

Figure 9. shows the results of Experiment #2 and #5 ("3DES encryption experiments"). The results in Figure 9. (a) show that the increase in the response time for performing 3DES encryption to each query for low to medium workload was 12.8 (5,000 queries) to 46.4ms (20,000 queries) (graph in Figure 9. (a)). The weight of the overhead for performing 3DES encryption to the total response time (graph in Figure 9. (a)) reduced for high workload (more than 35,000 queries). This implies that the 3DES encryption was not the performance bottleneck for scaling workload. The response time for UPDATE queries constantly increased up to 9,000 queries, but for higher workload, the overhead for 3DES encryption was almost constant (290ms) as shown in graph in Figure 9. (b). The response time for UPDATE queries exceeded the 500ms threshold at around 17,000 queries.

Figure 10. shows the results of Experiment #3 and #6 ("SSL connection experiments"). The results of SELECT queries (Figure 10. (a)) showed a similar pattern observed in Experiment #2. For low and medium workload (up to 15,000 queries), the overhead for establishing SSL connections continuously increased (graph in Figure 10. (a)) and the overhead occupied the majority (approximately 80%) of the increase in the response time for SELECT queries (graph in Figure 10. (a)). After the peak, the overhead from establishing SSL connections decreased to 20-30% of the response time for SELECT queries. Establishing SSL connections for UPDATE queries (with 3DES encryption) resulted in significant increase in the response time. The response time exceeded the 500ms threshold for the workload of approximately 6,000 queries, resulting in more than 1,000ms delay after 13,000 queries.

Figure 11. Increase in the Response Time When Shadow Auditing was Performed for UPDATE Queries

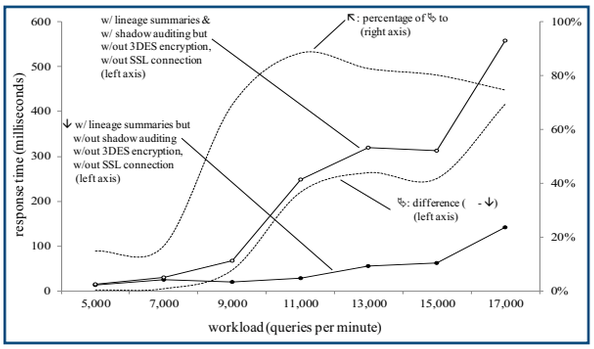
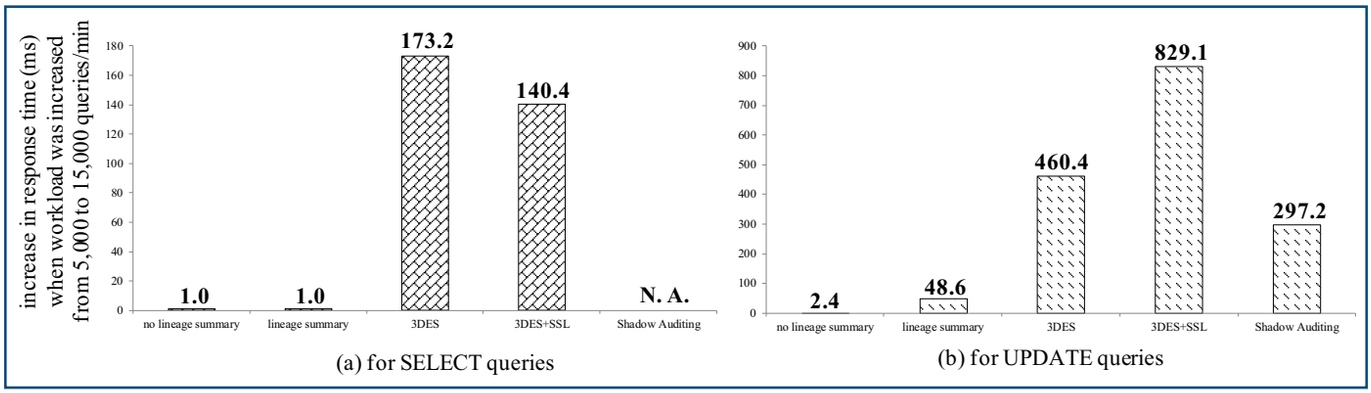


