Integration of Role Based Access Control with Homomorphic Cryptosystem for Secure and Controlled Access of Data in Cloud

Kamalakanta Sethi
Indian Institute of Technology Bhubaneswar
Orissa, India
ks23@iitbbs.ac.in

Anish Chopra
Indian Institute of Technology Bhubaneswar
Orissa, India
ac14@iitbbs.ac.in

Padmalochan Bera
Indian Institute of Technology Bhubaneswar
Orissa, India
plb@iitbbs.ac.in

Bata Krishna Tripathy
Indian Institute of Technology Bhubaneswar
Orissa, India
bt10@iitbbs.ac.in

ABSTRACT

Recent advances in cloud technology facilitates data owners having limited resources to outsource their data and computations to remote servers in Cloud. To protect against unauthorized information access, sensitive data are encrypted before outsourcing. However, traditional cryptosystems need decrypting ciphertext for outsourced computations that may violate data security as well may introduce higher computational complexity. Homomorphic encryption is a solution that allows performing computations directly on ciphertext. On the otherhand, it is evident that the computations on data may vary from users to users depending on the requirements. So, it is not always feasible to allow all computations to different users on the whole ciphertext stored in cloud. In this paper, we proposed a framework for integration of role based access control (RBAC) mechanism with homomorphic cryptosystem for secure and controlled access of data in cloud. Our proposed framework is developed based on trust and role hierarchy with multi-granular operational access rights to heterogeneous stakeholders or users.

CCS CONCEPTS

• **Security and privacy** → **Security services**; **Access control**;

KEYWORDS

Homomorphic encryption, RBAC, Trust, Outsorced computation, Cloud

ACM Reference Format:
Kamalakanta Sethi, Anish Chopra, Padmalochan Bera, and Bata Krishna Tripathy. 2017. Integration of Role Based Access Control with Homomorphic Cryptosystem for Secure and Controlled Access of Data in Cloud. In *Proceedings of SIN '17.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3136825.3136902

SIN '17, October 13–15, 2017, Jaipur, IN, India

ACM ISBN 978-1-4503-5303-8/17/10…$15.00
https://doi.org/10.1145/3136825.3136902

1 INTRODUCTION

Cloud platform provides robust computing power to the society with reduced cost. One fundamental advantage enabled by cloud is outsourcing of computation. In the context of outsourced computation, data owners having inadequate computational resource can outsource their task with data to a server in the cloud. However, this outsourcing mechanism compromise the privacy of the outsourced data and the integrity of the computation, since the server may not be fully-trusted. Therefore, sensitive data has to be encrypted before outsourcing and computations can be performed on the encrypted data. Homomorphic encryption is a potential solution to achieve this[7].

It is not always feasible for data owners to perform computations on ciphertext. So there is a need for data owners to share the data to different users with granularity. Then the users can perform necessary operations on data which are only allowed by the data owner. This motivates us to integrate Role Based Access Control (RBAC) with Homomorphic encryption.

For secure and controlled access of data, many approaches have been developed. One of the well accepted approach is access control methods. The commonly used access control mechanisms used are Mandatory Access Control (MAC), Discretionary access control (DAC) and Role-based access control (RBAC). Role-based access control is well known model for providing restricted access to authorized users. In RBAC, roles are used to associate users with permission on resources. Users are assigned roles and permissions are allocated to roles instead of individual users. The users who have been granted membership to roles can access the permissions associated with the roles and hence can access the resources.

In our paper, we have proposed a RBAC mechanism for the purpose of integrating with Homomorphic cryptosystem. The proposed framework is based on trust and role hierarchy. In our model, the trust value of the role is calculated, when data owners with a particular role want to store data in cloud.

If the trust value of role is above a threshold (set by the data owner), then the data owner can store the ciphertext on the cloud. Permissions on data are allocated to specific roles instead of individual users. Roles grant membership to users, whose trust value is more than threshold (set by role). The users can access the permissions associated with the roles and hence can access the resources.

The rest of the paper is organized as follows. Section 2, describes the related work. In Section 3, our proposed RBAC system integrated with Homomorphic cryptosystem has been presented. Work flow of our proposed system has been presented in Section 4. The efficacy of our proposed framework with experimental results has been presented in Section 5. Finally, we concluded in Section 6.

