ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 6, June 2017

A Data Centric Access Control Solution with Role Based Expressiveness for Protecting User Data on Cloud

[1] Nisha Ranjan Sah, [2] Tejashwini P, [3] Veeresh S, [4] Yathish, [5] Mrs. Prema.
[1][2][3][4] Students of Department of Computer science and Engineering , RRIT.
[5] Asst Prof , Dept of Computer Science and Engineering ,RRIT

***Abstract*— Most current security solutions are based on perimeter security. However, Cloud computing breaks the organization perimeters. When data resides in the Cloud, they reside outside the organizational bounds. This leads users to a loss of control over their data and raises reasonable security concerns that slow down the adoption of Cloud computing. Is the Cloud service provider accessing the data? Is it legitimately applying the access control policy defined by the user? This paper presents a data-centric access control solution with enriched role-based expressiveness in which security is focused on protecting user data regardless the Cloud service provider that holds it. Novel identity-based and proxy re-encryption techniques are used to protect the authorization model. Data is encrypted and authorization rules are cryptographically protected to preserve user data against the service provider access or misbehavior. The authorization model provides high expressiveness with role hierarchy and resource hierarchy support. The solution takes advantage of the logic formalism provided by Semantic Web technologies, which enables advanced rule management like semantic conflict detection. A proof of concept implementation has been developed and a working prototypical deployment of the proposal has been integrated within Google services**

***Keywords:* Data-centric security, Cloud computing, Role-based access control, Authorization.**

I. INTRODUCTION

SECURITY is one of the main user concerns for the adoption of Cloud computing. Moving data to the Cloud usually implies relying on the Cloud Service Provider (CSP) for data protection. Although this is usually managed based on legal or Service Level Agreements (SLA), the CSP could potentially access the data or even provide it to third parties. Moreover, one should trust the CSP to legitimately apply the access control rules defined by the data owner for other users. The problem becomes even more complex in Inter cloud scenarios where data may flow from one CSP to another. Users may loss control on their data. Even the trust on the federated CSPs is outside the control of the data owner. This situation leads to rethink about data security approaches and to move to a data-centric approach where data are self-protected whenever they reside.

This paper presents *SecRBAC*, a data-centric access control solution for self-protected data that can run in un trusted CSPs and provides extended Role-Based Access Control expressiveness. The proposed authorization solution provides a rule-based approach following the RBAC scheme, where roles are used to ease the management of access to the resources. This approach can help to control and manage security and to deal with the complexity of managing access control in Cloud computing. Role and resource hierarchies are supported by the authorization model, providing more expressiveness to the rules by enabling the definition of simple but powerful rules that apply to several users and resources thanks to privilege propagation through roles and hierarchies. Policy rule specifications are based on Semantic Web technologies that enable enriched rule definitions and advanced policy management features like conflict detection. A data-centric approach is used for data self-protection, where novel cryptographic techniques such as Proxy Re-Encryption (PRE) [10], Identity Based Encryption (IBE) [11] and Identity-Based Proxy Re Encryption (IBPRE) [12] are used. techniques are used to protect both the data and the authorization model. Each piece of data is ciphered with its own encryption key linked to the authorization model and rules are cryptographically protected to preserve data against the service provider access or misbehavior when evaluating the rules.

II. RELATED WORK

Different approaches can be found in the literature to retain control over authorization in Cloud computing. In [13] authors propose to keep the authorization decisions taken by the data owner. The access model is not published to the Cloud but kept secure on the data owner premises. However, in this approach the CSP becomes a mere storage system and the data owner should be online to process access requests from users. Another approach from [14] deals with this issue by enabling a plug-in mechanism in the CSP that allows data owners to deploy their own security modules. This permits to control the authorization mechanisms used within a CSP. However, it does not establish how the authorization model should be protected, so the CSP could potentially infer information and access the data. Moreover, this approach does not cover Inter-cloud scenarios, since the plug-in module should be deployed to different CSPs. Additionally, these approaches do not protect data with encryption methods. In the proposed SecRBAC solution, data encryption is used to prevent the CSP to access the data or to release it bypassing the authorization mechanism.

