iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 **Issue:** VI **Month of publication:** June 2017

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

Volume 5 Issue VI, June 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Securing Data Using Role Based Access Control for Data Centric Approach

Shila Ravte [1], Prof. Priti Subramanium[2]
[1]M.E. Student, Dept. of Computer Science and Engineering, SSGBCOET, Bhusawal, Maharashtra, India
[2] Assistant Professor, Computer Science and Engineering, SSGBCOET, Bhusawal, Maharashtra, India

Abstract: ***Currently all security arrangements are based on perimeter security. However, Use of cloud computing breaks the organization borders. When organisational data stored on the Cloud, indirectly data reside outside the organizational boundaries. This may cause users to lost control over their data and raises security problems that slow down the efficiency of Cloud computing. Data-centric access control approach with role-based expressiveness in which security is concentrated on protecting user data regardless the Cloud service provider who is responsible to store and manage data. Identity-based and proxy re-encryption techniques are used to Secure the authorization model. Data is in encrypted form and authorization rules are cryptographically protected to stored user data against the service provider access or misused. The authorization model provides very high expressiveness for datas with role hierarchies and resource hierarchical support. The solution takes advantage of the logic formalism provided by Semantic Web technologies, which make possible advanced rule management like semantic conflict detection. Real-time implementation has been developed and a working prototypical deployment of the proposal has been integrated within Google services.***

Keywords: ***RBAC, Data centric, Access Control , Identity-Based Proxy Re-encryption, Data-Centric Security***

I. INTRODUCTION

Security is one of the main issue user consider while using Cloud Computing. Moving data to cloud usually user is dependent on Cloud service provider (CSP) for data protection. Whereas this management is based on legal or Service Level Agreements (SLA), in this situation CSP may potentially access the data or he can provide it to third parties. So user should trust the CSP to apply the access control protocol specified by the data owner for different users. The main problem arises in Inter-cloud scenarios where data exchange from one CSP to another CSP. Sometimes users may loss control on their data. Even the trust on the united CSPs is out of the control of the data owner. This situation leads to think about data security approaches again and to move to a data-centric approach so that we can ensure that data are self-protected whenever they reside. Encryption is the most commonly used method to provide security to data in the Cloud. Encrypting data avoids unauthorized accesses. However, it requires new issues related to access control management. A rule-based strategy would be suitable to provide expressiveness. But this big challenge for a data-centric approach as data has no computation capabilities by itself. It is not able to determine or compute any access control rule policy. This raises the problem of policy decision for a self-protected data package: who should examine the rules upon an access request? The first choice would be to have them calculated by the CSP, but it may bypass the rules. Another option would be to have rules determined by the data owner, but this entail that either data could not be shared or the owner should be online to make a decision for each access request. To overcome this issues, several proposals try to provide data-centric solutions based on Attribute-based Encryption (ABE). This is based on Attribute-based Access Control (ABAC), in which accesses are granted to users according to set of attributes. But there is no data-centric approach providing an RBAC approach for access control in which data is encrypted and self-protected. The proposal in this paper considers a first solution for a data-centric RBAC model, offering an alternative solution to the ABAC model. An RBAC approach would be closer to current access control strategies, resulting more natural to apply for access control effectiveness than ABE-based mechanisms. This paper presents a data-centric access control strategy to self-protect data that can work in utrusted CSPs and provides extended Role-Based Access Control. The proposed authorization solution provides a rule-based model by using the RBAC scheme, where roles are used to simplify the management of access to the resources. This approach can help to control and manage