2 RELATED WORK

The concept of Homomorphic encryption was first proposed by Rivest et al. [7] after the discovery of public key cryptography. A majority of the known public key cryptosystems, i.e., RSA, Elgamal, Pailler, etc. support homomorphic addition or multiplication on ciphertext but not both [8]. These are partially homomorphic cryptosystems [6]. However, the partially homomorphic encryption is not sufficient for various practical environments such as finance and statistical applications like online voting system, multiparty computations, etc. This is because of the fact that these applications require both addition and multiplication operation for realizing various computations.

In 2009, Gentry proposed Fully homomorphic encryption (FHE) [2] which first constructs an SHE (somewhat homomorphic encryption) and then converts the same to an FHE which in turn can evaluate circuits of arbitrary depth. This approach uses bootstrapping procedure for the conversion from SHE to FHE. However, the computations in SHE based scheme adds noise to the ciphertext in each iteration and thereby makes it difficult to retrieve the original plaintext after the noise exceeds the key-dependent threshold. Gentry uses bootstrapping procedure to reduce this noise. However, this procedure introduces high overhead and thus making the cryptosystem computationally inefficient.

A number of schemes have been proposed based on Gentry's approach. One of these approach is DGHV scheme-2010 [10] which implements homomorphic encryption on integers and not on lattices. The basis behind the security of this scheme lies on the hardness of approximate GCD problem [3].

It has been studied that the Gentry's scheme is impractical for use in real time applications[5]. Kamara et al. from Microsoft research proposed the idea of parallel homomorphic encryption(PHE) [4]. This proposal uses MapReduce framework for realizing parallel computations on ciphertext.

In 1992, D.F. Ferraiolo and D.R. Kuhn proposed a formal RBAC model [1], that defines RBAC with access only through roles, hierarchies and specific constraints. In 2015, Lan Zhou et al.[11] proposed a RBAC model which takes feedbacks of roles and users into account and addresses the issue of trust in RBAC model.

In our previous work [9], we have developed a novel implementation of parallel homomorphic encryption for secure data storage in cloud. The basis of our parallel implementation is multithreading. In that work, a data owner can perform different benchmark computations like addition, multiplication, division, searching etc. efficiently on encrypted data. However, that approach does not incorporate the secure and multi-granularity sharing of data and computations to the authorized users depending on their requirements. In addition, it is not always necessary to allow all computations to different users on the whole ciphertext stored in cloud. This challenges drives our current work to integrate Role Based Access Control (RBAC) with our proposed cryptosystem[9].

The next section describes our proposed RBAC system integrated with Homomorphic cryptosystem.

3 PROPOSED RBAC SYSTEM ARCHITECTURE

Our proposed RBAC system includes four entities namely administrator, roles, users, and data owners. The administrator is the certificating authority of the system. It can add new roles, can manages role hierarchical structure, can shut down and start the whole cloud storage whenever something goes wrong in the cloud. It is evidence that a large numbers of authorized administrators and users manage and assess stored data in large-scale cloud system. Therefore, many roles are assigned to manage the cloud storage.

Our proposed cloud storage consists of following five parts:

1.Cloud Parameters: These parameters such as role hierarchy are used to manage access to the data in cloud storage. These are managed by the administrator.

2. Users Management: This includes management process of the users. User management process is driven by roles.

3. Data Section: This section records all the data stored by the data owners. In cloud storage, private section is a part of data section which is used to store computed data. These computed data can only be accessed by the requesting user.

4. History Section: All the interaction histories of roles and users are stored in History section. The section can be accessed publicly by entities of system. Here, users history indicate access history, i.e. which files are accessed by which users on what time. Data owners history means upload history, i.e. which files are binded to which role on what time by which data owner.

5. Feedback Section: This section is used to store trust records of the users as well as roles. These trust records are used to evaluate trust value of users and roles

Roles are the entities that set the relationship between users and data owners. A role can give membership to new users in its role. A role can remove users as well and share information about file leakage in the system.

Data owners are the parties, who store the data in the cloud for users. Whenever a data owner (DO) stores the data in the cloud, he or she specifies who can access the data in terms of role-based policies, and which operations can be done by these users. In the RBAC model, they are the parties who manage the relationship between permissions and roles.

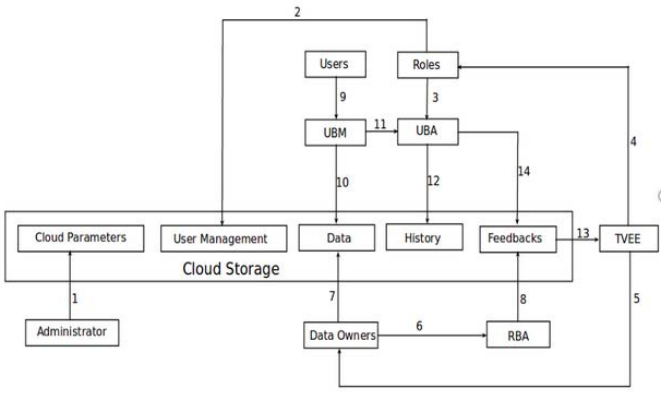

Figure 1: System Architecture

Users are the parties, who want to access the data stored in the cloud. A user can access the data, if he or she is allowed by the data owner and perform computations.

In addition to these four entities, our proposed RBAC system contains a trust management module, which includes four components to store feedbacks, to evaluate trust value and to monitor users' activity. These components are described as follows

1.User Behavior Monitor (UBM): UBM is used to monitor users activities. It acts as proxy server between users and cloud. When a user wants to send a request to access resources stored in the cloud, he or she does not send the request directly to the owners of resources. Instead, the request is sent to the UBM which monitors and passes the request to the owner of the file, if the user is authorized. This component informs the user behavior auditor (UBA) about the access history of users and UBA stores this History section.

2.User Behavior Auditor (UBA): User behavior auditor is used to collect the feedbacks for users from roles on users' behavior. Roles give leakage information to UBA which accesses the History section and determines whether a user is involved in the leakage of data and update the records in Feedback section. The UBA uses the information in History section to determine leakage information.

3.Role Behavior Auditor (RBA): It is used to collect feedbacks for roles from authorized data owners and feedbacks are forwarded to Feedback section. It is also used to collect information about data assignment to role by data owner. When a data owner stores encrypted data into a role, then RBA monitors that by which data owner, the data is stored in which role.

4.Trust Value Evaluation Engine (TVEE): TVEE is used to evaluate trust value of users and roles. TVEE accesses trust records stored in Feedback section and evaluates trust value of users and roles.

The work flow of our proposed system is shown in Figure 1. The next section explains the overall workflow of the proposed RBAC model.

4 SYSTEM WORKFLOW

In this section , we present the workflow shown in Figure 1. All the entities of our proposed RBAC model integrated with Homomorphic cryptosystem as discussed in Section 3 are connected through different duplex communication channels which are labeled with numbers to indicate sequential flow of activities. The detailed System Workflow is presented as follow.

The Administrator initializes system and specifies cloud parameters such as role hierarchy and uploads these via channel 1. Whenever a role is initialized by the system administrator, role hierarchy is updated by the administrator via channel 1.

Roles grant the membership to users and upload it into user management section via channel 2. Roles give file leakage information to UBA via channel 3. Roles request for trust value evaluation for users to TVEE via channel 4. Then TVEE accesses trust records of users stored in Feedback section via channel 13 and evaluates the trust value of users and sends to roles via channel 4. Roles can remove users, if the trust value is below threshold trust value. If a role removes a user then the role updates in user management section via channel 2.

It is necessary to assign a particular file to a role. This to enable users with valid membership of roles accessing the file and perform the operations allowed on the file. Therefore, mapping between files, users and roles is important in our proposed RBAC system. Figure 2 shows the different types of mapping in our proposed RBAC system, which are discussed in detail in the following subsections.

4.1 Role-File Mapping

Files are resources that are stored in cloud servers and computations are performed by users on it. Thus, we shall use file and resource interchangeably in this section. The role to file (resource) mapping [refer Figure 2] is a many to many association. A file can be mapped to many roles and a role can be mapped to many files. This mapping is controlled by data owners. The data owner assign file to role and specify the set of operations that can be performed by the user of the role. For example, a data owner assigns a file F_1 to the role R_2 and data owner wants to allow only addition, multiplication and division operation to the user of the role R_2. The data owner specifies these operations and maps the data F_1 associating with Role R_2.

In addition, a data owner assigns resources to a role R, if R's trust value is above a given threshold. The procedure of uploading an encrypted file F and assigning it to role R is presented in Algorithm 1. The procedure for trust calculation of role R is presented in Algorithm 2 which is designed on the basis of concept presented [11].

A data owner maps resources in a role via channel 7. When a data owner wants to assign a resource to a role, the data owner requests for trust value evaluation for the role to TVEE via channel 5. The TVEE accesses trust records of the role from Feedback section via channel 13, evaluates trust value and sends it to the data owner via channel 5. If the trust value is above a given threshold, the data owner

Algorithm 1 *File Uploading Procedure*

1: **procedure** UPLOAD(ENC-FILE F, ROLE R)
2: Set TR := TRUST-VALUE(ROLE R)
3: **if** TR > THRESHOLD **then**
4: Specify a set of operations that can be performed by users of ROLE R
5: Store the ENC-FILE F to ROLE R
6: **end if**
7: **end procedure**

Algorithm 2 *Trust value Calculation of a Role*

1: **procedure** TRUST-VALUE(ROLE R)
2: TVEE access Trust records of ROLE R and calculate TRUST VALUE of ROLE R
3: Return TRUST VALUE of ROLE R
4: **end procedure**

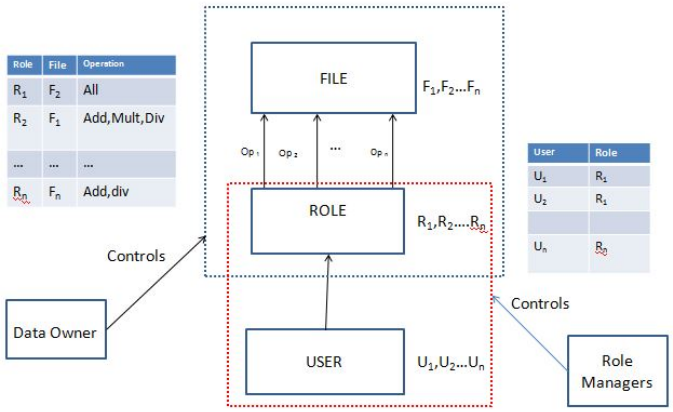

Figure 2: User-Role-File Mapping

uploads the resource to the role via channel 7 and specifies set of operations can be performed by users and accordingly provides information to RBA about the resource identity via channel 6. Then positive feedbacks of the associated roles is increased by one. This is done by RBA via channel 8.

Algorithm 3 *File leakage Handling Procedure*

1: **procedure** FILE-LEAKED(ROLE R,FILE F)
2: Data owner sends feedback to RBA for Role R
3: Role R sends identity of file F to UBA
4: UBA monitors user access history and updates trust records of users
5: **end procedure**

If a data owner finds a leakage in his or her resource, he or she gives feedback on the role specifying the mapping of the resource to RBA via channel 6 and RBA forwards this to Feedback section via channel 8. The associated role sends the resource identity to the UBA via channel 3, then the UBA accesses the History section via channel 12. Finally, UBA determines whether users are involved in file leakage and updates the trust record of users in Feedback section via channel 14 if they are involved. The procedure of file leakage handling is presented in Algorithm 3.

4.2 User-Role Mapping

The mapping of user to role [refer Figure 2] is controlled by the concerned roles. This is again a many to many association, i.e., a user can have membership in many roles and a role can have many users.

An user can accesses resources from the role or from descendant roles in role hierarchy. An user is allowed perform a set of operations on a resource, which are specified by the data owner of that resource.

The user can access the file, once the mapping between files, roles and users is rendered. Algorithm 4 describes the procedure for accessing files. Data owner respond procedure is used in file access algorithm is presented in Algorithm 5.

Algorithm 4 *File Access Procedure*

1: **procedure** FILE-ACCESS(USER U,FILE-ID F,OPERATION-SET S)
2: USER U sends request UBM to perform a set of operations
3: UBM sends request DO of F
4: **if** DO-RESPONDS(F,S,U)=TRUE **then**
5: USER U accesses the data from Private Section
6: **end if**
7: **end procedure**

Algorithm 5 *Data Owner Respond Procedure*

1: **procedure** DO-RESPONDS(FILE-ID F,OPERATION-SET S,USER U)
2: Set TR := TRUST-VALUE(USER U)
3: **if** TR > THRESHOLD **then**
4: DO does specific operations using homomorphic cryptosystem and stores in private section
5: Return TRUE
6: **else**
7: Return FALSE
8: **end if**
9: **end procedure**

When a user wants to perform some operations on resource, he or she sends requests to the UBM via channel 9. UBM passes the request to the data owner, if the user is authorized. The request is stored in private section and the data owner receives the request when he becomes online. Then the data owner perform these operations on encrypted file and uploads the result in private section of the cloud via channel 7. This uploaded resource can be accessed by the user who requested for the same. Then UBM sends the user and the resource identity information to UBA via channel 11. The UBA stores these information in History section via channel 12.

The next section describes the evaluation of our proposed RBAC system with experimental results.

5 EXPERIMENTAL RESULTS AND DISCUSSIONS

For the purpose of experimentation, we have considered six different roles R1..R6 with five different operations, i.e., addition, multiplication, division, searching and sorting on resources. Each role is mapped to 100 users. Finally, we evaluate trustworthiness of roles and users as these are subsequently required by data owners and roles respectively for allowing fine grained resource access to the user. We have implemented our proposed RBAC system in Java programming under Linux operating system platform.

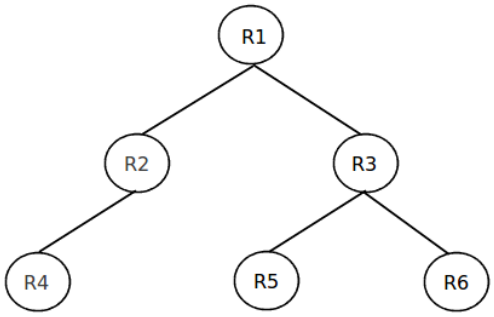

Figure 3: Role Hierarchy for Result

5.1 Trust Evaluation for Roles

This subsection describes about how trust value of different roles is affected by different contextual parameters. Trust value of a role depends on number of users in the roles and feedbacks given to different roles. We use the role hierarchy as shown Figure 3 to evaluate the trust values of the role and we assume that all the users have equal suspicion in leaking the data.

5.1.1 Impact on trust value of a role by other roles' feedbacks. First, we evaluate how trust value of a role varies when feedbacks are given to different roles. Figure 4 shows the variation of the trust value of Role R2 when feedbacks are given to different roles. Trust value of R2 is the minimum of its own trust and same of its ancestors; i.e., R1. If there is no feerback (individual and inheritance feeback[11]) for a role or for any of its ancestors, the role is not included in evaluation of effective trust value for that role. Here, we consider the weight of inheritance trust is 0.45 (w=0.45)[11].

It has been observed, when good feedback percentage(GFP) is 80%, the trust value for role R2 increases with increasing number of feedbacks. Increase in trust value of role R2 is fastest when feedbacks are given to only role R1. Because in such case, all the feedbacks are used in calculation of individual trust of role R1, and the combination trust of R1 is not affected by other roles. This is because of the fact that there is no feedback from other roles. The overall trust value of R2 remain same R1 as there is no feedback (individual and inheritance feedback). On the other hand, increase in

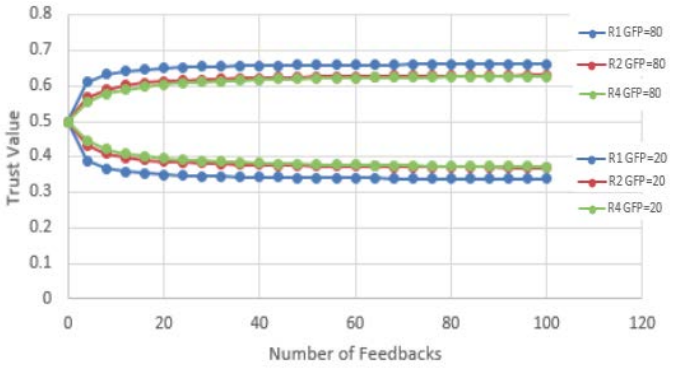

Figure 4: Trust value of role R2 considering feedbacks to different roles

trust value of role R2 is slowest when feedbacks are given to R4. Because in this case, inheritance trust value for R1 or R2 is calculated by taking average of feedbacks over three roles, R1 ,R2, and R4 . This means 1/3 of the feedbacks are considered in the calculation of inheritance trust value of R2 or R1. When feedbacks are given to R5 or R6, trust value of R2 remain same considering feedbacks are given to R4. This is because of the fact that inheritance trust of R1 is calculated by taking average of feedbacks from roles R1, R2 and R5 or R1, R2, R6. The increase in trust of R2 considering feedbacks to R2 is relatively faster than the same considering feedbacks to R4. This is due to inheritance trust calculation of R1 involves averaging out feedbacks of two roles R1 and R2.

5.1.2 Impact of number of users on trust value of roles. Here, we explain how trust value of a role varies as number of users changes in a role. We assume that R1, R2, R4 and R5 do not have any feedback and each of R3 and R6 have 120 feedbacks and number of users in each of R1 and R6 is 100. We set GFP of R3 and R6 to 25%.

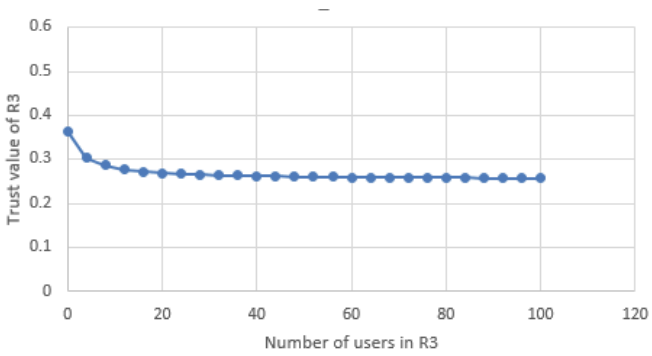

Figure 5: Trust value of R3 considering number of users changes in roles

The variation of trust value of R3 is shown in Figure 5 as the number of users changes in R3. It has been observed,

when good feedback percentage is 25%, trust value or R3 decreases as number of users in R3 increases. This is because of introduction of more suspicion in the users of R3 as number of users increasing.

5.2 Trust Evaluation for Users

In this subsection, we evaluate, how trust of roles in users is affected by different aspects. Trust of role to an user depends on the number of files (feedbacks) assigned to roles and on the role in which the user has membership.

Suppose users U_1, U_2, U_3, U_4, U_5, U_6 have membership in R1, R2, R3, R4, R5, R6 respectively. We assume that number of files (feedbacks) in each role is same and we set the weight of recommended trust to 0.45 (w=0.45). The trust for these users evaluated by R2 is shown Figure 6.

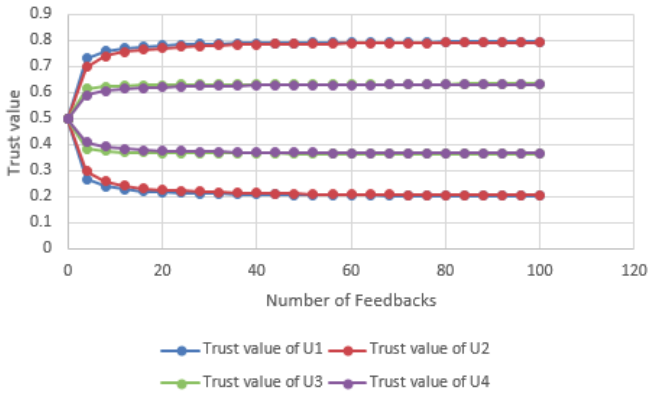

Figure 6: Trust value of users computed by role R2

When the GFP is 80%, the trust value of R2 in users increases as number of feedbacks increases and when the GFP is 20%, trust value decreases as number of feedbacks increases.

When GFP is 80%, increase in trust value of R2 for U_1 is fastest because U_1 has feedbacks in every role. So U_1 has direct trust in R2 and recommended trust in R1, R3, R4, R5, R6. Increase in trust value of R2 for U_4 is slowest because U_4 has only recommended trust in R4 and does not have any direct trust. Trust value of R2 for U_5 and U_6 is same as U_4 because they also have same recommended trust in R5 and R6 respectively and do not have any direct feedback. Increase in trust value of R2 in U_2 is slower than U_1 because U_2 has direct trust same as U_1 but has recommended trust only in R4 whereas U_1 has recommended trust in R1, R3, R4, R5, R6. Increase in trust value for U_3 evaluated by R2 has been seen faster than trust value of R2 in U_2 because U_3 has recommended feedbacks in three roles whereas U_2 has direct feedback in one role and recommended feedback in one role. This increase depends on weight given to recommended trust. If recommended feedback has more weightage, then increase in trust value of R2 in U_3 is faster than increase in trust value of R2 in U_2.

The above experimental evaluation shows our proposed RBAC framework in integration with homomorphic cryptosystem is capable of assigning fine grained access control to different users for performing computations on encrypted file stored in cloud. It has been demonstrated that the framework is also scalable for large number of users and roles in an Enterprise cloud environment.

6 CONCLUSION AND FUTURE WORKS

In this paper, we proposed a RBAC system integrated with Homomorphic cryptosystem for secure and controlled access of data in cloud. The proposed system is developed based on trust and role hierarchy with multi-granular operational access rights to heterogeneous stakeholders or roles. We have also presented how trust value of a role depends on numbers of users in a role and also feedbacks provided to different roles. Our proposed system can evaluate the impact of trust on a role for different users based on various aspects. In Future, we will develop spatio-temporal RBAC model integrated with homomorphic cryptosystems that take into account location and time before making access decision. The efficacy of our proposed system has been represented with experimental results. The proposed system can help in strengthen the security perimeter in cloud.

REFERENCES

[1] David Ferraiolo and Richard Kuhn. 1992. Role-Based Access Control. In *In 15th NIST-NCSC National Computer Security Conference*. 554–563.
[2] Craig Gentry. 2009. *A fully homomorphic encryption scheme*. Ph.D. Dissertation. Stanford University.
[3] Nick Howgrave-Graham. 2001. Approximate integer common divisors. In *Cryptography and Lattices*. Springer, 51–66.
[4] Seny Kamara and Mariana Raykova. 2013. Parallel homomorphic encryption. In *International Conference on Financial Cryptography and Data Security*. Microsoft Research, 213–225.
[5] Kristin Lauter, Michael Naehrig, and Vinod Vaikuntanathan. 2011. Can Homomorphic Encryption be Practical?. *Proceedings of the 3rd ACM workshop on Cloud computing security workshop* (2011), 113–124.
[6] Jian Liu, Lusheng Chen, and Sihem Mesnager. 2016. Partially homomorphic encryption schemes over finite fields. *IACR Cryptology ePrint Archive* 2016/430 (2016).
[7] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. 1978. On data banks and privacy homomorphisms. *Foundations of secure computation* 4, 11 (1978), 169–180.
[8] Ronald L Rivest, Adi Shamir, and Leonard Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21, 2 (1978), 120–126.
[9] Kamalakanta Sethi, Amartyal Majumdar, and Padmalochan Bera. [Manuscript accepted for publication]. A Novel implementation of parallel Homomorphic Encryption For Secure Data Storage in Cloud. *International conference on cyber security and protection of digital services* ([Manuscript accepted for publication]).
[10] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. 2010. Fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 24–43.
[11] L. Zhou, V. Varadharajan, and M. Hitchens. 2015. Trust Enhanced Cryptographic Role-Based Access Control for Secure Cloud Data Storage. *IEEE Transactions on Information Forensics and Security* 10, 11 (Nov 2015), 2381–2395. https://doi.org/10.1109/TIFS.2015.2455952