Received: September 30, 2017

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Cloud Based RDF Security: A Secured Data Model for Cloud Computing

Clara Kanmani Arulanandu[1*] Suma Vasu Deva Murthy[2] Guruprasad Nagraj[3]

[1]*Department of Computer science and Engineering, Visvesvaraya Technological University, India*
[2]*Dayananda Sagar College of Engineering, India*
[3]*New Horizon College of Engineering, India*
* Corresponding author's Email: clarakanmani0623@yahoo.com

Abstract: In this paper, a new secured data model for cloud computing is proposed which uses partial resource description framework (RDF) encryption and token based access control system in which sensitive data in an RDF graph is encrypted and all other non-sensitive data are publicly lucid. The security process, the decryption process, and query process are the three essential procedures in this framework. The consequence of the security process is the encrypted data, encrypted metadata, and plain text fragments. The proposed technique permits the token based access control system for the decryption procedure. The query process incorporates the map reduce framework is for lessening the immense measure of employments. At long last, the query answer is sent to the user in light of the access token list (AT-list) of the system administrator. Our test comes about demonstrate that, the performance of the proposed technique is assessed in view of the precision, recall and execution time of the framework. Our proposed approach is actualized using Java and keep running on Windows XP framework and the Lehigh University Benchmark (LUBM) datasets are used for our examination. In the paper this new secured RDF data model is deployed and tested using AWS elastic beanstalk.

Keywords: RDF graph, Partial encryption, Token based access control, SPARQL query, Map-reduce framework.

1. Introduction

Cloud computing assumes a noteworthy part in the IT and data processing people group [1]. These days the prominence of the cloud computing is expanding quickly, because of this reason a considerable measure of security challenges is looked by the specialist co-ops [2, 3]. For retrieving and understanding the data to both human and machine, the data are displayed in an institutionalized frame by utilizing the semantic web innovations [4, 5]. The semantically annotated data are consequently merged and assembled by the semantic web vision operators. The imperfections in the data upkeep process are explained by the standardized semantic web advancements [6, 7].

The Resource Description Framework (RDF) is the most capable standard of the semantic web because of its expressive power, semantic interoperability, and reusability. The prominence of the RDF data shows [8] and the RDF schema language (RDFS) [9] is expected to the adaptable and extensible portrayal of data under the type of triples (subjects, objects, and predicates). Creating metadata for the web is the fundamental objective of the RDF plan [10]. The mixture of many interconnected networks and PCs is known as the web, the vast majority of the presently best RDF stockpiling arrangements is bound to a solitary node [11, 12]. The large scale RDF querying is the most critical part in the RDF data management, yet the substance and the structure of the client aren't effectively comprehended by the large scale RDF [13, 14]. The RDF data security is the essential one, in light of the fact that, in the RDF data management, once in a while numerous security issues are happening in the season of encryption and decryption is finished [15, 16]. Numerous systems are utilized to take care of the security issues in the RDF data management.

The RDF access control framework [17], RDF Security based Framework [18], access control model for securing RDF triples [19] etc., are existing RDF access control methodologies. In any case, the above existing RDF data management procedures don't deal with a lot of data since all the current works are actualized in Jena which isn't productive to scale the colossal measure of data. Along these lines, a SPIDER framework [20] was utilized to deal with a lot of data; nevertheless the access control component is inadequate in this method.

Our progressed secured data model for cloud computing is the blend of partial RDF encryption and token based access control system. The need of the partial RDF encryption technique is to re-integrating the data into the RDF graph after decryption and furthermore, it is used to deal with all serialization format of the RDF graph not just the XML serialization of RDF graphs. At first, the RDF data is changed over into encrypted fragments (sensitive data) and plain text fragment (non-sensitive data). In the security process, the sensitive data in the RDF graph is encrypted and the rest of the data are publicly meaningful. Our proposed procedure executed an authentication method, to be specific token based access control system, the extra security is offered by this framework. This system also utilized for allocating AT for every security level data as indicated by agents' need and security.

The rest of the paper is delineated in the area underneath. The current research works are portrayed in section 2. The proposed approach is portrayed in section 3. The evaluation results and the conclusions are delineated in sections 4 and 5.

2. Recent research work: a brief review

Numerous research works have previously existed in literature which was based on the RDF security based access control techniques and schemes. Some of the works are reviewed here.

To deal with the issue of empty or too little answers returned from RDF query Li Yan et al. [21] have presented a query relaxation approach. In their method, the algorithm of query relaxation isn't given, so this technique does not bolster customary path queries. For complex RDF analytics, Ibrahim Abdelaziz et al. [22] has suggested a versatile framework. The limitation of this work is, the computational cost is high, in light of the fact that the cost based optimizer isn't utilized with this strategy. Zhiyuan Lin and Mahesh Tripunitara [23] have presented a threat model, and watch that the specialized test truly secluded from everything information that might be uncovered by the structure of an RDF graph. The main issue of this work is, if any conjunction happens, the structure of the graph released the information. The state-of-the-art of data partitioning and secured data partitioning in the multi cloud environment ideas are examined by Hazila Hasan and Suriayati Chuprat [24]. Nevertheless, every one of the queries is not appropriate for this model, so the data partitioning is changing, because of this reason, the execution of the framework will be decreased.

To deal with the extensive RDF graph, the SPARQL query processing systems was proposed by Lei Zou et al. [25]. But this approach may endure bring down performance because of the troubles of adjusting MapReduce to graph calculation. For the effective and scalable distributed RDF data management, the DiploCloud systems were portrayed by Marcin Wylot and Philippe Cadre-Mauroux [26]. The main drawback is, this approach makes more inter-process traffic, given that related triples, winds up being scattered on all machines.