II. LITERATURE REVIEW

Different strategies can be found in the literature to gain control over authorization in Cloud computing. In [13] authors propose to keep the authorization decisions taken by the data owner only. The access model is not explore to the Cloud but kept secure on the

data owner premises. However, in this strategy the CSP becomes a mere storage system and the data owner should be online to response access requests from users. Another strategy from [14] deals with this problem by using a plug approach in the CSP that allows data owners to prepare their own security modules. This permits to control the authorization model used within a CSP. However, it does not assure that how the authorization model should be secured, so the CSP could know information and access the data. Moreover, this strategy does not cover Inter-cloud scenarios, since the plug-in module should be delivered to different CSPs. Additionally, these strategies do not protect data with encryption methods. In the proposed data encryption is used to prevent the CSP to access the data or to release it bypassing the authorization strategies. However, applying data encryption entails additional challenges related to authorization expressiveness. Following a direct approach, one can include data in a package encrypted for the authorized users. This is done while sending a file or document to a specific receiver and assures that only the receiver with the appropriate key is able to decrypt it. From an authorization point of view, this can be seen as a simple rule where only the authorized user to access the data will be able to decrypt it (i.e. the one owning the key). However, no access control expressiveness is delivered by this approach. Only that simple rule can be emphasized and just one single rule can apply to each data package. Thus, multiple encrypted copies should be created in order to deploy the same data to different receivers. To cope with these issues, a data-centric approach that is able to cryptographically secure the data while providing access control capabilities. Number of data-centric approaches are developed, mostly based on Attribute-based Encryption (ABE) [5], have developed for data protection in the Cloud [4].

In ABE, encrypted cipher-text is labelled with a set of attributes defined by the data owner. Users also have a set of attributes with their private keys. They would be able to access data (i.e. decrypt it) or not depending on the match between cipher-text and key attributes. User required attribute to decrypt the data specified by an access structure, which is defined as a tree with AND, OR nodes. There are two main strategies for ABE depending on where the access structure resides: Key-Policy ABE (KP-ABE) [5] and Cipher-text-Policy ABE (CP-ABE) [3]. In Key Policy-ABE the access structure or policy is defined within the private keys of users. This allows encrypting data with attributes as labelled and then controlling the access to such data by delivering the appropriate keys to users. However, in this case the policy is defined by the key issuer not by encryptor of data, i.e. the data owner. So, the data owner should trust the key issue for this to properly generate an appropriate access policy. To overcome this issue, CP-ABE specifies including the access structure within the cipher-text, which is under control of the data owner. Then, the key issuer just affirm the attributes of users by including them in private keys.. Different proposals have been also developed to try to alleviate ABE expressiveness prescription. Authors in [15] propose a solution depend on CP-ABE allows sets of attributes called Cipher-text Policy Attribute Set Based Encryption (CP-ASBE). Attributes are arranged in a recursive set structure and access policies can be defined upon a single set or joining attributes from multiple sets. The definition of compound attributes and specification of policies that influence set of attributes. An strategy named Hierarchical Attribute-based Encryption is presented in [16]. It defines a hierarchical generation of keys to achieve subtle access control, scalability and delegation. However, this strategy implies that attributes should be managed by the same root domain authority. In [16], authors extend CP-ASBE with a hierarchical structure to users in order to improve scalability and flexibility. This approach provides a hierarchical solution for users within a domain, which is obtained by a hierarchical key structure. One more approach is Flexible and Efficient Access Control Scheme (FEACS) [2]. It is based on KP-ABE and provides an access control structure represented by a formula which include AND, OR and NOT, enabling more expressiveness for KP-ABE. ABE-based solutions developed for access control in Cloud computing are based on the Attribute-based Access Control (ABAC) model. ABAC and RBAC both models have their own advantages and disadvantages [7] [9]. Sometimes RBAC require the detailed specification of a large number of roles for authorization (role explosion problem in RBAC).

ABAC is also easier to set up without need to make an effort on role description as needed for RBAC. On another hand, ABAC may result in a huge number of rules since a system with n attributes would have up to 2^n possible rule combinations (rule explosion problem in ABAC). Moreover, the cryptographic operations used in ABE approaches usually repress the level of expressiveness provided by the access control rules. Concretely, role hierarchy and object hierarchy capabilities provided by Secure RBAC can-not be obtained by current ABE schemes. Moreover, a private key in ABE requires the attributes of the user, which tights the keys to permissions in the access control policy. In Secure RBAC, user keys only recognize their holders and they are not tied to the authorization model. That is, user access right are completely independent of their private key. Finally, no user-centric approach for authorization rules is proposed by current ABE solutions. In Secure RBAC, a single access policy specified by the data owner is able to secure more than one piece of data, resulting in a user-centric approach for rule management.

