Access Control for Sensitive Data in Hadoop Distributed File Systems

Yenumula B. Reddy
Department of Computer Science
Grambling State University, USA
email: ybreddy@gram.edu

Abstract—**User access limitations are very valuable in Hadoop distributed file systems to access sensitive and personal data. Even though the user has access to the database, the access limit check is very relevant at the time of MapReduce to control the user and to receive only permissible data. Data nodes do not enforce any access control on its access to the data blocks (read or write). Therefore, Kerberos ticket granting model for user login and user access permissions to MapReduce jobs do not limit the unauthorized data to obtain from the access granted database. In addition, to secure the data during processing, the authentication and authorization of data is required. The problems broadly include a) who will access the data, b) how it will be encrypted, and c) stability of data processing while the data are continuously growing. The current study includes the security mechanisms currently available in Hadoop systems, requirements of access control mechanisms, and change of access control depending upon the sensitivity of data.**

Keywords—Hadoop systems; access control; distributed file systems; MapReduce; network environment.

I. INTRODUCTION

The term big data is misleading in size and organization. Big data means exceeding the normal capacity and unmanageable with current available technologies. The data can be in the form of structured or unstructured category. It is large in the form of volume, variety, and velocity (increasing at any given time). In Defense organizations, big data can contain defense related documents that include text, images, and videos. The high sensitive data need to be analyzed and processed for real-time response. The initial designers of data models (hierarchical, relational, and the network) did not have an idea of volume of current data, type of data, and speed of growing. The Federal Government currently has a big challenge of managing big data volume compared to any other organization. The US health care system is expected to grow in size to yottabytes (10^{24} gigabytes) soon. Similarly, the Defense data will grow in size to yottabytes.

From 1960 to 2000, the storage, communication and processing time was very expensive. So, the concentration was storing the data and organizing the data using different data management techniques (indexed, index sequential, random, etc.). The recent developments made the storage, communication, and processing time very cheap. Therefore, corporations began storing all kinds of data (images, text, symbols, different languages, and various volume sizes) to meet the various types of emergencies. Thus, analysis and control of data has become unmanageable. The trend is built to elaborate models with more test data rather than simple models. The task in Defense organizations is to manage a large chunk of data sets, provide additional benefits like statistical data, and real-time answers to responsible officers. Therefore, threats to the database need to be identified sooner, so that the data can be protected. The query processing and threat detection must happen in real time and with lower cost.

After year 2000, the internet usage increased exponentially. The server and storage technology underwent many changes. Therefore, the distributed computing with many nodes was extremely difficult for many reasons such as distributed locking, data parallelism and distribution, load balancing, failure recovery, and network congestion. The big data revolution started after the release of "MapReduce: Simplified Data Processing on Large Clusters" in 2004 by Google. Soon, Google became the number one search engine. Next, Google released another paper "The Google File System", which is a high-performance, widely available distributed file system that provides the storage layer for the MapReduce processing. The key advantage is 'move processing to data' rather than 'moving the data to computer resources'.

The Hadoop project was developed in Java around these two papers and became a new paradigm [1, 3]. The system was adopted more in the DoD (Department of Defense) than in other agencies. The problem in Hadoop system design is lack of reusability of existing applications. The Apache Hadoop defines the new Java Application Programming Interfaces (API's) for data access. This means the existing applications must be rewritten to access their big data in the Hadoop environment. Further, more work is needed to protect the data at source and at communication lines. Research is in progress in understanding the Hadoop system design, usage, and security status. Security can be improved at the source level using Honeypots database, and at communication level through protocol modifications.

Federated systems enable collaboration of multiple systems, networks, and organizations with different trust levels. Authentication and authorization procedures of a client must be kept separate from services. The deployment of federated security architecture was discussed in Windows Communication Foundation (WCF)

 ISBN: 978-1-61208-310-0

[1]. WCF provides support for building and deploying distributed systems that employ federated security. Domain/realm, federation, and security token service consist of primary security architecture of federated systems.