In writing, not a lot of work is shown to handle above issues and the inconveniences of the work have influenced to do this examination work. Our proposed architecture bolsters partial RDF encryption and access control for vast data-sets by including a token based access control system. Rather than relegating access controls specifically to clients or operators, our technique produces a token is for particular access levels and allocate these token to specialists. One of the upsides of utilizing tokens is that they can be reused if the necessities and security prerequisites for different specialists are indistinguishable. Here, our method also increase the recall level and reduce the execution time and query processing time.

3. Proposed method

The fundamental point of our present work is to plan and execute a secured RDF data management in cloudy conditions. The colossal measure of RDF data management is a testing undertaking in light of their sheer size and heterogeneity. Our proposed approach for secured data model for cloud computing in terms of RDF data management first requires a security scheme, keeping in mind the end goal to encrypt the sensitive data of RDF graphs. On the following stage, the decryption process is done in light of the user query. In the last stride, an access token (AT) list is made for every agent in light of the user query. The proposed approach demonstrates the three fundamental stages to be specific; security process, the decryption process, and query process.

- In security process, the RDF sensitive data is encrypted to secure the sensitive data from the uncertain condition.
- In decryption process, the encrypted data are decrypt based on the user query.
- In query process, the access token list is created to each agent for who is allowed to use which data.

3.1 System model for proposed method

The proposed framework of secured data model for cloud is displayed in Fig. 1. The important focus of our proposed technique is to secure RDF sensitive data from the unreliable condition. Dealing with the extensive volume of RDF data is extremely troublesome, so at first, we need to do the encryption for the sensitive data previously transferring it. For the decryption process, the user presents the query request to the access control unit, if the request of the user is conceivable the query is rewritten to uphold at least one access control policies. There are three sub segments in the map-reduce framework. They are; input selector, plan generator and job execution unit. The input selector and the plan generator take the SPARQL query from the query interface engine to choosing what number of jobs are required. This data is passed to the job execution unit which sends the correct job to the public cloud. To land the best possible Position and to get the applicable policies to enforce, the public cloud is spoken with the access control unit. In the meantime, the public cloud sends the demand to the encryption container, it decrypts the RDF encrypted data in light of the user query, at last, and it transfers the decrypted data from the decryption transformation unit to the user.

3.2 Framework for secured data model

The framework of the proposed data model for cloud computing appears in Fig. 1. It incorporates three procedures; (1) security process (2) decryption process (3) query process. There are seven stages in RDF data encryption, decryption and query process.

3.2.1. Security process

(a) Fragment Selection: An RDF dataset sets of triples *(s, p, o)* from $(U \cup B) \times U \times (U \cup L \cup B)$ where *'s'* is the, *rdf : subject 'p'* is *rdf: predicate* and *'o'* is the *rdf: object*. Where U is the arrangement of URIs (Uniform Resource Identifiers) [5], the blank node identifiers are represented by B , and L means the arrangement of RDF literals. Here,

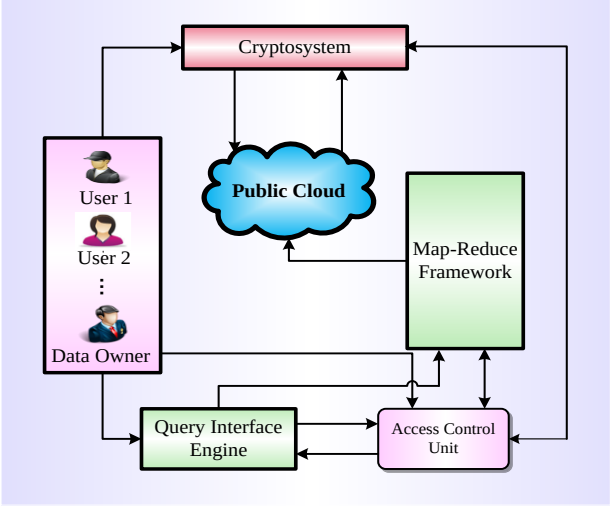

Figure.1 System model for proposed method

the RDF triples are changed over into encryption parts M_i (sensitive data) and plain text fragments M_j (non sensitive data). The Choice should be possible, for instance unequivocally specifying the sensitive data.

The sensitive data of the RDF is encrypted by using the advanced encryption algorithms like AES [28], DES [29] and RSA [30].

(b) Encryption: In encryption, all M_i is serialized and encrypted. The arrangement of client's information is encrypted by utilizing an arrangement of session keys. In the encryption process, we can consider a non-empty set of messages, encryption functions f and g. The non-empty set of messages M_i is given by,

$$M_i = \{m_1, m_2, \ldots m_n\} \quad (1)$$

The data $m_i \in M_i$ is the encrypted fragments of the encryption process. A new session key k_i is created for each data m_i. The session key k_i is used to parameterize the symmetric function f for encrypting the data M_i . So that,

$$f_{k_1}^{-1}\left(f_{k2}\left(M_i\right)\right) = M_i \quad (2)$$

Where, k_1, k_2 is the session keys. For encrypting $m_i \in M_i$ symmetrically, we need some session keys k_i. The output is *f(C, K)* where; the part of ciphertext is C and $k \in K$ is the key used to encrypt the data M_i. After finishing the symmetrical encryption process, the result is message cipher MC_i.