Volume 5 Issue VI, June 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

III.SYSTEM IMPLEMENTATION

A. *System Architecture*

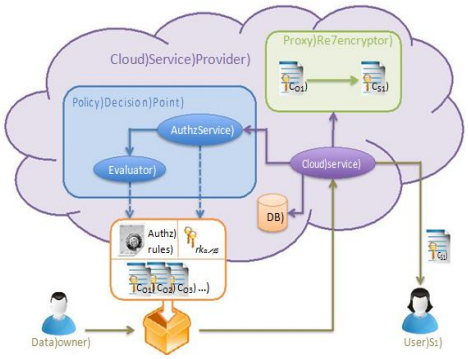


Fig:1 System Architecture

Above figure shows Working of proposed system in which the application of these functions make use of the re-encryption scheme to lose the Multi-use feature, which is needed as described in this paper. That is, once a Re-encryption Key generated is used to re-encrypt, no further re-encryptions allowed to that encrypted object. However, for the purposes of authentication in this paper, this kind of re-encryption only required to be done to re-encrypt the protected object And this is executed in the last re-encryption, which is the one that results in the data being encrypted using the user public key. Thus, re-encryption of keys generated with by original function should be applied for re-encryptions with proper authorization path, without the affecting the user, which is the last re-encryption. With this strategy, the data owner uses the public key while defining rules in the authorization model. When user send a request, the data object is re-encrypted by public key of user. So that user can then decrypt the data by using the appropriate private key.

When data is going to stored on the cloud, Data owner automatically create a self-protected package This package contains: the encrypted data objects, the authorization rules and the corresponding re-encryption keys. Data objects are encrypted at the time of uploading them to the Cloud in order to prevent the CSP to access them. This step is executed by data owners by using the encrypt() function. Data must be encrypted using the identity of the object which is going to uploaded . A digital envelope approach can be used to secure data objects without applying direct encryption. This would extends crypto-graphic operations like re-encryptions for large number of data objects. This strategy consists in using a symmetric encryption algorithm (e.g. DES) to protect the data object at its own. The encryption of data is done by using a random symmetric key generated for the single encryption. Then, this key is encrypted using the encrypt() function. By using this procedure, big objects (e.g. large documents) are encrypted using symmetric cryptography, by using more efficient algorithms. Authorization rules are specified by the data owner and directly mapped into the authorization model. This is done by considering the corresponding elements in the binary relations.

B. *Identity Based Proxy Re-encryption*

It combines both IBE and PRE, allowing a proxy to translate a cipher text encrypted under a user's identity into another cipher ext under another user's identity. In this approach, a Master Secret Key (MSK) is used to generate user secret keys from their identities. These secret keys are equivalent to private keys in IBE. No public keys are needed, since identities are directly used in the cryptographic operations. With this approach, a user can encrypt a piece of data m using his identity to obtain a cipher text encrypted under identity. A re-encryption key can be generated to re-encrypt . Then, a proxy can use to obtain another cipher text under the identity of another user. This can then use his own secret key to obtain the plain piece of data m. As for IBE approaches, the MSK should be kept private and users can obtain their secret key from the PKG. This IBPRE scheme is the one selected for the authorization solution proposed in this paper. It has been selected because it combines both PRE and IBE. It fulfils the three aforementioned requirements of proxy re-encryption and supports IBE, what allows to use the identities of the authorization elements for cryptographic operations, avoiding the need to generate and manage a key pair for each element. As mentioned before, the proposed solution is not tied to any PRE scheme or implementation. For the purpose of providing a comprehensive and feasible solution, the rest of this paper is based on the IBPRE approach and notation. However, the proposal could be applied to use other Proxy Re-Encryption schemes that fulfil the three aforementioned required features. This includes current or future schemes that

could improve performance or security. It could be even a pure PRE scheme without combination with IBE, although that could imply the generation and management of extra key pairs. Moreover, some functionality provided by this solution might be lost, like compatibility with PKI, which is supported by IBPRE and avoids the usage of a PKG.