Figure 11. shows the results of Experiment #7 ("shadow auditor experiments"). The results show that shadow auditor occupied the majority of the increase in the response time for medium to high workload (graph in Figure 11.). Up to 7,000 queries, its contribution to the response time was around 20%, while it was more than 70% for the workload higher than 9,000 queries. Similarly, the amount of increase in the response time was less than 50ms for workload less than 9,000 queries. At around 16,000 queries, the increase by shadow auditor exceeded the 500ms threshold.

Figure 12L. Increase in the Response Time (In *Ms*) When the Workload was Increased from 5,000 To 15,000 Queries Per Minute for (A) SELCET and (B) UPDATE Queries



(a) for SELECT queries (b) for UPDATE queries

Figure 12. shows the increase in the response time for each query when the number of queries was increased from 5,000 to 15,000 queries per minute. When real-time lineage summary was not performed for SELECT and UPDATE queries, the increase was only 1.0 and 2.4ms respectively. These results imply that going through the owner agent was not the bottleneck for scaling workload. A similar result was also observed for performing real-time lineage summary for SELECT, although it was 48.6ms longer for UPDATE queries. Assuming that the increase of 48.6ms will not be noticeable to human perception, those results conclude that the real-time lineage summaries were not the bottleneck for the response time.

We observed the major increase in the response time when 3DES encryption and SSL connections were used at the same time. For SELECT queries, the response time increased by 173.2 and 140.4ms, while they were by 460.4 and 829.1ms for UPDATE queries. When shadow auditor was used, its contribution to the response time was 297.2ms, which had more than 200ms margin to the 500ms threshold. These results conclude that the real-time lineage summary will be an effective feedback method for the database applications to provide timely feedback on the quality of data to human users without disturbing their workflow. The Figure s also show that securing connections from the cloud servers to the owner agent caused large increase in the response time, concluding that secure connections between the cloud servers and the owner agent was the major bottleneck for the scalability of Split Cloud.

5. Conclusions and Future Work

In this paper, we proposed new security architecture to enhance direct control to the information stored in cloud servers. The core of the architecture is splitting the cloud stack to two components of the frontend and backend and inserting the owner agent between them. By executing security-critical operations of, user authentication, access control, and updating hash digests and real-time lineage summaries at the inline owner agent, the owner of the data logically preserves the essential security control to its data physically stored in a private cloud.

The two major mechanisms in the security architecture are shadow auditor and real-time lineage summaries. The shadow auditor periodically performs integrity checks to the data as well as to the real-time lineage summaries stored in private clouds. This detects illegal modifications of data even by the insiders of the cloud provider. The real-time lineage summaries are also proposed to effectively provide real-time feedback on the information quality to human data operators who make critical decisions.

We evaluated the performance of the proposed security architecture by experiments using a prototype of the architecture. From the results of our performance evaluations, we drew the following conclusions:

The results from Experiment #1 and #4 showed that the increase in the response time due to the real-time lineage summaries was up to 18ms (which was about 11.3% increase) at a high work load of 45,000 SELECT queries.

The results conclude that the real-time lineage summaries will be effective light-weight feedback regarding the quality in data to human operators, who are making frequent critical decisions using the data. For replace queries, the overhead for implementing the real-time lineage summaries occupied the majority of the increase in the response time. However, for up to 17,000 queries, the observed response time was less than 150ms, proving that real-time lineage summaries will be a feasible solution.

The results from Experiments #2, #3, #5, and #6 showed that the overhead to secure communication channel between the cloud server and the owner agent will be the primary performance bottleneck. The results indicate that the major contributing factors to the increase in response time were the overhead for performing 3DES encryption and SSL connections when the workload was low (below 20,000 queries), while the pattern was opposite (their contribution factor was less than 40%) for high workload, implying that there was another bottleneck, which we could not identify this time. For replace queries, the impact of 3DES encryption and SSL connection was significant. We observed that the response time exceeded the 500ms threshold at 16,000 queries for 3DES and at only 6,000 queries for 3DES and SSL. These results conclude that Split Cloud, as designed this time, will be efficient for database applications that have less frequent updates than reads.