From an authorization point of view, this can be seen as a simple rule where only the user with privilege to access the data will be able to decrypt it (i.e. the one owning the key). However, no access control expressiveness is provided by this approach. Only that simple rule can be enforced and just one single rule can apply to each data package. Thus, multiple encrypted copies should be created in order to deliver the same data to different receivers. To cope with these issues, SecRBAC follows a data-centric approach that is able to cryptographically protect the data while providing access control capabilities.

There are two main approaches for ABE depending on where the access structure resides: Key-Policy ABE (KP-ABE) [5] and Cipher text-Policy ABE (CP-ABE) [3]. In KP-ABE the access structure or policy is defined within the private keys of users. This allows to encrypt data labeled with attributes and then control the access to such data by delivering the appropriate keys to users. However, in this case the policy is really defined by the key issuer instead of the encryption of data, i.e. the data owner. So, the data owner should trust the key issues for this to properly generate an adequate access policy. To solve this issue, CP-ABE proposes to include the access structure within the cipher text, which is under control of the data owner. Then, the key issuer just asserts the attributes of users by including them in private keys. However, either in KP-ABE or CP-ABE, the expressiveness of the access control policy is limited to combinations of AND or OR-ed attributes. The data-centric solution presented in this paper goes a step forward in terms of expressiveness, providing a rule-based approach following the RBAC scheme that is not tied to the limitations of current ABE approaches.

Different proposals have been also developed to try to alleviate ABE expressiveness limitations. Authors in [15] propose a solution based on CP-ABE with support for sets of attributes called Cipher text Policy Attribute Set Based Encryption (CP-ASBE). Attributes are organized in a recursive set structure and access policies can be defined upon a single set or combining attributes from multiple sets. This enables the definition of compound attributes and specification of policies that affect the attributes of a set. An approach named Hierarchical Attribute-based Encryption is presented in [16]. It uses a hierarchical generation of keys to achieve fine-grain access control, scalability and delegation. However, this approach implies that attributes should be managed by the same root domain authority. In [17], 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 achieved by a hierarchical key structure. Another 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 involving AND, OR and NOT, enabling more expressiveness for KP-ABE.

The aforementioned ABE-based solutions proposed for solving access control in Cloud computing are based on the Attribute-based Access Control (ABAC) model. As commented in Section 1, both ABAC and RBAC models have their own advantages and disadvantages [7] [9]. On one hand, RBAC may require the definition of a large number of roles for fine-grain authorization (role explosion problem in RBAC). ABAC is also easier to set up without need to make an effort on role analysis as needed for RBAC. On another hand, ABAC may result in a large number of rules since a system with n attributes would have up to 2^n possible rule combinations (rule explosion problem in ABAC). ABAC separates authorization rules from user attributes, making it difficult to determine permissions available to a particular user, while RBAC is deterministic and user privileges can be easily determined by the data owner.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 6, June 2017

Moreover, the cryptographic operations used in ABE approaches usually restrict the level of expressiveness provided by the access control rules. Concretely, role hierarchy and object hierarchy capabilities provided by SecRBAC cannot be achieved by current ABE schemes. Moreover, private keys in ABE should contain the attributes of the user, which tights the keys to permissions in the access control policy. In SecRBAC, user keys only identify their holders and they are not tied to the authorization model. That is, user privileges are completely independent of their private key. Finally, no user-centric approach for authorization rules is provided by current ABE solutions. In SecRBAC, a single access policy defined by the data owner is able to protect more than one piece of data, resulting in a user-centric approach for rule management. Additionally, the proposed solution provides support for the ontological representation of the authorization model, providing additional reasoning mechanisms to cope with issues such as detection of conflicts between different authorization rules.

III. PROXY RE-ENCRYPTION AND IDENTITY-BASED ENCRYPTION

In an Identity-Based Proxy Re-Encryption (IBPRE) approach is proposed. It combines both IBE and PRE, allowing a proxy to translate a cipher text encrypted under a user's identity into another cipher text 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 u_α can encrypt a piece of data m using his identity id_α to obtain a cipher text$c_{id\alpha}$ encrypted under id_α. A re-encryption key $rk_{\alpha\rightarrow\beta}$ can be generated to re-encrypt from id_α to id_β. Then, a proxy can use $rk_{\alpha\rightarrow\beta}$ to obtain another cipher text$c_{id\beta}$ under the identity of another user u_β. This can then use his own secret key sk_β 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 fulfills 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 fulfill 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:

$$\mathbf{setup(p,k) \rightarrow (p,msk)} \quad (1)$$

$$\mathbf{keygen(p,msk,id_\alpha) \rightarrow sk_\alpha} \quad (2)$$

$$\mathbf{encrypt(p,id_\alpha,m) \rightarrow c_\alpha} \quad (3)$$

$$\mathbf{rkgen(p,sk_\alpha,id_\alpha,id_\beta) \rightarrow rk_{\alpha\rightarrow\beta}} \quad (4)$$

$$\mathbf{reencrypt(p,rk_{\alpha\rightarrow\beta},c_\alpha) \rightarrow c_\beta} \quad (5)$$

$$\mathbf{decrypt(p,sk_\alpha,c_\alpha) \rightarrow m} \quad (6)$$

IV. AUTHORIZATION MODEL WITH ENRICHED ROLEBASED EXPRESSIVENESS

The management of access control and security could become a difficult and error prone task in distributed systems like Cloud computing. Authorization models providing high expressiveness can help to control and manage security and to deal with this complexity. They can aid administrators with this task by enabling the specification of high-level access control rules that are automatically interpreted by system for this to behave as defined by the administrator. Role-Based Access Control (RBAC) is an authorization scheme supported by most of the current authorization solutions. In this approach, the authorization model makes use of the *Role* concept to assign privileges to subjects. A set of subjects can be assigned to one or more roles which, in turn, can be

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 6, June 2017

associated to a set of privileges. This provides more expressiveness to the authorization model, making it easier to manage privilege assignments through roles.

V. SELF-PROTECTED AUTHORIZATION MODEL FOR DATA-CENTRIC SECURITY

The authorization model presented in Section 4 determines the privileges that are granted to subjects. It should be evaluated by the Cloud Service Provider upon an access request in order to decide whether such a request is permitted or not. However, if data is not cryptographically protected then the CSP could potentially access the data for its own benefit. Moreover, the data owner should trust the CSP to legitimately evaluate the model and enforce the authorization decision. If the authorization rules are not cryptographically protected then they can be overridden by the CSP, making it able to access the data or to release it to any third party. A self-protected authorization model is needed to achieve a data-centric mechanism that technically guarantees the CSP cannot access or disclose data to unauthorized parties.

This section describes a protected authorization model for a data-centric solution. A self-protection mechanism is provided to assure data can only be accessed by authorized subjects according to the data owner rules. It is achieved by the application of the cryptographic techniques described in Section 3. Then, a representation and evaluation mechanism based on Semantic Web technologies is also proposed.

VI. DATA-CENTRIC SOLUTION FOR DATA PROTECTION IN THE CLOUD

In the protected authorization model specified in Section 5.1, it should be observed that data is not encrypted with the data owner identity, but with the object's own identity (e.g. id_{o1}). This follows a data-centric approach for data protection, in which data is encrypted with its own key under the cryptographic scheme. If a pure PRE scheme is used, the object would be also encrypted using its own key pair. On another hand, a user-centric approach is used for the authorization rules, where a unified access control policy is defined by the data owner for its data. This allows to share common definitions and to greatly simplify access control management, getting the most from role hierarchy and resource hierarchy capabilities.

An architecture is also proposed for the deployment within a CSPs. This architecture takes into consideration the different elements that should be deployed in order to give an overview of how access to protected data is done in this approach. Fig. 2 depicts the proposed architecture.

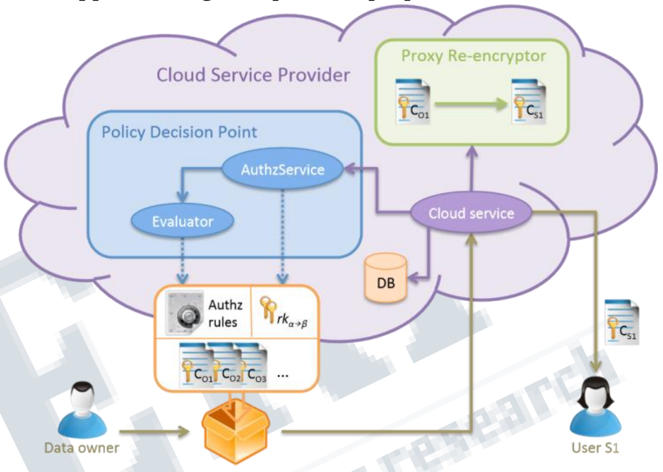

Fig. 2: Architecture for deployment in a CSP

Data objects are encrypted before uploading them to the Cloud in order to prevent the CSP to access them. This is done by data owners by using the *encrypt*() function (3). According to Def. 8, data should be encrypted using the identity id_{o1} of the object being uploaded o_1. A digital envelope approach can be applied to protect data objects instead of direct encryption. This would enhance cryptographic operations like re-encryptions for large data objects. This approach consists in using a symmetric encryption algorithm (e.g. AES) to protect the data object itself. The encryption of data is done with a random symmetric key generated for the purpose of a single encryption. Then, this key is encrypted with the *encrypt*() function. With this procedure, potentially big objects (e.g. large documents) are encrypted using symmetric cryptography, whose algorithms are more efficient. In turn, more costly operations are only applied to the keys used for the symmetric encryption, which are usually small pieces of data of some bytes length. They make use of a database to store the protected packages uploaded by data owners.

IFERP
connecting engineers... developing research

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 6, June 2017

Thus, it contains the information of these packages, i.e. encrypted data objects, authorization rules and re-encryption keys. It can also contain the parameters p to initialize the cryptographic functions. The information can be kept in data packages as provided by the data owner or it could be stored on any other format that facilitates data processing to the CSP.

An authorization service (*AuthzService*) acts as entry point to the PDP for Cloud services allowing to query it for authorization decisions. This module takes decisions upon a request from a user s_1 to access to a piece of data o_1 managed by the service. These decisions usually return an access granted or denied statement. For granted accesses, the response also contains the re-encryption chain that should be applied, together with the needed re-encryption keys. This information allows to re-encrypt from c_{o1} as provided by the data owner to c_{s1}, which can be decrypted by the requesting user. The service passes this information together with c_{o1} to the *Proxy Re-encryptor* for this to perform the re-encryption operations. It results in c_{s1}, which is sent to the requesting user. Making use of its own secret key sk_{s1} the user can decrypt the data with the *decrypt*() function (6). Note that during this process, the CSP is not able to access the data since it only applies a set of *reencrypt*() functions which do not disclose any information about the data being processed.

6.1 Key management and PKI compatibility.

IBPRE does not use public and private key pairs in cryptographic operations. Instead, a Master Secret Key (MSK) is used in combination with identities. This MSK is generated during the setup phase and it should be kept private. On another hand, users accessing the data need their own Secret Key (SK) to compute the *decrypt*() function. Secret keys are generated based on the user identity and the MSK. There are several approaches for the distribution of these keys to users. In a straightforward approach, SKs can be generated internally by the data owner to keep the MSK protected. However, this will lead to the need of distributing SKs securely to each user.

To this end, IBE schemes -including IBPRE- define a Private Key Generator (PKG) for the generation and distribution of SKs. This entity should be trusted by the data owner because it holds the MSK to generate the SKs. It can be deployed as a service by the data owner in its own premises. This would allow to keep the MSK under control, although it would result in a critical service that should be protected. Another choice would be a third

As an alternative, an hybrid proxy re-encryption approach can be applied. This concept was introduced in [23] and it consists in creating a bridge between IBE and Public Key based Encryption (PKE). The IBPRE scheme used in this proposal supports this feature. Thus, it can be used to manage user keys by using well known and standard technologies like a Public Key Infrastructure (PKI). This feature implies the inclusion of two new functions:

$$rkgen\_pke\,(p, sk_\alpha, id_\alpha, pub_\beta\ ) \rightarrow rk_{\alpha\rightarrow\beta} \tag{32}$$
$$decrypt\_pke\,(p, priv_\alpha, c_\alpha) \rightarrow m \tag{33}$$

The functions are similar to the original ones (4) and (6), but including some modifications. Details about the modifications that need to be done to these functions can be found in [12]. These functions take public and private keys pub_β and $priv_\alpha$ to apply PKE instead of identity id_β and Secret Key sk_α used for IBE.

The application of these functions makes the re-encryption scheme to lose the *Multi-use* feature, which is required That is, once a Re-encryption Key generated by *rkgen pke*() is used to re-encrypt, no further re-encryptions can be done to that encrypted object. However, for the purposes of authorization in this paper, this kind of re-encryption only needs to be done to re-encrypt the protected object under the requesting user public key. And this is done in the last reencryption, which is the one that results in the data being encrypted under the user public key. Thus, re-encryption keys generated with the original *rkgen*() function should still be applied for re-encryptions along the authorization path, except the one affecting the user, which is the last re encryption.

In practical terms, using the hybrid approach only implies that re-encryption keys affecting subjects $s_i \in S$ should be generated with the *rkgen pke*() function. That is, when the data owner defines a rule to grant a privilege or assigns a role to a given subject, the corresponding re-encryption key should be generated with (32). Otherwise, re-encryption keys should be generated with the original *rkgen*() function for the rest of the authorization elements.

It is worth mentioning that the two approaches can be combined. Some users can use PKE while others can still use IBE. The only thing that needs to be done by the data owner is to use the proper function (*rkgen pke*() or *rkgen*()) when generating the corresponding re-encryption keys.

6.2 Security considerations.

SecRBAC provides a self-protected mechanism to upload data to the Cloud assuring that no unauthorized party is able to access the data, including the CSP. In this case, the CSP is considered a curious adversary that would be willing to a) try to disclose the information to use it on its own benefit and b) try to neglect the authorization rules in order to release the information to an unauthorized third party. However, it is assumed that the CSP would still behave honestly according to the agreed service by releasing the data to the requesting users if they are authorized. That is, the CSP could intentionally provide corrupted ciphertexts, making users unable to access the data.

However, this would result in a bad service perceived by the users, making them to avoid using that CSP. It should be noticed that SecRBAC does not hamper the ability to provide data to the CSP if the data owner wants to do so. In this context, the CSP is considered as any other user. It could access to some pieces of data (e.g. to provide some service) only if the data owner has defined the corresponding rules in the authorization model. The solution allows the release of information, but enabling the data owner to keep control over its data.

VII. IMPLEMENTATION AND PERFORMANCE

A prototypical implementation has been developed to demonstrate the feasibility of the proposal. It has been integrated in Google Cloud services to provide security to documents in Google Drive. Since the core of Google Services cannot be modified, integration has been done by developing a Web application running on Google App Engine. This application is registered as a Google Drive application that integrates in Google Drive user interface.

The Web application contains the modules depicted in Fig. 2. The authorization model has been represented as described in Section 5.2 using Semantic Web technologies. So, the *Evaluator* in this implementation consists on an ontology reasoner. The Apache Jena framework has been used to manage the ontology and to perform reasoning. The *AuthzService* also makes use of this library to process the output of the reasoner and retrieve the authorization chain.

An implementation of the IBPRE scheme has been developed using elliptic curve cryptography. The Java PairingBased Cryptography Library (JPBC) and the Bouncy Castle Crypto APIs. The implementation supports both PKG and PKI for key management, corresponding to the two approaches described in Section 6.1. It also allows the data owner to directly generate and store user keys in case he wants to distribute them by other means. When generating a re encryption key related to a privilege or role assignment, the application asks the data owner to choose between an IBE or PKE key for that user. In case of IBE, the key will be generated automatically based on the user identity, otherwise it will ask for the public key of the user.

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 i5 CPU at 2.7 GHz and 6 GB of RAM. A first set of tests consisted on measuring execution times for the cryptographic functions exposed in Section 3. 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 by using the on topology based approach described

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 (1) is executed to initialize the cryptographic scheme. Then a piece of data m is encrypted under a randomly generated identity id_1 with the *encrypt*() function (3) to obtain a cipher text c_1. The corresponding Secret Key sk_1 is generated with the *keygen* () function (2). Then, another random identity id_2 is generated and a re-encryption key $rk_{1\rightarrow 2}$ is generated with the *rkgen* () function (4). This is used to re-encrypt the c_1 with the *reencrypt* () function (5). These three last steps are repeated several times, resulting in a cipher text c_n under identity id_n after n re-encryptions. Finally, the *decrypt*() function (6) is used to decrypt c_n and obtain the plain data m. The length of the plain data m, the length of identities id_i 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 reencryptions for each execution set. Fig. 3 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 ciphertext. The first four functions do not have c_α as parameter, so they are agnostic to the number of re-encryptions done to the ciphertext. In turn, *reencrypt*() takes this parameter, but operations within this function only process the last encrypted data, independently of the number of re-encryptions previously done to c_α. On another hand, *decrypt*() increases with the number of re-encryptions. This is because re-encryptions are applied one over another in the ciphertext 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 in (21). At least one re-encryption should be done. This is the case when an access grant in the binary relation G_a 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 in *D*. 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. As can be observed in (21), the number of re-encryptions depends on the number of role and object levels between the subject s_1 and the object o_1. The test has been done up to 100 re-encryptions in order to stress the system, considering 100 levels in role and object hierarchies from s_1 to o_1. 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 as shown in Fig. 3.

VIII. CONCLUSION

A data-centric authorization solution has been proposed for the secure protection of data in the Cloud. SecRBAC allows managing authorization following a rule-based approach and provides enriched role-based expressiveness including role and object hierarchies. Access control computations are delegated to the CSP, being this not only unable to access the data, but also unable to release it to unauthorized parties. Advanced cryptographic techniques have been applied to protect the authorization model. A re-encryption key complement each authorization rule as cryptographic token to protect data against CSP misbehavior.

Future lines of research include the analysis of novel cryptographic techniques that could enable the secure modification and deletion of data in the Cloud. This would allow to extend the privileges of the authorization model with more actions like *modify* and *delete*. Another interesting point is the obfuscation of the authorization model for privacy reasons. Although the usage of pseudonyms is proposed, but more advanced obfuscation techniques can be researched to achieve a higher level of privacy.

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, "Ciphertext-policy attribute-based encryption: An expressive, 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 Communications Security*, ser. CCS '06, New York, NY, USA, 2006, pp. 89–98.

[6] InterNational 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 rolebased access control," *Computer*, vol. 43, no. 6, pp. 79–81, 2010.

[10] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved Proxy Re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security*, vol. 9, no. 1, pp. 1–30, 2006.

[11] F. Wang, Z. Liu, and C. Wang, "Full secure identity-based encryption scheme with short public key size over lattices in the standard model," *Intl. Journal of Computer Mathematics*, pp. 1–10, 2015.

[12] M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *Proceedings of the 5th International Conference on Applied Cryptography and Network Security*, ser. ACNS '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 288–306.

[13] A. Lawall, D. Reichelt, and T. Schaller, "Resource management and authorization for cloud services," in *Proceedings of the 7th International Conference on Subject-Oriented Business Process Management*, ser. S-BPM ONE '15, New York, NY, USA, 2015, pp. 18:1–18:8.

[14] D. Y. Chang, M. Benantar, J. Y.-c. Chang, and V. Venkataramappa, "Authentication and authorization methods for cloud computing platform security," Jan. 1 2015, uS Patent 20,150,007,274.

[15] R. Bobba, H. Khurana, and M. Prabhakaran, "Attribute-sets: A practically motivated enhancement to attribute-based encryption," in *Computer Security - ESORICS 2009*. Springer Berlin Heidelberg, 2009, vol. 5789, pp. 587–604.

[16] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, ser. CCS '10, New York, NY, USA, 2010, pp. 735–737.

[17] J. Liu, Z. Wan, and M. Gu, "Hierarchical attribute-set based encryption for scalable, flexible and fine-grained access control in cloud computing," in *Information Security Practice and Experience*. Springer Berlin Heidelberg, 2011, vol. 6672, pp. 98–107.

[18] W3C OWL Working Group, "OWL 2 Web Ontology Language: Document overview (second edition)," World Wide Web Consortium (W3C), W3C Recommendation, Dec. 2012.

[19] J. M. A. Calero, J. M. M. Perez, J. B. Bernabe, F. J. G. Clemente, G. M. Perez, and A. F. G. Skarmeta, "Detection of semantic conflicts in ontology and rule-based information systems," *Data & Knowledge Engineering*, vol. 69, no. 11, pp. 1117 – 1137, 2010.