The following set of functions is provided by IBPRE. It constitutes the cryptographic primitives for the proposal:

Initializes the cryptographic scheme. It takes as input a security parameter to initialize the cryptographic scheme (e.g. parameters to generate an elliptic curve) and outputs both the Master Secret Key and a set of public parameters that is used as input for the rest of functions.

Generates Secret Keys. It takes as input the msk and an identity outputs the Secret Key corresponding to that identity.

Encrypts data. It takes as input an identity and a plain text m; and outputs the encryption of m under the specified identity.

Generates Re-encryption Keys. It takes as input the source and target identities l as the Secret Key of the source identity and outputs the Re- encryption Key that enables to re-encrypt .

Re-encrypts data. It takes as input a cipher text under identity and a Re-encryption Key and outputs the re-encrypted cipher text under identity .

Decrypts data. It takes as input a cipher text and its corresponding Secret Key and outputs the plain text m .

C. RNS Algorithm

1) Key Generation Algorithm

Step 1: Start
Step 2: Generate two distinct prime number P1 & P2
Step 3: Let M = P1 * P2
Step 4: Let A1 = M / P1
Step 5: Let A2 = M / P2
Step 6: Solve the equation ((A1 * T) mod P1) == 1 and get the value of T
Step 7: T1 = T
Step 8: Solve the equation ((A2 * T) mod P2) == 1 and get the value of T
Step 9: T2 = T
Step 10: Form Encryption Key (P1,P2)
Step 11: Form Decryption Key (A1,A2,T1,T2,M)
Step 12: Stop

2) Encryption Process

Step 1: Start
Step 2: Let F is the File Input and (P1,P2) is encryption key
Step 3: Let N = Number of bytes in F
Step 4: Create a file EF to store encrypted data
Step 5: Let I = 1
Step 6: Read the Ith byte (B) of F
Step 7: Convert the B to ascii value V
Step 8: Let R1 = V % P1
Step 9: Let R2 = V % P2
Step 10: Append (R1,R2) in EF
Step 11: I = I + 1
Step 12: if I <= N Goto Step 6
Step 13: Close the File EF
Step 14: Stop

3) Decryption Process:

Step 1: Start
Step 2: Let EF is the Encrypted File Input and (A1,A2,T1,T2,M) is decryption key

Step 3: Let N = Number of Pair values in EF
Step 4: Create a file F1 to store decrypted data
Step 5: Let I = 1
Step 6: Read the Ith Pair value (R1,R2) from EF
Step 7: Let E = [(A1 * T1 * R1) + (A2 * T2 * R2)] mod M
Step 8: Convert E to Byte Value E1
Step 10: Append E1 in F1
Step 11: I = I + 1
Step 12: if I <= N Goto Step 6
Step 13: Close the File F1
Step 14: Stop

D. DES Algorithm

The main parts of the algorithm are as follows:

1) Fractioning of the text into 64-bit (8 octet) blocks;

2) Initial permutation of blocks;

3) Breakdown of the blocks into two parts: left and right, named L and R;

4) Permutation and substitution steps repeated 16 times (called **rounds**);