The results from Experiment #7 showed that shadow auditor was not the primary performance bottleneck. As Figure 12. indicates, its contribution to the response time was smaller than 3DES encryption and SSL connections. Figure 12. shows that the net contribution of shadow auditor to response time was 297.2ms, which had more than 200ms margin to the 500ms threshold, when the workload was increased from 5,000 to 15,000 queries.

The future work includes the three areas. We are currently conducting additional performance evaluations to measure the effect of UPDATE queries to the response time when a non-primary key is used for replace and the delta-update queries. Although they are implemented by a combination of SELECT and UPDATE queries, we are evaluating the actual cost of the query types. The other area is a method to reduce the impact from 3DES encryption and SSL connections. Especially for SSL connections, we are testing persistent SSL connections to reduce its overhead, leaving 3DES encryption is the remaining major bottleneck. The third area is the implementation of the data integrity layer.

References

[1] Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z. & Song, S. (2007). *Provable Data Possession at Un-trusted Stores*. The ACM Conference on Computer and Communications Security, Chicago, Illinois, USA.

[2] Bain, A., Mitchell, J. C., Sharma, R., Stefan, D. & Zimmerman, J. (2011). *A Domain-Specific Language for Computing on Encrypted Data*, Foundations of Software Technology and Theoretical Computer Science, IIT Bombay, Mumbai, India.

[3] Card, S. K., Robertson, G. G. & Mackinlay, J. D. (1991). *The Information Visualizer: An Information Workspace*. The SIGCHI Conference on Human Factors in Computing Systems, New Orleans, LA, USA.

[4] Cloud Security Alliance (CSA). (2011). *Top Threats to Cloud Computing V 1.0*.

[5] Groth, P., Luck, M. & Moreau, L. (2004). *A Protocol for Recording Provenance in Service-Oriented Grids*. The International Conference on Principles of Distributed Systems, Grenoble, France.

[6] Mell, P. & Grance, T. (2011). The NIST Definition of Cloud Computing (NIST 800-145). U.S. Department of Commerce, USA.

[7] Moitra, A., Barnett, B., Crapo, A. & Dill, S. (2009). *Data Provenance Architecture to Support Information Assurance in a Multi-Level Secure Environment*. IEEE Military Communications Conference. Boston, Massachusetts, USA.

[8] Paul, R. A. (2005). *DoD Towards Software Services*. The IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, Sedona, Arizona, USA.

[9] Ristenpart, T., Tromer, E., Shacham, H. & Savage, S. (2009). *Hey, You, Get off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds*. The ACM Conference on Computer and Communications Security, Chicago, Illinois, USA.

[10] Sahai, A. (2008). *Computing on Encrypted Data*. The International Conference on Information Systems Security, Hyderabad, India.

[11] Srinivasamurthy, S. & Liu, D. (2010). *Survey on Cloud Computing Security*. The IEEE International Conference on Cloud Computing Technology and Science, Indianapolis, Indiana, USA.

[12] Tsai, W. T., Wei, X., Zhang, D., Paul, R., Chen, Y. & Chung, J. Y. (2007). *A New SOA Data-Provenance Framework*. The International Symposium on Autonomous Decentralized Systems, Sedona, Arizona, USA.

[13] Wang, C., Ren, K., Lou, W. & Li, J. (2010). Toward publicly auditable secure cloud data storage services. *IEEE Network*, 24(4), 19-24.

[14] Zhang, X., Wuwong, N., Li, H. & Zhang, X. (2010). *Information Security Risk Management Framework for the Cloud Computing Environments*. The IEEE International Conference on Computer and Information Technology, University of Bradford, Bradford, United Kingdom.