Traditional security is about protecting the resources within the boundary of the organization. The federated systems require the security specification for each function. Further, the federated systems change over time with new participants joining and they may not all be trusted. Therefore, individual protection domains are required for each entity. Boundary constraints are required depending upon the system. Therefore, monolithic security domain will not work in federated systems.

Existing federated security systems separates the client access, associated authentication and authorization procedures. As the federated systems are distributed, they need collaboration between networking, organizations and associated systems. Maintaining security among these is not an easy task, since multiple entities are involved. Security threat may be unavoidable from a variety of sources. Therefore, a high level coarse-grained security goals need to be specified in the requirements.

The rest of the paper discusses the review of research, security policies and proposed design. The recent developments in federated systems security are discussed in Section II. The proposed model is analyzed in Section III, and controlling the user access in Hadoop systems to the proposed model in Section IV. Section V provides the conclusions and suggestions for future research.

II. RECENT DEVELOPMENTS

Data reside in mobile devices, social media, cloud, and many static and dynamic storage media. IBM announced real-time security for big data environments that include data from mobile devices, social media, and cloud computing [2]. Tight coupling of the name space and block storage is possible by limiting the use of block storage directly. Further, IBM offers data masking to de-identify sensitive information as it moves into and out of big data systems. The implementation is limited to 60K tasks, but the next generation of Apache MapREduce supports 100K concurrent tasks [3]. In MapReduce, users specify the computation in terms of the map and a reduce function. The underlying scheme in MapReduce package automatically paralyzes the computation and schedules the parallel operations so that the task uses the network and computational facilities to perform the operations faster. An average of a hundred thousand MapReduce jobs are executed on Google clusters every day.

Halevy et al. [4] suggested that representing the data of large data sources with a nonparametric model is needed compared to summarizing parametric model because the large data sources hold a lot of details. The authors felt that choosing unsupervised learning on unlabeled data is more powerful than on labeled data. They pointed out that creating required data sets by automatically combining data from multiple tables is an active research area. Combining data from multiple tables with data from other sources, such as unstructured Web pages or Web search queries, is an important research problem.

User logs of search engines on query clustering, query expansion, and user generated query reformulations have attracted many researchers in recent years. Riezler et al. [5] studied the user queries and snippets of clicked results to train a machine translation model to bridge the “lexical gap” between query and document space. The authors claimed that the model has better results on contextual query expansion than other systems (based on term correlations). Further, dissecting the contribution of translation model and language model improved the search results. The improved search models include the combination of the correlation-based system, language-based system, and language model filter.

Queries without understanding the full complexity of schemas, source relationships, and query languages are another task of the research area. Talukdar et al. [6] presented a system in which non-expert user can create a new query templates and Web forms to be used by any user with related information needed. The proposed system works with the combination of user queries with keywords and sequence of associations used by the system. The system processes with multiple ranked queries, linking the matches to keywords. The query sets attached to Web form creates a particular query so that the user can input or fill in the blanks for search. The authors claimed that their approach is highly promising compared to other approaches for practical bioinformatics queries.

Thuraisingham [7] used federated database systems as cooperative, autonomous, heterogeneous, and distributed. The author discussed various types of security policies such as local, component, generic, export and federated. He further outlined the ways to generate and enforce the security policies. Jonsher and Dittrich [8] discussed the access control mechanisms to be applied at a global layer and showed how they can be mapped onto less powerful mechanisms of component database management systems. Tari and Fernadez [9] discussed the federated access control and secure access control in distributed object kernel for federated database systems. They proposed a unified security model aiming for the integration of existing access control models, such as mandatory access control and discretionary access control that could be imposed on local components. The authors introduced task agents within database agents to enforce the federated security policies using security procedures. The proposed idea may work better than existing procedures to secure the Hadoop systems with limitations. Geon and Qian [10] discussed the interoperation and security in a large database and recommended a security design. One of the proposals made by the authors is defining the interfaces to preserve the security. They suggested restricted access on direct and indirect access in these distributed systems. The suggestions may extend to honeypot databases. Ebmayr et al. [11] discussed the taxonomy of major design choices and access controls for federated database environments. Their work presents the main security mechanisms,

 ISBN: 978-1-61208-310-0

authorization, and access control through simple examples.

Neuman [12] discussed the security in federated systems and reasons for failure. As the future technology is based on federated systems, it is necessary to understand the interactions and proper definition of security boundaries. Further, engineering, development and deployment must enforce these boundaries. Security and fail-recovery are very important in small or large data sets. The data recovery from failure was discussed by Gamble [13]. Big data policies and law enforcement in Hadoop systems was discussed by Oleskeer [14]. The research discussed the monitoring and user accountability, predictive policies, and the use of Hadoop systems by enforcement agencies.

Hadoop security was discussed in the reports [15-18]. Ravi's [15] comment on *Security for Big Data in Hadoop* is a two key solution that includes authentication and encryption. According to him, Kerberos file system has better protection to keep the intruders away from accessing the file system. Since the discipline is new, there has not been much research in Hadoop security. Chary et al. [16] presented security implementation in Hadoop systems. Their research discussed the security in distributed systems and current level of security in Hadoop systems. It also includes the client access for Hadoop cluster using Kerberos Protocol and authorization to access. O'Malley et al. [17-18] discussed the security in Hadoop design. They discussed the security threats from different user groups, Kerberos authentication system, role of delegation token in Kerberos and MapReduce implementation details. These papers discuss the encryption, limiting access to specific users by application, isolation between customers, and information protection. Their research further emphasizes the need for internal and external security for Hadoop systems.

Roy et al. [19] presented a prototype system called "Airavat", a MapReduce-based system which provides strong security and privacy guarantees for distributed computations of sensitive data. The authors provided a method to estimate the different parameters that should be used and tested on several different problems. Their proposed system has weak use cases, a complicated process in specifying the parameters, and is not efficient in general composition of MapReduce computations (the MapR function followed by another mapper). Some organizations raised critical questions on privacy and trust of the system.

III. PROPOSED MODEL

In this paper, a dependable security model is presented and explained how it differs from existing federated systems. Further, it discusses the difference between securities in federated systems and general distributed systems.

Hadoop system users are of four types. First, use of the facilities for storing, accessing, processing, and transferring the information; second, acquiring information (Google, for example) for research (to acquire knowledge for scientific purposes); third, accessing the globally stored information (including blogs) for current and future applications (research & business); and the fourth is hackers, who misuse the information or destroy sensitive information. The main purpose of security is to avoid unauthorized access and save the sensitive information from the intruders.

In federated systems, security is available through authorization and authentication. The authorization is provided using the Key Distributed System (KDS). Every federated user requires a security key to access the data. The client access in Hadoop high-level architecture has job tracker as well as task tracker. Once the client is authorized to access the data, the data will be processed on HDFS data nodes and package MapReduce will fuse the information as required by the client.

If a user from one business needs service in another business, a security token will be created to access the resources. The security token is the authorization to access and build the trust of the client. In Kerberos, the new delegation token is obtained with TokenID (as in (1)).

$$\text{TokenID} = \{\text{ownerID, renewerID, issueDate, maxDate, sequenceNumber}\} \quad (1)$$

Once the delegation token is obtained, it is valid for a day as a default or the token can be renewed for a week. Normally, the token is issued until the job is completed. The additional requirements for a job token are access limits and verification in MapR process. Access limits are the controls on stored information. The MapR module must verify the access rights of a user before filtering (reduce) and deliver the filtered response to the user query. The equation (1) is rewritten as (2) to incorporate the access rights of a user.

$$\text{TokenID} = \{\text{ownerIDc, renewerID, issueDate, maxDate, sequenceNumber, outputCheck}\} \quad (2)$$

where ownerIDc is the identification of user access rights and OutputCheck is verification of information after MapR function is performed. Formula (2) also helps in accessing blocks. Therefore, the OwnerIDc is formulated as in (3).

$$\text{ownerIDc} = \{\text{ownerID, accessLimit, keyID, expirationDate}\} \quad (3)$$

where ownerID is the right to use, accessLimit is the limitation of access to blocks or files (read/write/execute/update), and expirationDate is a valid time limit.

The access limits depend on types of files accessed by the client. For example, a nurse in the hospital can access patient information, unless it is restricted. The accessLimit keyword stops access to unauthorized information. Similarly, depending on the information,

 ISBN: 978-1-61208-310-0

sensitive or clustered, the accessLimit keyword helps in securing the files.

In the federated security, the access limits are required within and outside the organization. Federated user jobs include job/task localization and shuffling the information. For example, in Yahoo, once the user is authenticated, the servlets check the authentication of the user to permit the operations. For example, in a hospital, a nurse is allowed to access the patient records. This should be limited to the type of user and permitted information. Similarly, for a doctor the limits apply to the medical records of the patient. These limits are designed in equation (2).

In federated databases, key management, access control, policy management, auditing, and distributed authentication are very important. The access control design changes depending on the environment and sensitivity of the information. Kerberos access control is available for the federated databases. Additional controls are required as introduced in (2) for sensitive data.

IV. CONTROLLING THE USER ACCESS TO DATABASES

Each ownerIDc is defined with a set of permissions to the database within or outside the organization for processing and retrieving information. Every user is tagged an access limit 'token' during login. If a login user is an intruder, the system must detect the access rights of the user (intruder) before permitting access to the database. The intruder is generally caught at login time or MapR time to retrieve unauthorized information. If an intruder is internal user and tries to access the unauthorized information, the intruder will be controlled with access limits to resources and further verified at MapReduce time. Consider the objective function G with a set of users (N), set of access rights (A), set of allowed resources in the database (D), and the response (U).

$$G = \{N, A, D, U\} \quad (4)$$

where

$n_i \in N$ (Set of users);

$a_i \in A$ (Set of access rights);

$d_i \in D$ (Set of allowed resources in the database);

$u_i \in U$ (The return result of the user query)

In the current research, the healthcare model is used for access control. For example, if a person is admitted to the emergency room, doctors can access the most up-to-date medical history and related information. This information helps the medical team to develop a personalized treatment plan while avoiding duplicate tests and procedures. It is assumed that a patient may be admitted to any hospital on the globe and patient's medical records are available on-line to the doctors to provide personalized treatment. Further, the access rights to patient's information are clearly defined to each person attending the patient.

The following assumptions, definitions, propositions, and theorems help us to generate the algorithms that control the unauthorized information.

- Every authenticated user n_i $(n_i \in N)$ will be provided a service token to a resource within its domain with a set of access types a_i. The limitation helps to control the user for resource access.
- For each service requested by the user, the system generates a set of access permissions to the resources. The services requested should not exceed the user limits. If the requested resources are outside the user boundaries, then the system alarms the security and denies the request.
- Hacker is a user that does not have any role in the system.
- An authorized user will be treated as hacker if the user tries to access unauthorized information. For example, the health care staff member will be treated as an intruder if the user accesses unauthorized data or misuses (printing and forwarding, for example) the authorized information.

Proposition 1: If user n_i possesses access types $\bigcup_{i=1,k} a_i$ where k is the access limit, then the user n_i gets permission to access the database only if $\bigcup_{i=1,k} a_i \in U$ (the results must match with access permission). That is accessed information and MapR result must satisfy the access rights.

Proposition 2: Given any two users n_i and n_j the operation $n_i \bigcap n_j \neq \phi$; where ϕ is null and $\neq$ represents 'different from' (not equals). That is, two users may have common access types. There are many examples using doctor and nurse combination.

- Two nurses working on a patient in different shifts have access to see the primary data about a patient (access to the data to collect or examine the type of medicine given at different times).
- A doctor and a nurse may have some common access rights on a patient.
- Two doctors in different shifts attending the patient have the same access rights.
- Two doctors working on a patient can have access to the previous data or data collected by the nurses.

Proposition 3: For any two users n_i and n_j the operation $n_i \bigcap n_j = \phi$; then the two users do not have any common access type. That is, they must be performing different operations on the resource (a doctor and a

cleaning person; a doctor and an accountant) or two different doctors do not have any common access rights.

Proposition 4: If user n_i possesses resources d_i and d_j with access types $\bigcup_{i=1,k} a_i$ and $\bigcup_{j=1,l} a_j$ respectively, then $\bigcup_{i=1,k} a_i \bigcap \bigcup_{j=1,l} a_j = \phi$, shows the user n_i do not have the same access types on both data sets d_i and d_j.

In other words, if user n_i has access types $\bigcup_{i=1,k} a_i \bigcap \bigcup_{j=1,l} a_j \neq \phi$ then the user n_i can have one or more common access rights between the two data sets. These cases are bound for close observation by security at the time of data consolidation by MapR; otherwise security threat will be alarmed.

Definition 1: The user with complete authorization access is called a super user (S). The super user 'S' possesses access rights of all users $S \supseteq \bigcup_{i=1,n} a_i$ where $\supseteq$ means contains. All accesses of super user on the database must be recorded.

Definition 2: The user that does not have any authorization is called hacker (h_i) and represented as H ($h_i \in H$) and $\forall H$ (hackers) the access rights $a_{ih} \mapsto d_i \equiv \phi$ is true; a_{ih} is access rights of the hackers, $\mapsto$ implication to, and $\equiv$ is equivalent to.

Proposition 5: If $Q(h_i, d_i)$ is a query placed by the hacker h_i on data source d_i simulating the user query $Q(n_i, d_i)$ then a mismatched query will be locked and alarmed the security.

Since the hacker could get the authorization and does not know the set of access rights a_i and access to permutable resources d_i, then the security takes control of the hacker and alarms the security manager. If an unauthorized user repeatedly accessing through a particular terminal, then the security system locks the terminal and user access to resources till the problem is resolved. Similar action will be taken for batch submission.

Furthermore, the query and assigned information is recorded as part of utility, and if the hacker poses the same query then the query and utility will be recorded for future intruder detection. For example, if h_i pretends as user n_i and try to access the information (that user n_i has permissions) then the system stamps on h_i as $h_i u_i$ (called hacker utility) and process as the hacker action.

Proposition 6: If an internal hacker tries to access unauthorized information then the system will alarm the warning to user and then send the internal security threat to security administrator.

Let internal hacker place a query $Q(In_i, d_i)$ on data source d_i simulating the user query $Q(n_i, d_i)$. A mismatched query will get information from security token service. Algorithm-1 helps to handle the internal hacker while trying to retrieve the information outside the user bounds.

Algorithm 1:

If $Q(n_i, d_i)$ matches the ownerIDc of TokenID, then the corresponding utility function u_i will be generated, else the query reflects as $Q(n_i, hd_i)$, where h is a hacker.

If the hacker is an internal user then

$hu_i \supseteq u_i + h' d_i$ (u_i Internal user), alarms security manager about internal hacker.

If $Q(n_i, d_i) \subset u_i$ then exit;

else if $Q(n_i, d_i) \not\subset u_i \;\&\&\; Q(n_i, d_i) \cong hu_i$ then

convert $Q(n_i, d_i)$ as $Q(n_i, hd_i)$ and generate $hu_i \supseteq u_i + h' d_i$

Store the user utility hu_i that contains $u_i + h' d_i$ and inform security and keep the counter (log) in alert for further attempts.

The Algorithm 1 helps to detect the hacker if the user tries to gain the information with unauthorized access from the database. The following query and Table I explains the unauthorized access to information.

If $Q(n_i, d_i) \equiv Q(hn_i, d_i) \not\subset u_i$ or $Q(hn_i, d_i) \approx hu_i$ then

$Q(hn_i, d_i) = hu_i$, retrieve hu_i (utility from the Hacker alarm to database) and alarm security alert;

where hu_i is available in log or identified as a new hacker and logged as new entry. The log is provided in Table I.

TABLE I. HACKER LOG AND ACTION

Hacker	Status	Result	Action
A	new	hu_i	New hacker, alarm
A	repeat	hu_i	Alarm and freeze

In general, if the hacker attempts to gain access to the database at different trimmings, the time attribute plays an important role to detect the hacker. Algorithm 1 is modified as Algorithm-2.

 ISBN: 978-1-61208-310-0

Algorithm-2

If $Q(n_i, t_i, d_i)$ is genuine and attempted during duty times then corresponding utility function u_i will be generated,

else the query reflects as $Q(n_i, t_j, hd_i)$ then user will get $hu_i \supseteq u_i + h'd_i$ (where u_i is internal user information and $h'd_i$ is the hacker alarm at time t_j).

If $Q(n_i, t_i, d_i) \subset u_i$ then exit (user access accepted)

else

if $(Q(n_i, t_j, d_i) \not\subset u_i) \,\&\&\, (Q(n_i, t_j, d_i) \cong hu_i)$

Convert $Q(n_i, t_j, d_i)$ as $Q(n_i, t_j, hd_i)$ and generate $hu_i \supseteq u_i + h'd_i$ (alarm alert to Security manager)

Note: Store the user utility hu_i that contains $u_i + h'd_i$ and alert security and keep the counter for further attempts.

If the hacker is external, then divert to the KDS. If the user hacks with authentication, then the time stamp will help to detect the hacker. For example,

If $Q(n_i, t_j, d_i) \equiv Q(hn_i, t_j, d_i) \not\subset u_i \; or \subseteq hu_i$ then

$Q(hn_i, t_j, d_i) = hu_i$, retrieve hu_i, and alarm the security;

where hu_i is available in log or identified as a new hacker and logged as a new entry. Table II provides the log entries.

TABLE II. HACKER LOG AND DETECTION

Hacker	Status	Time	Result	Action
A	New, internal	Outside-bounds	hu_i	Detect as internal hacker and alarm
A	Repeated, internal	Within-bounds	hu_i	Check for presence of real user and alarm and find real user

Depending upon the security level, Algorithm-2 will be modified by adding the terminal type and log-on timings. Terminal type and time of access attributes along with access type attributes will protect the secret and top secret information.

Let us assume the hospital environment in the healthcare system. A doctor and nurse have common access to certain information (the doctor prescribes the medicine and the nurse is responsible for giving it to the patient). Then, the attributes patient id, type of medicine, and scheduled time medicine to be given to patient is accessible by the nurse. The same attributes are also accessible by the doctor. Therefore, the system security depends upon the merge and decomposition of two or more users.

Theorem 1: The security of the hospital system depends upon time and terminal type attributes but not on the merge and decomposition operators.

Let $u^t_{i,d}$ be the doctor user and $u^t_{i,n}$ be the nurse user at any time *t*. The decomposition of these attributes at time *t* is

$$u^t_{i,d} \cap u^t_{i,n} \neq \phi \;\; or \;\; u^t_{i,d} \cap u^{t1}_{i,n} \neq \phi \qquad (5)$$

The relation is true, since the duty timings are different. Similarly,

$$u^{t,\tau}_{i,d} \cap u^{t,\upsilon}_{i,n} \neq \phi \;\; or \;\; u^{t,\tau}_{i,d} \cap u^{t,\tau}_{i,n} = \phi \qquad (6)$$

where $\tau \; and \; \upsilon$ are terminal types and $u^{t,\tau}_{i,d} \cap u^{t1,\tau}_{i,n}$ shows the doctor and nurse on the same terminal at different times. For security purposes, the nurse is not allowed on the doctor's terminal, since the terminal is involved in access rights. We can show the similar operation for merge. This concludes that the merge and decompose operations provide the common and combined access types without compromising security.

Theorem 2: Any change in resource access will affect the utility (result of merge and decomposition operators).

The resource access changes will be done by the security authority through the Systems Administrator. The change in access rights in any user will affect the utility function and as a result the output of merge and decompose (in MapR process) operations change. The change reflects the presence of intruder. For example, if the nurse becomes head nurse (hn) then:

$$u^t_{i,d} \cap u^t_{i,n} \neq u^t_{i,d} \cap u^t_{i,hn} \; or \; u^t_{i,d} \cap u^{t1}_{i,n} \neq u^t_{i,d} \cap u^{t1}_{i,hn} \qquad (7)$$

$$u^t_{i,hn} \cap u^t_{i,n} \neq \phi \; or \; u^{t1}_{i,n} \cap u^{t1}_{i,hn} \neq \phi \qquad (8)$$

$$u^{t,\tau}_{i,hn} \cap u^{t,\upsilon}_{i,n} \neq \phi \;\; or \;\; u^{t,\tau}_{i,hn} \cap u^{t,\tau}_{i,n} \neq \phi \qquad (9)$$

It shows that the head nurse has the access rights of a nurse and additional access permissions to resources. The change must reflect, otherwise security alarm alerts. The proposed system was implemented using CGI (Common Gateway Interface) framework in Python language (V2.6) for hospital environment. The program also uses HTML, JavaScript, AJAX and PHP for support. The database used is MySQL database and Hash security to encrypt the

 ISBN: 978-1-61208-310-0

password. The results were satisfactory. We are extending the work to Hadoop database (health care environment).

V. CONCLUSIONS AND FUTURE EFFORTS

The research was concentrated on security in Hadoop distributed file systems during access and MapReduce operations. The authentication access limits proposed in this paper ensure the limitations of user access to global data and avoids the unauthorized access to data. An objective function was created for the user access to the database and theoretical foundations were provided. The algorithms help to detect an intruder. The security model includes the access rights and resulting return value. The return value depends on the access permissions (Theorem 2).

The healthcare database is treated as a 'Big Database' (for global availability) because a patient may be admitted in the hospital of his/her choice (irrespective of place and time). The doctor on-duty needs to see the history of the patient to avoid unnecessary tests and provide personalized treatment. At the same time, the data are confidential. For example, news reporters are eager to collect information on a celebrity or an administrator (President, Senator, or movie actor, or any similar important person). Easy target is a hospital employee (nurse, for example). Some unauthorized doctors may be eager to know the sensitive data. In such cases, access rights restrict the unauthorized access to such data.

Future efforts of projects related to 'Federated Hadoop systems include the design, implementation and processing of data with security tags for sensitive data. Develop the algorithms with appropriate access rights to provide security to sensitive data within and outside organization is an important research problem.

ACKNOWLEDGEMENTS

The research work was supported by the Minority Leaders Program through contract number GRAM 11-S567-0017-02-C2. The author wishes to express appreciation to Dr. Connie Walton, Provost and Vice President, Academic Affairs, GSU for her continuous support. The author also wishes to appreciation to Prof. Rama, T, Dean of professional studies, GSU, for proof reading and corrections.

REFERENCES

[1] Athontication, Microsoft Patterns & Practices, http://msdn.microsoft.com/en-us/library/ff649763.aspx 2012 [accessed: April 2013].

[2] "IBM Addresses Security Challenges of Big Data, Mobile and Cloud Computing", www.03.ibm.com/press/uk/en/pressrelease/39166.wss, ARMONK, N.Y., 18 Oct 2012 [accessed: April 2013].

[3] J. Dean and S. Ghemawat., "MapReduce: simplified data processing on large clusters", CACM 50th anniversary issue, Vol. 51, issue 1, Jan 2008, pp. 107-113.

[4] A. Halevy, P. Norvig and F. Pereira., "The Unreasonable Effectiveness of Data", IEEE Intelligent Syst., 2009, pp. 8-12.

[5] S. Riezler, Y. Liu and A. Vasserman,"Translating Queries into Snippets for Improved Query Expansion," Int. Conf. Computational Linguistics (Coling 08), 2008, pp. 737–744.

[6] P. P. Talukdar et al., "Learning to Create Data-Integrating Queries," Int. Conf. Very Large Databases (VLDB 08), Very Large Database Endowment, 2008, pp. 785–796.

[7] B. Thuraisingham, Security issues for federated database systems, Computers & Security, 13 (1994), pp. 509-525.

[8] D. Jonscher and K. R. Dittrich, An Approach for Building Secure Database Federations, VLDB '94 Proceedings of the 20th International Conference on Very Large Data Base, Morgan Kaufmann Publishers Inc. (1994), pp. 24-35.

[9] Z. Tari and G. Fernandez, "Security Enforcement in the DOK Federated Database System", Pierangela Samarati, Ravi S. Sandhu (Eds.): Database Security Volume X, Status and Prospects, IFIP TC11 / WG11.3 Tenth International Conference on Database Security, 22-24 July 1996, Como, Italy. IFIP Conference Proceedings 79 Chapman & Hall 1997, ISBN 0-412-80820-X, pp. 23-42.

[10] L. Gong and X. Qian, Computational issues in secure interoperation, IEEE Trans. on Software Engineering, vol. 22, 1996, pp. 43-52.

[11] W. Eßmayr, F. Kastner, G. Pernul, S. Preishuber and A. M. Tjoa, (1995) Access Controls for Federated Database Environments. Proc. Joint IFIP TC 6 and TC 11 Working Conf. on Comm. and Multimedia Security, Graz, Austria.

[12] C. Neuman, "Why Security fails in Federated Systems", University of Southern California, 7 March 2012, http://csse.usc.edu/csse/event/2012/ARR/presentations/20120307-neuman-csse.pdf [accessed: April 2013]

[13] G. Gamble, "Data Recovery Solutions", R3 Data Recovery, [accessed: October 2012].

[14] A. Olesker, "White paper: Big Data Solutions For Law Enforcement", June 2012, http://ctolabs.com/wp-content/uploads/2012/06/120627HadoopForLawEnforcement.pdf [accessed: January 2013].

[15] P. Ravi, "Security for Big Data in Hadoop", http://ravistechblog.wordpress.com/tag/Hadoop-security/, April 15, 2013 [Retrieved - April 2013].

[16] N. Char, K. M. Siddalinga and Rahman, "Security Implementation in Hadoop", http://search.iiit.ac.in/cloud/presentations/28.pdf [accessed: January 2013].

[17] O. O'Malle, K. Zhang, S. Radia, R. Marti and C. Harrell, "Hadoop Security Design", http://techcat.org/wp-content/uploads/2013/04/Hadoop-security-design.pdf, 2009, [accessed: March 2013].

[18] D. Das, O. O'Malley, S. Radia and K. Zhang, "Adding Security to Apache Hadoop", hortonworks report, http://www.Hortonworks.com [accessed March 2013]

[19] I. Roy Srinath, T.V. Setty, A. Kilzer, V. Shmatikov, and E. Witchel, "Airavat: Security and Privacy for MapReduce", 7th USENIX conference on Networked systems design and implementation (NSDI'10), 2010, Berkeley, CA.

 ISBN: 978-1-61208-310-0