Re-joining of the left and right parts then inverse initial permutation

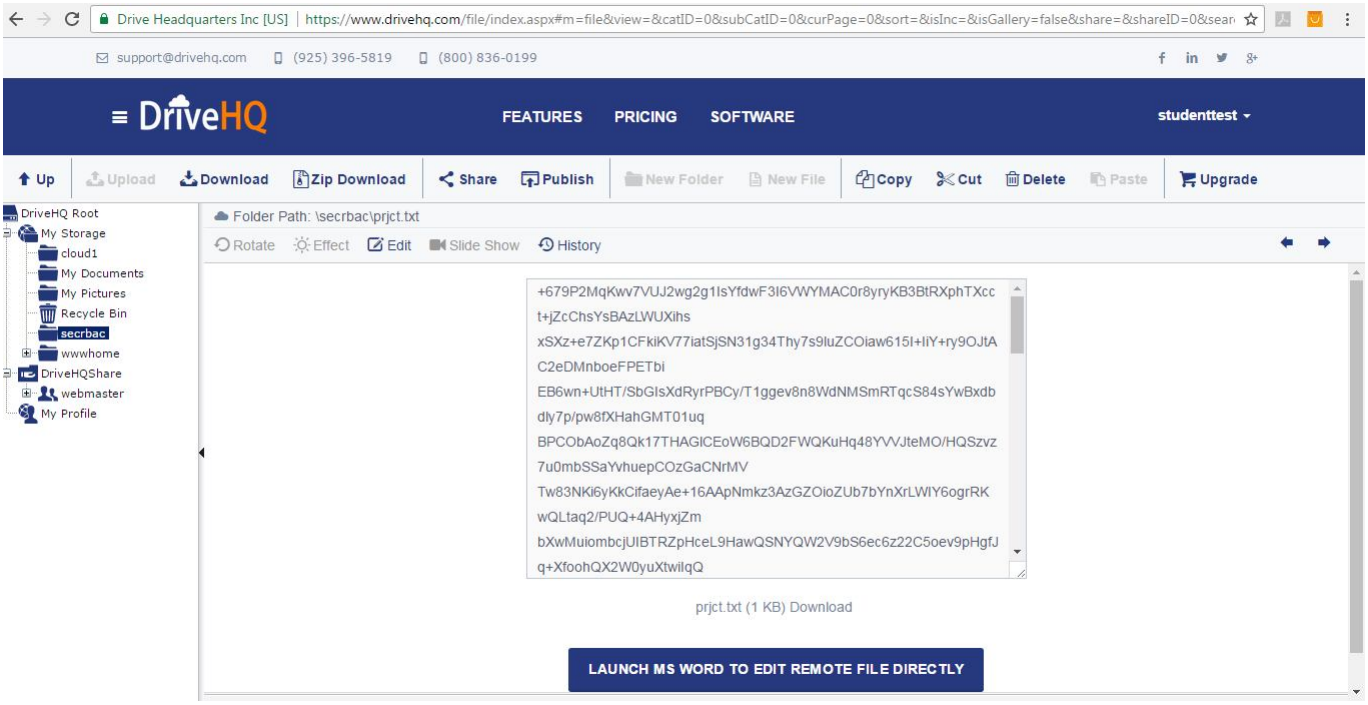


Fig 2: Encrypted file stored on cloud

IV. EXPERIMENTAL RESULTS

A prototypical implementation has been developed to demonstrate the feasibility of the proposal. It has been integrated in Drivehq Cloud services to provide security to documents in Drive. An implementation of the IBPRE scheme has been developed by using DES and RNS cryptography algorithms . The Java Pairing- Based Cryptography Library (JPBC) and the Bouncy Castle Crypto APIs have been used for the cryptographic operations. Documents uploaded by the data owner to Drive are encrypted by using digital envelopes. They are encrypted using DES and RNS. The implementation supports both PKG and PKI for key management,

corresponding to MD5 algorithm

An analysis has been carried out based on this implementation to test the feasibility of the proposal in terms of performance. Tests have been done with an Intel i3 CPU at 2.53 GHz and 3 GB of RAM. A first set of tests consisted on measuring execution times for the cryptographic functions. These have been done by varying different parameters in order to observe how these affect the execution times. Concretely, the following variations have been done: (1) number of re-encryptions, (2) length of identities and (3) length of encrypted data. Then, another set of tests have been done to measure the time needed to evaluate the authorization model. In order to obtain statistics significant results, operations have been performed in sets of 100 executions, whose average is used as result value. Each execution performs the following steps. First, the setup() function is executed to initialize the cryptographic scheme. Then a piece of data m is encrypted under a randomly generated identity id1 with the encrypt() function to obtain a cipher-text. The corresponding Secret Key sk1 is generated with the keygen() function. These three last steps are repeated several times, resulting in a cipher-text under identity id_n after n re-encryptions. Finally, the decrypt() function is used to decrypt and obtain the plain data m. The length of the plain data m, the length of identities and the number of re-encryptions may vary depending on the test.