$$MC_i = f_k(M_i) \quad (3)$$

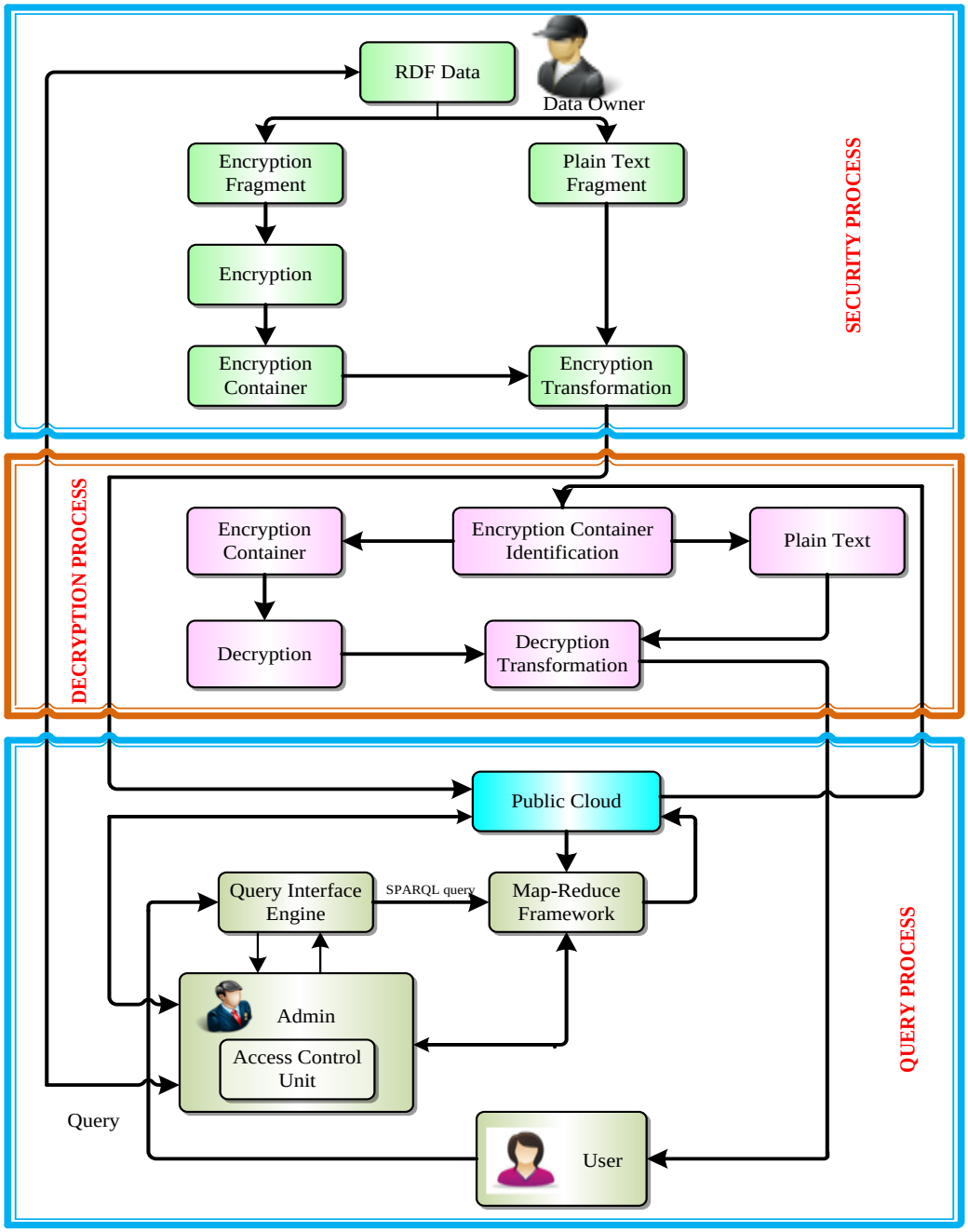

Figure.2 Framework of proposed system

For encrypting the k_i asymmetrically, it needs some public keys. The non-empty set of public keys is given by,

$$Pub = (pub_1, pub_2, \dots\ pub_n) \quad (4)$$

The result of the key ciphers is,

$$K_{ci} = k_{c1}, k_{c2}, \dots\ k_{cn} \quad (5)$$

Where, k is the key used for encrypt the data and c_i is the chipertext. The data $m_i \in M_i$ of the receiver is represented by the $p_i \in P_i$. The encrypted result C_i is the blend of encrypted data and encrypted metadata which is put away in the encryption container (EC). The encrypted metadata is the gathering of MC_i and K_c. At last consequence of the encryption process is,

$$C_i = \sum_{1=1}^{n}(MC_i, k_{c1}, k_{c2}, \dots, k_{cn}) \quad (6)$$

In the above equation, c_1, c_2,…c_n is the key ciphers. The relating encryption container supplanted all M_i and the outcome is, the RDF graph containing, encrypted data, encrypted metadata and plain text fragment.

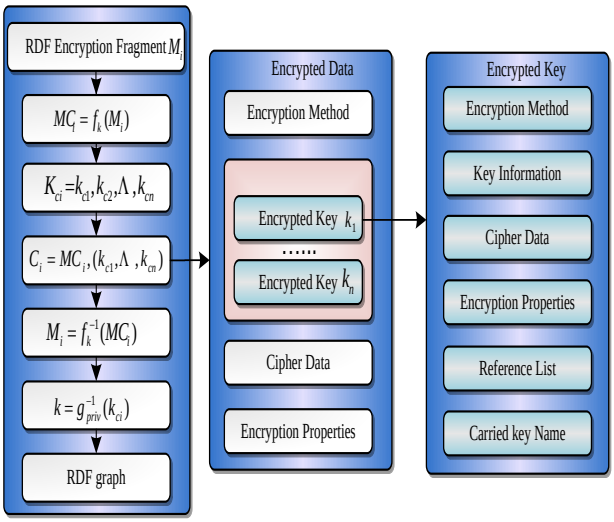

Figure.3 Structure of encryption container

(c) Encryption Transformations: All these encrypted data C_i and the plain text fragment M_j are put away in the encryption transformation. All C_i and M_j are isolated from the encryption container in light of the RDF user query for decryption. The comparing EC supplanted all the original data of RDF, at long last the RDF graph containing MC_i, M_i and plain text data M_j . Here,

$$k = g^{-1}priv\ (K_{ci}) \tag{7}$$

$$M_i = f_k^{-1}\ (MC_i) \tag{8}$$

Where k is the symmetric key, K_{ci} is the key cipher and MC_i is the message cipher f, g is the encryption function.

(d) Encryption Container Identification: The encryption container identification is utilized for isolating the M_j fragments and the encrypted data C_i for the decryption process. In this container, the encryption container and encryption metadata are recognized in light of the RDF query language.

3.2.2. Decryption process

(a) Decryption: In the decryption step, all the EC's data are decrypted in light of the user query prerequisites. The parameter of the encrypted metadata is utilized for decrypting the EC's data. The encryption container data are decrypted in light of the receiver's key parameter. The decryption functions g_f^{-1} and f^{-1} parameterized with *priv* to decrypting C_i for recovering the session key k to the receiver. The agent handles some *priv* key pub_1, pub_2 , … pub_n key for decrypting the ECs.

The decryption crashes and burns, on the off chance that a specialist does not have a proper key for decryption. Here, the reconstructed RDF graph is equivalent to the original RDF graph.

(b) Decryption Transformations: The decrypted data D_i and the plain text fragments are put away in the encryption transformation unit. The corresponding decrypted estimation of the ECs is reconstructed for RDF graph. The D_I will be the opposite of the C_i.

3.2.3. Cryptosystem for RDF fragments encryption

The cryptosystem utilized some cryptographic algorithm for an RDF data security benefit. They are; (1) key generation, (2) encryption and (3) decryption. Experimentally, a design or cryptosystem is portrayed as a tuple is *(P, C, K, ε, D)*, where, *'P'* is known as the plain content space and the segments of *'P'* is known as plaintexts, '*C'* is known as the cipher text space and its parts is known as the cipher texts, key space is connoted as *'K'* and its components is known as key. The set of encryption and decryption functions are characterized by,

$$\varepsilon = \{E_k : k \in K\} \tag{9}$$

$$D = \{D_k : k \in K\} \tag{10}$$

The component of the above function is known as the encryption component (ε) and decryption component (D). The encryption, decryption components is given by,

$$E_k : P \to C \tag{11}$$

$$D_k : P \to C \tag{12}$$

Here, if the cryptosystem is symmetric $k_e = k_d$, if the cryptosystem is asymmetric $k_e \neq k_d$.

3.2.4. Query process

In this last stride, the decryption is done in view of the users require. The query process incorporates users, query interface engine, access control unit and map reduce framework. The user sends the RDF query language to the admin through the query interface engine, the admin chooses if the users query request is conceivable or not on the off chance that it is conceivable and sends the input signal to the user. A lot of dataset is changed over into the little dataset by utilizing MapReduce framework, it likewise used for adjusting the SPARQL query of the user.

(a) SPARQL Query: SPARQL is the standard World Wide Web Consortium (W3C) proposal for querying RDF graph.

In the query process, the triple pattern *(s, p, o)* is generalized from the $(U \cup B \cup V) \times (U \cup V) \times (U \cup L \cup B \cup V)$, where the set of factors are represented as V. The subset of SPARQL query is the BGP (Basic Graph Pattern) queries of SPARQL. For instance, the client requests the verse's lyrics, which are shown in Museums situated in Newyork. The syntax of this query process is written as,

SELECT $? v_1 \ldots ? v_m$ *WHERE* $\{t_{p1} \ldots t_{pn}\}$
SELECT *?q* *?r*
WHERE *{*
?p : type : poetry
?p : poem ? q
?q : exhibited ? z
?z : location : newyork }

Here, $t_{p1} \ldots t_{pn}$ is the triple pattern and $? v_1 \ldots ? v_m$ is the arrangement of variable got from the triple pattern. The variable *?p* is the gathering of initial two triple patterns. The initial two triple patterns are joined on factor *?p*, the second and third variables are *?q* and the last two on variable *?r* . The last three triple patterns shape a way sub query.

(b) Map-Reduce Framework: The MapsReduce structure used to diminish the immense measure of jobs (i.e., changed over a great deal of Jobs into a little measure of the job). Here, each job has two phases, they are mapped and reduce. The key values have gone about as the commitment of the map process and the yield values are assembled in the reduction phase. In the event that any correspondence is happening between the maps and reduce stage, the speed and straightforwardness of the entire procedure are deficient. The Map-Reduce framework has three fundamental segments. They are input selector, plan generator and join executer.

The rewritten SPARQL query of the query interface engine is given to the input selector and plan generator to choose what number of jobs are needed from the input file. This information is passed to the job executer unit. For runs the job legitimately some applicable policies are required. To land those policies the job executer component communicates with the access control unit. At that point the query answer is sent from the public cloud to the user.

(c) Access Control Unit: Our proposed technique executes a token-based access control system. Here, the Access Token (AT) is dispensed by the admin of the framework for every security relevant data as per AGENTS' needs and security level. On the off-chance that at least one AT is assigned to a similar agent, the contention happens in the system. The time stamp based conflict identification and resolution algorithm is utilized for maintaining a strategic distance from the conflict happened in the system.

Access Token Assignment and Policies: The access of the security relevant data is accumulated from the AT. In view of the timestamp distinguishing, the admin assigned at least one AT's to the given agent, this number of AT's are known as the AT-list. In this procedure every agent has a different AT-list. In the event that any amendment is happening in the AT-list, the admin remedy them. The underlying AT-list and the timestamp are put away in the *TempAT* (impermanent variable). The conflict happened in the new AT-list is recognized before submitting the revision. In the last yield of the agent's ATs, each set of triples accumulated access from the AT, this the ATs result set. Here, the outcome set of the ATs are communicated as,

$$Z = Z_1, Z_2, \ldots, Z_n \quad (13)$$

Then the AT-list of the agents are expressed as,

$$AT = AT_1, AT_2, \ldots, AT_n \quad (14)$$

Then the triple set of the agent is expressed as,

$$ATT = Z_1 \cup Z_2 \cup \ldots \cup Z_n \quad (15)$$

For changing the SPARQL inquiry, the Map-Reduce framework utilizes a few policies. Query rewriting, embedded enforcement and post processing enforcement are the three primary enforcement policies. By utilizing the Map-Reduce, the policies are enforced at the season of data choice in embedded enforcement; they chose data are enforced amid the introduction of data to users. For diminishing quantity of jobs, the embedded enforcement reliably outflanks post processing enforcement.

4. Evaluation results and discussions

4.1 Experimental configuration

Our approach is actualized utilizing Java and run on a Windows XP system and the datasets of our investigation is taken from the Lehigh University Benchmark (LUBM). There are 14 standard queries

in the LUBM dataset. The LUBM dataset is generally utilized for testing scalability and impedance. A portion of the queries requires a derivation to reply. The created dataset is the RDF triples. In our proposed system, we execute a propelled partial encryption and token based access control system for RDF data management. In query rewriting, embedded enforcement, and post processing enforcement is the principal approaches in our proposed technique. Here, the embedded enforcement approach is contrasted and the post processing enforcement approach for the examination. The LUBM10, LUBM100, LUBM1000, LUBM2000, LUBM5000, LUBM10000 and LUBM15000 data sets are utilized for our examinations.

4.2 Security analysis

The security of our proposed approach is assessed by the satisfaction of the security ensure. The attacks, data leakage, conflict, modification, and privacy of users are specified in this segsment. Our proposed approach is intended to handle all these security issues productively.

4.2.1. Security level defaults

The admin allowed AT for every Agent, in some cases the conceded AT is a weight, because of this reason the default security level is allocated to every data in the framework for taking care of the AT trouble issue. Our proposed strategy executes a token based access control system for settling the security level defaults. In our proposed strategy each one of the data in the system has one default security level. Here, all the security level data (sensitive data) of the individual is secured in private by denying. This shields operators from influencing inductions about any individual to whom they to have not been surrendered unequivocal assent. In any case, if an operator is yielded explicit access to a particular sort or property, the specialist is moreover permitted default access to the sub types or sub-properties of that sort or property.

For instance, we can accept that the predicate document is *interest* that lists elements that a person's interest. Accept encourage that *interest* of *Tom (person)* is *playing* and 1 is an access token of ATTs *(1, Subject , URI, Tom)* and *(1, interest, Predicate, _)*. Of course, agent *Dani* having just AT 1 can't discover that *Kim* is in *Tom's interest* - list since *Kim's* data type is *persons*. Be that as it may, if *Dani* likewise has AT 2 depicted by ATT *(2, object, URI, Kim)*, then *Dani* will have the capacity to see *Kim* in *Tom's interest* –list.

4.2.2. Subset and subtype conflicts

A conflict emerges when the accompanying three conditions happen: (1) An agent has two AT's 1 and 2, (2) the result set of AT 2 is an appropriate subset of AT 1, and (3) the timestamp of AT 1 is sooner than the timestamp of AT 2. In this situation, the latter, more particular AT supersedes the previous, so AT 1 is disposed of from the AT-list to determine the conflict. Such conflicts emerge in two assortments, which we term subset conflicts and subtype conflicts.

If the timestamp of AT 1 is sooner than the timestamp of AT 2, the subset conflict occurred. In the subtype conflict approach, the dataset is most subjects, predicates or both. Let as consider, in ATT the AT 1 is characterized by *(1, Subject, URI, Tom)* and AT 2 is characterized by *(2, Subject, URI, Tom)* and *(2, Predicate, interest, _)*. At that place the conflict will occur because the set AT 2 is a subset of AT 1. Here, the time stamp of AT 1 is sooner than AT 2. To avoid this type of conflict the set AT 1 is removed from the AT-list. If the set AT 2 is the subset of AT 1, the subtype conflict may occur. The dataset is mostly subjects, objects, or both. Here, Subset (AT 1, AT 2) is a function that returns true if the result set of AT 1 is a proper subset of the result set of AT 2, and Subject Sub Type (AT 1, AT 2) returns true if the subject of AT 1 is a subtype of the subject of AT 2. Similarly, Object Subtype (AT 1, AT 2), decides sub typing relations for objects instead of subjects.

4.2.3. Brute force attack

The security level data are assaulted by different unauthorized interceptors amid the season of data exchanged to cloud over an internet network. To dispose of this attack, our approach utilized the partial encryption technique; this encryption will encrypt the security level data. Here, the number of keys is used for the partial encryption method. The encrypted data are stored in the private cloud; the remaining non-sensitive data are publicly readable. Our proposed methodology used more bits and key bits for the partial encryption, but the existing SSL encryption method used only 40 bits for the encryption. At this time, our partial encryption method makes the brute force attack mostly useless. Our proposed method uses the partial encryption and token based access control system for the RDF data security. This method does not support to the other attackers to view the secured level data. Hence this approach not only secures the data, it also gives

Table 1. Functionality analysis

Functions	Partial Encryption method [31]	State of art method [30]	Cryptographic method [29]	Proposed Method
Identification	No	Yes	Yes	Yes
Authorization	Yes	No	Yes	Yes
Confidentiality	No	Yes	No	Yes
Integrity	No	Yes	No	Yes
Partial Encryption	No	Yes	Yes	Yes
Token Based Access Control	Yes	No	Yes	Yes

promise to the customer, and the data is secured at the time of transferring.

4.3 Functionality analysis

In RDF data management, our proposed method implements an efficient partial RDF encryption for overcomes all the possible issues occurred at the time of transferring the data to the cloud. This method secures all the data from the risks associated. The comparison of functional analysis of the proposed method is compared with the other security based method shown in Table 1.

Our technique has the functions, for example, identification, authorization, confidentiality, integrity, partial encryption, token based access control functions. Along these lines, the present model demonstrates that the closeness of most practical security inconveniences by giving functions and to allow the security issues are sufficient in this method.

4.4 Performance analysis

The performance of our proposed approach is evaluated based on the precision, recall and execution time of the system. In the RDF data management, the query process is based on the SPARQL query. In the appraisal of execution time, the RDF query language is used to determine the precision and recall of the system. The precision and recall is computed depends on the number of triples in the query answer and number of triples in the dataset. Recall is the ratio of number of triples in query should have been returned divided by the total number of results that triples in the dataset.

$$Recall = \frac{Number\ of\ triples\ in\ query\ answer}{Number\ of\ triples\ in\ dataset} \quad (16)$$

The recall of each query is shown in the Fig. 4.

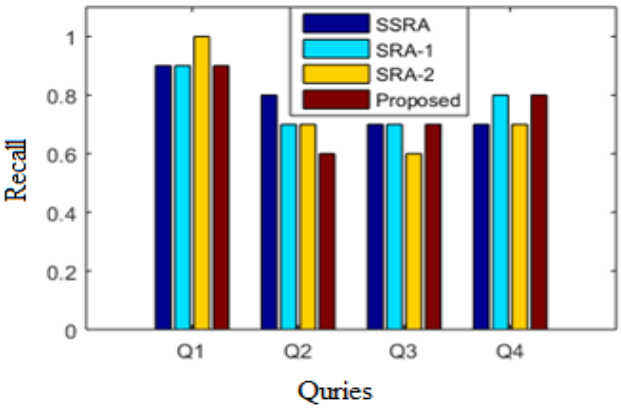

Figure.4 Recall comparison of our proposed method with other existing method

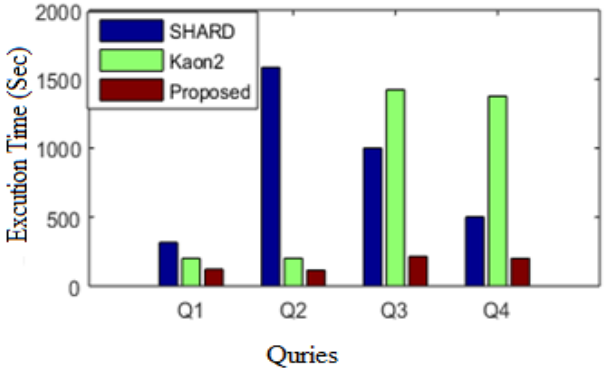

Figure.5 LUBM execution time

Here, our proposed method is compared with existing SSRA, SRA1, and SRA2 methods [35]. In query Q1, Q2, Q3, and Q4, the recall of our proposed method is 0.7, 0.8, 0.7 and 0.8 respectively. But the existing SSRA method recall values are 0.9, 0.9, 1 and 0.9 respectively. The recall of SRA1 is 0.8, 0.7, 0.7 and 0.6.

The recall of SRA2 is 0.7, 0.7, 0.6 and 0.7 respectively. Our proposed method has less recall than the existing method.

The total execution time of each query is shown in the Fig. 5. Here, our proposed method is compared with existing SHARD [34] and the Kaon2 [33] method. In query Q1, Q2, Q3, and Q4, the execution time of our proposed method is 122.568(s), 115.277 (s), 215.478 (s) and 199.755 (s) respectively. But the existing SHARD execution time is 316.2278 (s), 1584.893 (s), 1000 (s) and 501.1872 (s). The execution time of Kaon2 method is 201.54 (s), 201.54 (s), 1424.502 (s) and 1375.12 (s). Our proposed method has a better execution time than the existing method.

4.5 Query processing

The query processing time of our proposed

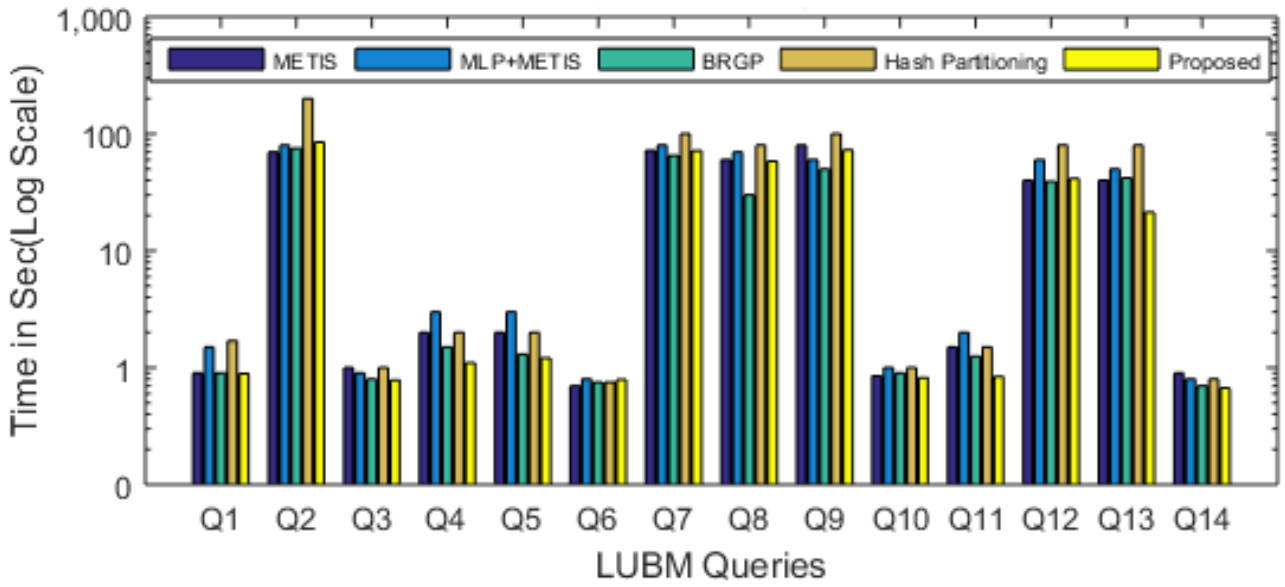

Figure.6 Query processing time

strategy is contrasted and the current techniques, for example, Balanced RDF Graph Portioning (BRGP) [36], Hash Partitioning Algorithm, METIS, a mix of METIS and Multilevel Label Propagation [36] (MLP+METIS) algorithm. This query processing time appears in Fig. 6. There are 14 benchmark LUBM queries Q1-Q14. The LUBM queries consolidate subject and subject-joins. Since there is no network correspondence the queries are executed with the handling nodes in light of the way that the best detachment between two nodes is only two. The test result exhibits that our proposed system has the better preparing time in 14 LUBM queries in differentiated and the four existing partitioning techniques. Differentiated and the graph partitioning technique, the query preparing time is high in the hash partitioning system. Our proposed procedure has remarkably extended the efficiency of the query procedure. In any case, the other existing strategy has the better execution time, yet it stores the duplicates triples.

The attacks, data leakage, conflict, modification, and privacy of users are managed by our technique such as identification, authorization, confidentiality, integrity, partial encryption, token based access control functions. Due to these above techniques, our proposed method has better recall, execution time and query processing time when compared with the existing works [31, 32].

5. Conclusion

The access control system for RDF data administration has executed in segment 2. Be that as it may, the current audit strategies are not appropriate for scaling the extensive data sets. Step by step, the volume of the RDF data in the cloud is expanding quickly. To address these issues the RDF data is secured by the cloud PCs. By and by, the cloud computers do not completely comprehend the above issue. Along these lines, we need to propose the token based access control system and partially encryption strategy for dealing with the RDF data. At first, the RDF sensitive data is encrypted to secure the sensitive data from the unverifiable condition and afterward the access token gifts to every client in view of their level of authorization. For reversing the SPARQL query, the MapReduce framework utilizes query rewriting, embedded enforcement and post processing enforcement policies.

The execution of our proposed secured data show for cloud computing is assessed in light of the precision, recall and execution time of the framework or more perception in the earlier area demonstrates that it is enhanced contrast and existing data models. In the paper this new secured RDF data model is deployed and tested using AWS elastic beanstalk. For future work, we will enhance the SPARQL query ideas for controlling the query procedure. For consolidating the SPARQL query and graph analytical in a similar query we will consider the cross optimization algorithm. To assess the numerous algorithms simultaneously, we will execute the multiple query optimization systems.

References

[1] R. Moreno-Vozmediano, R. Montero, E. Huedo and I. Llorente, "Cross-Site Virtual Network in

Cloud and Fog Computing", *IEEE Cloud Computing*, vol. 4, No. 2, pp. 46-53, 2017.

[2] P. Li, J. Li, Z. Huang, C. Gao, W. Chen, and K. Chen, "Privacy-preserving outsourced classification in cloud computing", *Cluster Computing*, pp. 1-10, 2017.

[3] K. Arvind and R. Manimegalai, "Secure data classification using superior naive classifier in agent based mobile cloud computing", *Cluster Computing*, vol. 20, No. 2, pp. 1535-1542, 2017.

[4] J. Aikat, A. Akella, J. Chase, A. Juels, M. Reiter, T. Ristenpart, V. Sekar, and M. Swift, "Rethinking Security in the Era of Cloud Computing", *IEEE Security & Privacy*, Vol. 15, No. 3, pp. 60-69, 2017.

[5] Q. Cao, D. Schniederjans, and M. Schniederjans, "Establishing the use of cloud computing in supply chain management", *Operations Management Research*, Vol. 10, No. 1-2, pp. 47-63, 2017.

[6] A. Salem, J. Shaffer, R. Kublik, L. Wuertemberger, and D. Satko, "Microstructure-Informed Cloud Computing for Interoperability of Materials Databases and Computational Models: Microtextured Regions in Ti Alloys", *Integrating Materials and Manufacturing Innovation*, Vol. 6, No. 1, pp. 111-126, 2017.

[7] R. Ferreira da Silva, R. Filgueira, I. Pietri, M. Jiang, R. Sakellariou, and E. Deelman, "A characterization of workflow management systems for extreme-scale applications", *Future Generation Computer Systems*, Vol. 75, No. 10, pp. 228-238, 2017.

[8] H. Lin, J. Hu, Y. Tian, L. Yang, and L. Xu, "Toward better data veracity in mobile cloud computing: A context-aware and incentive-based reputation mechanism", *Information Sciences*, Vol. 387, No. 1, pp. 238-253, 2017.

[9] L. Yan, R. Ma, D. Li, and J. Cheng, "RDF approximate queries based on semantic similarity", *Computing*, Vol. 99, No. 5, pp. 481-491, 2017.

[10] A. Oudani, M. Bahaj, and I. Cherti, "Creating an RDF Graph from a Relational Database Using SPARQL", *Journal of Software*, Vol. 10, No. 4, pp. 384-391, 2015.

[11] I. Santana-Perez, R. Ferreira da Silva, M. Rynge, E. Deelman, M. Pérez-Hernández, and O. Corcho, "Reproducibility of execution environments in computational science using Semantics and Clouds", *Future Generation Computer Systems*, Vol. 67, No. 2, pp. 354-367, 2017.

[12] S. Habib, S. Ries, M. Mühlhäuser, and P. Varikkattu, "Towards a trust management system for cloud computing marketplaces: using CAIQ as a trust information source", *Security and Communication Networks*, Vol. 7, No. 11, pp. 2185-2200, 2013.

[13] A. Frankel, G. Iaccarino, and A. Mani, "Optical depth in particle-laden turbulent flows", *Journal of Quantitative Spectroscopy and Radiative Transfer*, Vol. 201, No. 16, pp. 10-16, 2017.

[14] J. Li, Y. Liu, and J. Zhong, "3D DWT-DCT and Logistic MAP Based Robust Watermarking for Medical Volume Data", *The Open Biomedical Engineering Journal*, Vol. 8, No. 1, pp. 131-141, 2014.

[15] P. Singh and B. Raman, "Reversible data hiding for rightful ownership assertion of images in encrypted domain over cloud", *AEU - International Journal of Electronics and Communications*, Vol. 76, No. 6, pp. 18-35, 2017.

[16] Y. Verginadis, A. Michalas, P. Gouvas, G. Schiefer, G. Hübsch, and I. Paraskakis, "PaaSword: A Holistic Data Privacy and Security by Design Framework for Cloud Services", *Journal of Grid Computing*, Vol. 15, No. 2, pp. 219-234, 2017.

[17] J. Kim, "Efficient Authorization Conflict Detection Considering RIF Inference in RDF Access Control", *The Journal of Korean Institute of Information Technology*, Vol. 12, No. 5, pp. 197-198, 2014.

[18] A. Jain and C. Farkas, "Secure resource description framework", In: *Proc. of the eleventh ACM symposium on Access control models and technologies - SACMAT '06*, Lake Tahoe, California, USA, pp. 121-129, 2006.

[19] J. KIM and S. PARK, "RDFacl: A Secure Access Control Model Based on RDF Triple", *IEICE Transactions on Information and Systems*, Vol. 92-, No. 1, pp. 41-50, 2009.

[20] H. Choi, J. Son, Y. Cho, M. Sung, and Y. Chung, "SPIDER", In: *Proc. of the 18th ACM conference on Information and knowledge management - CIKM '09*, Hong Kong, China, pp. 20187-2088, 2009.

[21] L. Yan, R. Ma, D. Li, and J. Cheng, "RDF approximate queries based on semantic similarity", *Computing*, Vol. 99, No. 5, pp. 481-491, 2017.

[22] I. Abdelaziz, M. Al-Harbi, S. Salihoglu, and P. Kalnis, "Combining Vertex-centric Graph Processing with SPARQL for Large-scale RDF Data Analytics", *IEEE Transactions on Parallel and Distributed Systems*, Vol. PP, No. 99, pp. 1-1, 2017.

[23] Z. Lin and M. Tripunitara, "Graph Automorphism-Based, Semantics-Preserving Security for the Resource Description Framework (RDF)", In: *Proc. of the Seventh ACM on Conference on Data and Application Security and Privacy - CODASPY '17*, Scottsdale, Arizona, USA, pp. 337-348, 2017.

[24] H. Hasan and S. Chuprat, "Secured data partitioning in multi cloud environment", In: *Proc.of the 4th World Congress on Information and Communication Technologies (WICT 2014)*, Bandar Hilir, Malaysia, pp. 146-151, 2014.

[25] P. Peng, L. Zou, M. Özsu, L. Chen, and D. Zhao, "Processing SPARQL queries over distributed RDF graphs", *The VLDB Journal*, vol. 25, No. 2, pp. 243-268, 2016.

[26] M. Wylot and P. Cudre-Mauroux, "DiploCloud: Efficient and Scalable Management of RDF Data in the Cloud", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 28, No. 3, pp. 659-674, 2016.

[27] A. Adeshina and R. Hashim, "Computational Approach for Securing Radiology-Diagnostic Data in Connected Health Network using High-Performance GPU-Accelerated AES", *Interdisciplinary Sciences: Computational Life Sciences*, Vol. 9, No. 1, pp. 140-152, 2016.

[28] I. Bhargavi, D. Veeraiah, and T. Maruthi Padmaja, "Securing BIG DATA: A Comparative Study Across RSA, AES, DES, EC and ECDH", In: *Proc. of the Computer Communication, Networking and Internet Security*, Springer, Singapore, pp. 355-362, 2017.

[29] P. Patil, P. Narayankar, D.G. Narayan, and S.M. Meena, "A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish", *Procedia Computer Science*, Vol. 78, No. 1, pp. 617-624, 2016.

[30] S. Cebiric, F. Goasdoue, and I. Manolescu, "Query-oriented summarization of RDF graphs", In: *Proc. of the 41st International Conference on Very Large Data Bases*, Kohala Coast, Hawaii, pp. 2012-2015, 2015.

[31] M. Giereth, "On Partial Encryption of RDF-Graphs", In: *Proc. of the 4th International Semantic Web Conference*, Galway, Ireland, pp. 308-322, 2005.

[32] K. Arindam, H. Mohammad Farhan, K. Latifur, H.W. Kevin, and T. Bhavani, "A token-based access control system for RDF data in the clouds", In: *Proc. of the Second International Conference on Cloud Computing Technology and Science*, Indianapolis, IN, USA, pp. 104-111, 2010.

[33] R. Punnoose, A. Crainiceanu, and D. Rapp, "SPARQL in the cloud using Rya", *Information Systems*, Vol. 48, No.2, pp. 181-195, 2015.

[34] K. Rohloff and R. Schantz, "High-performance, massively scalable distributed systems using the MapReduce software framework", In: *Proc. Of the Programming support innovations for emerging distributed applications*, Reno, Nevada, pp. 4, 2010.

[35] C. Lee, S. Park, D. Lee, J. Lee, O. Jeong, and S. Lee, "A comparison of ontology reasoning systems using query sequences", In: *Proc. of the 2nd international conference on Ubiquitous information management and communication - ICUIMC '08*, Suwon, Korea, pp. 543-546, 2008.

[36] Y. Leng, C. Zhikui, F. Zhong, X. Li, Y. Hu, and C. Yang, "BRGP: a balanced RDF graph partitioning algorithm for cloud storage", *Concurrency and Computation: Practice and Experience*, Vol. 29, No. 14, p. 3896, 2016.

[37] U. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks", *Physical Review*, Vol. 76, No. 3, p. 036106, 2007.