Several tests have been done by changing one of these parameters to test the functions under different circumstances. When a parameter do not change in a test, default values are 512 bytes for data length, 32 bytes for identity lengths and 100 for the number of re-encryptions. In a PRE scheme, some operations could be affected by the number of re-encryptions, while others may be independent. A first test has been done by varying the number of re-encryptions from 1 to 100 by incrementing in 10 re-encryptions for each execution set. shows the results for this test. The encrypt() time is not shown because it is the same as the keygen() time and their lines are overlapped in the graphic. Times for setup() and decrypt() are shown in a separate graphic because they present higher values and showing them with the rest of functions would distort the Y axis scale. As can be observed, setup(), keygen(), encrypt(), rkgen() and reencrypt() remain constant. This is because these operations do not process the re-encrypted cipher-ext. On another hand, decrypt() increases with the number of re-encryptions. This is because re-encryptions are applied one over another in the cipher-text and decrypt() has to undo these re-encryptions. It is worth mentioning that the number of re-encryptions depends on the expressiveness used by the data owner when defining the authorization rules. Re-encryptions for an access request can be observed. At least one re-encryption should be done. This is the case when an access grant in the binary relation is directly granting the requesting user access to the requested object. If roles are used, then at least two re-encryptions should be done. The one for the access grant and another one for the subject role assignment . Then, if hierarchical expressiveness is used, several re-encryptions could be needed for the parent role and parent-object assignments in E and F, respectively. Thus, the number of re-encryptions would depend on the hierarchical levels that are defined between the role of the requesting user and the granted role plus the levels between the requested and the granted object. It should be noticed that this does not mean the number of roles or objects managed by the model, but only the levels in their hierarchies. The number of re-encryptions depends on the number of role and object levels between the subject and the object . The test has been done up to 100 re-encryptions in order to stress the system, considering 100 levels in role and object hierarchies. However, in practical terms a number of 10 levels (20 at most) would be enough for a realistic scenario. For this number of re-encryptions, decrypt() remains under acceptable execution times . In turn, in an IBE scheme, the length of the identities used for the cryptographic operations may also affect the execution times. Another test has been done by varying the length of the identities from 8 to 512 bytes by incrementing in 64 bytes for each execution set. the results for this test. Results do not show any variation for the cryptographic functions. Initially, it should affect functions dealing with identities, i.e. those taking id as parameter. These functions are keygen(), encrypt() and rkgen(). However, processing of these strings within the functions is so small that it is negligible for the execution time. Fig. 4. Times changing the length of identities Finally, it is also interesting to see how times could be affected by the length of the plain data being encrypted. Another test have been carried out by varying this parameter. As described earlier, the usage of digital envelope is encouraged as usual for asymmetric cryptography schemes. In this approach, data is encrypted using a symmetric algorithm such as DES with a random key that is then encrypted using the asymmetric scheme. Hence, four lengths have been considered for this test: 128, 192, 256 and 512 bits. The first three correspond to the common key lengths used for DES encryption. The last one has been also added to test stronger RNS encryption with 512 bytes keys as some proposals are currently arising test. Again, results show a constant execution time for all functions. Theoretically, encrypt(), reencrypt() and decrypt() could be affected by the length of the data since they take the plain data m or its encrypted counterpart as parameter. However, the mathematical operations within the cryptographic functions of IBPRE deal with a numeric representation that maps to numbers of the same length for all the considered lengths and performance is not affected. To sum up, the average execution times obtained during the tests are: 178 ms for setup(), 29 ms for keygen(), 29 ms for

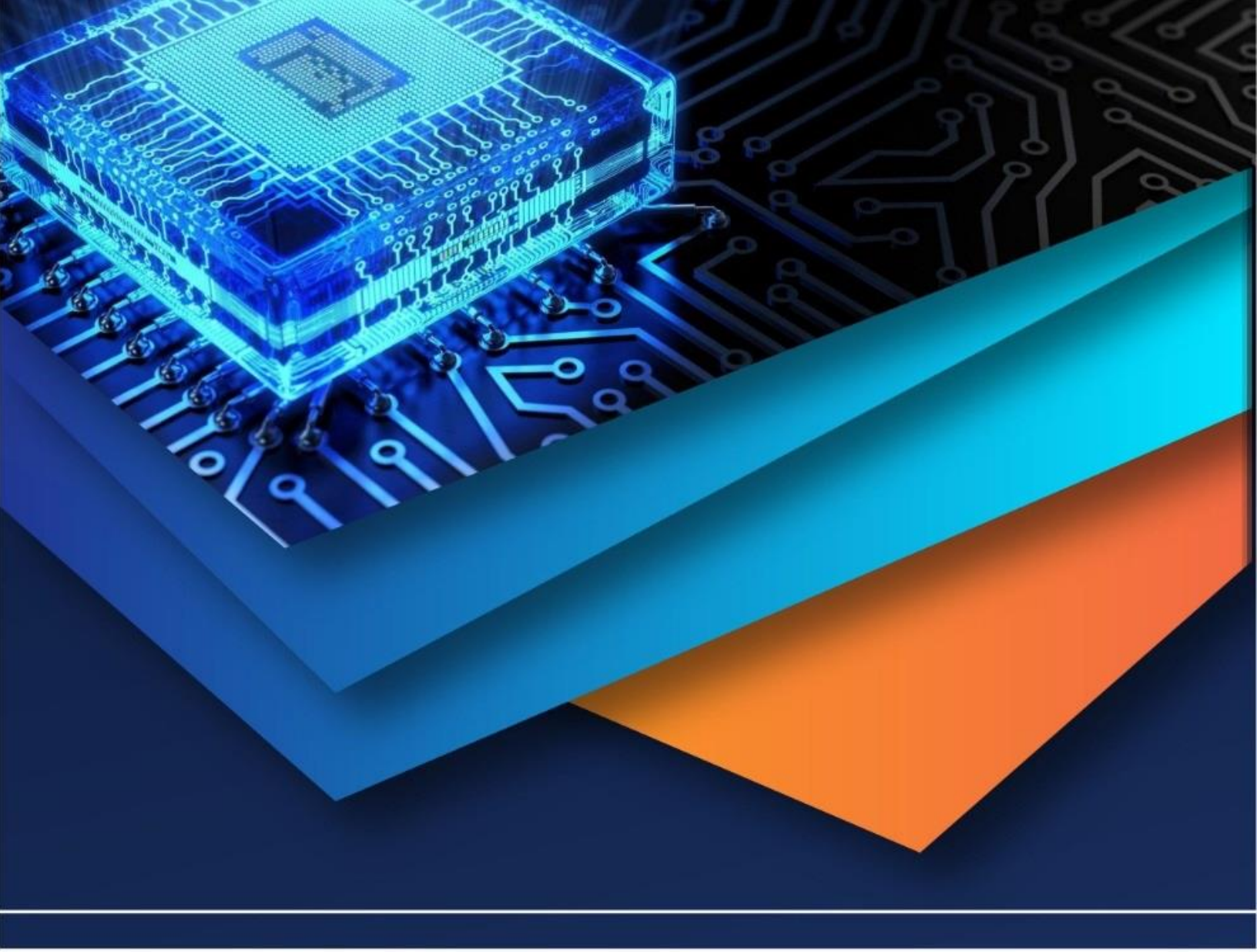
encrypt(), 48 ms for rkgen(), 6 ms for reencrypt() and 247 ms for decrypt() with 10 re-encryptions. Another set of tests has been also performed to measure the time needed by the AuthzService to take authorization decisions. The authorization model has been represented based on Semantic Web technologies . Times changing the length of encrypted data. Decision times are related to the number of authorization elements defined within the model, i.e. number of roles, users, grants, objects, etc. These elements are represented in what is called a Knowledge Base (KB), which is used by the reasoner implementing the Evaluator to evaluate the authorization query. The elements contained in the KB are called individuals and the set of individuals for a specific execution is referred to as population. In order to study the scalability of the proposal, 6 executions have been done using incremental populations. Access denied decisions would result in shorter times since the authorization chain does not need to be computed..Authorization decision times As can be observed, the query times are incremented as the population grows, while times for building the authorization chain and retrieving re-encryption keys remain con This is because the query need to analyze the ontology in order to check the conditions. The more individuals in the ontology, the more data should be analyzed by the reasoner during the query evaluation. However, computing the authorization chain from a given answer is independent of the ontology. So is the retrieval of re-encryption keys. Thus, times remain constant for these two, independently of the number of individuals. Finally, it should be noted that times for authorization decisions remain below 13 ms for populations of more than 6000 authorization elements, which is a reasonable decision time.

V. CONCLUSIONS

A data-centric authorization solution has been introduced to provide security to data stored on Cloud. Secure RBAC allow to manage authorization which follow rule-based approach and provides high -leveled role-based expressiveness including object and role hierarchies. Access control computations are delegated to the CSP, being this not only unable to data the access, but also not able to release it to unauthorized parties. Advanced cryptographic strategies have been applied to provide security to the authorization model. A re-encryption key complement each authorization rule as cryptographic token to provide security to data against CSP misbehavior. A concrete IBPRE technique has been used in order to provide a comprehensive and feasible option with Public Key Cryptography that enables the usage of standard PKI for key distribution and management. Future lines of research include the detailed study of novel cryptographic techniques that could enable the modification and deletion of data on the Cloud. This would allow to expand the privileges of the authorization model with more actions like delete and modify. One more interesting point is the obscure of the authorization model for privacy reasons. Although the usage of pseudonyms is introduces, but more advanced obfuscation techniques can be researched to achieve a higher level of security.

REFERENCES

[1] Cloud Security Alliance, “Security guidance for critical areas of focus in cloud computing v3.0,” CSA, Tech. Rep., 2003.
[2] Y. Zhang, J. Chen, R. Du, L. Deng, Y. Xiang, and Q. Zhou, “Feacs: A flexible and efficient access control scheme for cloud computing,” in Trust, Security and Privacy in Computing and Communications, 2014 IEEE 13th International Conference on, Sept 2014, pp. 310–319.
[3] B. Waters, “Cipher text-policy attribute-based encryption: An ex-pressive, efficient, and provably secure realization,” in Public Key Cryptography - PKC 2011, 2011, vol. 6571, pp. 53–70.
[4] B. B and V. P, “Extensive survey on usage of attribute based encryption in cloud,” Journal of Emerging Technologies in Web Intelligence, vol. 6, no. 3, 2014.
[5] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in Proceedings of the 13th ACM Conference on Computer and Communi-cations Security, ser. CCS ’06, New York, NY, USA, 2006, pp. 89–98.
[6] Inter National Committee for Information Technology Standards, “INCITS 494-2012 - information technology - role based access control - policy enhanced,” INCITS, Standard, Jul. 2012.
[7] E. Coyne and T. R. Weil, “Abac and rbac: Scalable, flexible, and auditable access management,” IT Professional, vol. 15, no. 3, pp. 14–16, 2013.
[8] Empower ID, “Best practices in enterprise authorization: The RBAC/ABAC hybrid approach,” Empower ID, White paper, 2013.
[9] D. R. Kuhn, E. J. Coyne, and T. R. Weil, “Adding attributes to role-based access control,” Computer, vol. 43, no. 6, pp. 79–81, 2010.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY