ENFORCING ACCESS CONTROL DELEGATION TO SECURE AND PRESERVE PRIVACY OF CLOUD DATA

Nicholaus Gati, K. Suresh Babu

Abstract— Conventional access control models often assume that the entity enforcing access control policies is also the owner of the data. This assumption is no longer holding as it forces the data owner to do a lot of computations as the third party such as cloud only provide facilities for data storage, where the approaches to enforce fine grained access control on confidential data hosted in the cloud are based on fine grained encryption of data. Under these models the owner of data is forced to perform the fine grained encryption of data before uploading on the cloud and once user dynamics or credentials change the data owner must download re-encrypt and re-upload the data. Data owners thus incur high computational and communication costs. A better approach should delegate the enforcement of fine- grained access control to the cloud and provide a separate Access control provider, so to minimize the overheads at the data owner, while assuring data confidentiality from the cloud. The proposed approach that can well delegate the enforcement of access control is based on two layers of encryption (TLE), where the data owner performs course-grained encryption and the clouds perform fine grained encryption on top of the owner encrypted data. An efficient AES algorithm is used to provide higher confidentiality and privacy for several users in the cloud and stores the data in multi-clouds where the users can retrieve with the keys later while delegating it through access control from the cloud. To provide more protection to the cloud cloudTraceBack and Cloud protector is user to protect against XDoS/HDoS attacks. This system assures confidentiality, integrity of data and preserves the privacy of the end user from multiple clouds while delegating most of the access control enforcement to the cloud

Index Terms— Privacy preserving, Access Control, Protection, Delegation, Layer Interleaving, Policy breakdown, Data privacy, DDoS

1 INTRODUCTION

Cloud computing is an emerging paradigm that offers a cost effective way for outsourcing data storage and computation. Nevertheless, despite its intriguing properties, enterprises are reluctant to fully adopt it, since they are concerned–among other things–about losing the governance of their outsourced assets, i.e., losing the ability to enforce their own, enterprise specific, security policies. According to PwC's Global State of Information Security Survey 2012 [13], the largest perceived Cloud security risk is the "uncertain ability to enforce provider security policies", whereas according to the survey of Subashini and Kavitha [14] one of the biggest security challenges for providing Cloud-based services is the "adherence of the Cloud provider to the security policies of its clients", as well as "the administration of user authorization systems". An approach to mitigate these concerns is the use of encryption. However, whereas encryption assures the confidentiality of the data against the cloud, the use of conventional encryption approaches is not sufficient to support the enforcement of fine-grained organizational access control policies (ACPs). Many organizations have today ACPs regulating which users can access which data; these ACPs are often expressed in terms of the properties of the users, referred to as identity attributes, using access control languages such as XACML. Expressive access control model (XACML), allows one to specify ACPs on protected objects in terms of the properties of subjects, referred to as identity attributes. The email address, the role a user plays in her organization; the age and the location a user accesses from are a few examples of such identity attributes. The identity attributes that subjects should possess in order to access protected objects are referred to as conditions. Such an attribute based access control model is crucial in order to support fine grained access control policies to data.

With the involvement of the third-party cloud services, a crucial issue is that the identity attributes in the access control policies may reveal privacy-sensitive information about users and organizations and leak confidential information about the content. The confidentiality of the content and the privacy of the users are thus not assured if the identity attributes are not protected. It is well-known that privacy, both individual as well as organizational, is considered a key requirement in all solutions, including cloud services, for digital identity management. Further, as insider threats [15] are one of the major sources of data theft and privacy breaches, identity attributes must be strongly protected even from accesses within organizations. With initiatives such as cloud computing the scope of insider threats is no longer limited to the organizational perimeter.

However, while the existing approach addresses some limitations of previous approaches, it still requires the data owner to enforce all the Access control policies by fine-grained encryption, both initially and subsequently after users are added/ revoked or the Access control policies change. All these encryption activities have to be performed at the owner that thus incurs high communication and computation cost. For example, if an access control policy changes, the owner must download from the cloud the data covered by this access control policy, generate a new encryption key, re-encrypt the downloaded data with the new key, and then upload the re-encrypted data to the cloud.

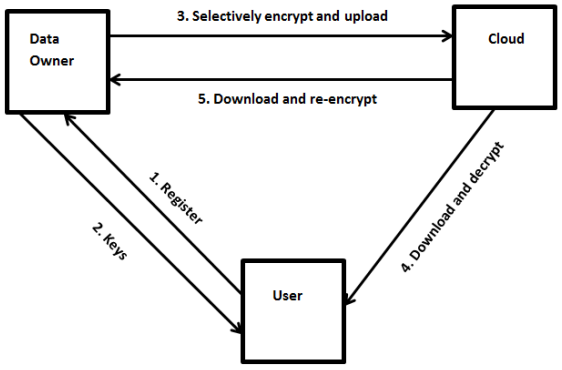

Figure 1: Traditional approach

Therefore, the better way to reduce the communication and computational cost, is to use the two layer encryption approach, where the fine grained encryption is enforced at the cloud and the data owner enforce the course grained encryption. Data integrity is defined as the accuracy and consistency of stored data, in absence of any alteration to the data between two updates of a file or record. Cloud services should ensure data integrity and provide trust to the user privacy. Hence, the system must have some sort of mechanism to ensure the data integrity. The current Cloud security model is based on the assumption that the user/customer should trust the provider. This is typically governed by a Service Level Agreement (SLA) that in general defines mutual provider and user expectations and obligations.

In order to ensure the integrity and availability of data in Cloud and enforce the quality of cloud storage service, efficient methods that enable on-demand data correctness verification on behalf of cloud users have to be designed. However, the fact that users no longer have physical possession of data in the cloud prohibits the direct adoption of traditional cryptographic primitives for the purpose of data integrity protection. Hence, the verification of cloud storage correctness must be conducted without explicit knowledge of the entire data files. The data stored in the cloud may not only be accessed but also be frequently updated by the users, including insertion, deletion, modification, appending, etc. Thus, it is also imperative to support the integration of this dynamic feature into the cloud storage correctness assurance, which makes the system design even more challenging.

The paper is organized as follows. Section 2 discusses related work. Section 3 describes in details our scheme. Section 4 describes decomposition algorithm. Section5 reports experimental results for policy decomposition algorithm, and AES. Finally, Section 6 concludes the paper.

2. RELATED WORK

Access Control: In many application scenarios, such as those in enterprises or organizations, users' access to data is usually selective and highly differentiated. Different users enjoy different access privileges with regard to the data. When data are outsourced to the cloud, enforcing secure, efficient, and reliable data access among a large number of users is thus critical. Traditionally, to control the dissemination of privacy-sensitive data, users establish a trusted server to store data locally in clear, and then control that server to check whether requesting users present proper certification before letting them access the data. From a security standpoint, this access control architecture is no longer applicable when we outsource data to the cloud. Because data users and cloud servers aren't in the same trusted domain, the server might no longer be fully trusted as an omniscient reference monitor for defining and enforcing access control policies and managing user details. In the event of either server compromise or potential insider attacks, users' private data might even be exposed. One possible approach to enforce data access without relying on cloud servers could be to encrypt data in a differentiated manner and disclose the corresponding decryption keys only to authorized users. This approach usually suffers from severe performance issues, however, and doesn't scale, especially when a potentially large number of on-demand users desire fine-grained data access control. Researchers have been working on how to realize a fine-grained access control design that fully leverages the cloud's computation resource richness. Via this approach, data users would be able to securely delegate to the cloud most cumbersome user/ data management workloads — such as handling frequent user access privilege updates in large dynamic systems — while still preserving the underlying data confidentiality against any unauthorized access.

Attribute Based Encryption: The concept of attribute-based encryption (ABE) has been introduced by Sahai and Waters [17]. ABE can be considered as a generalization of identity based encryption (IBE), where the encryption is based on some identity. Thus, ABE is more expressive than IBE. In an ABE system, the plaintext is encrypted with a set of attributes. The key generation server, which possesses the master key, issues different private keys to users after authenticating the attributes they possess. Thus, these private keys are associated with the set of attributes each user possesses. In its basic form, a user can decrypt a ciphertext if and only if there is a match between the attributes of the ciphertext and the user's key. The initial ABE system is limited only to threshold policies where there should be at least k out of n attributes common between the attributes used to encrypt the plaintext and the attributes users possess. Pirretti et al. [20] gave an implementation of such a threshold ABE system using a variant of the Sahai-Waters Large Universe construction [17]. Since the initial threshold scheme, a few variants have been introduced to provide more expressive ABE systems. Goyal et al. [21] introduced the idea of key-policy ABE (KP-ABE) systems and Bethencourt et al. [1] introduced the idea of ciphertext-policy ABE (CP-ABE) systems. Even though these constructs are expressive and provably secure, it is hard to support group management, especially to provide forward security when a user leaves the group (i.e. attribute revocation) and to provide backward security when a new user joins the group. Some of the above schemes suggest using an expiration attribute along with other attributes. However, such a solution is not suitable for a dynamic group where joins and departures are frequent.

Approach to Manage Group Encryption Keys: An approach to support fine-grained selective ABAC is to identify the sets of data items to which the same access control policy (or set of policies) applies and then encrypt each such set with the same encryption key. The encrypted data is then uploaded to the cloud and each user is given the keys only for the set(s) of data items that it can access according to the policies. Such approach addresses two requirements: (a) protecting data confidentiality from the cloud; (b) enforcing fine-grained access control policies with respect to the data users. A major issue in such an approach is represented by key management, as each user must be given the correct keys with respect to the access control policies that the user satisfies. One approach to such issue is to use a hybrid solution whereby the data encryption keys are encrypted using a public key cryptosystem such as attribute based encryption (ABE) and/or proxy re-encryption (PRE). However, such an approach has several weaknesses: it cannot efficiently handle adding/revoking users or identity attributes, and policy changes; it requires keeping multiple encrypted copies of the same key; it incurs high computational costs; it requires additional attributes to support revocation. Therefore, a different approach is required. It is also worth noting that a simplistic group key management (GKM) scheme by which the

content publisher directly delivers the symmetric keys to the corresponding users has some major drawbacks with respect to user privacy and key management. On one hand, user private information encoded in the user identity attributes is not protected in the simplistic approach. On the other hand, such a simplistic key management scheme does not scale well when the number of users becomes large and multiple keys need to be distributed to multiple users.

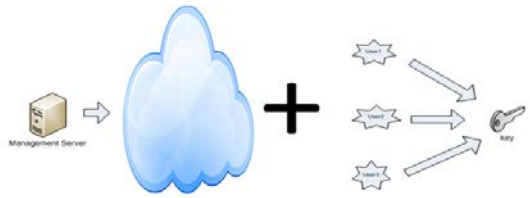

Figure 2: Shows the concept behind GKM

Fine-grained Access Control: Fine-grained access control (FGAC) allows one to enforce selective access to the content based on expressive policy specifications. FGAC can be categorized into two dissemination models: push based and pull-based models. In a push-based system, content publishers push the content to users either by broadcasting or making the content available in a public location. In a pull based system, every time users want to access some content, they login to the content provider and retrieve based on the access control policies. Under the push-based model, sub-documents are encrypted with different keys, which are provided to users at the registration phase, and broadcast the encrypted sub-documents to all users. However, such approaches require all or some keys be distributed in advance during user registration phase. This requirement makes it difficult to assure forward and backward key secrecy when user groups are dynamic with frequent join and leave operations. Further, the rekey process is not transparent, thus shifting the burden of acquiring new keys on existing users when others leave or join. In contrast, our approach makes rekey transparent to users by not distributing actual keys during the registration phase. Another distinction is that all these approaches focus on achieving confidentiality of the content and privacy of the users who access the content is not considered. Under the pull-based model, the content publisher is required to be online in order to access the content. There have been some recent research efforts to construct privacy preserving access control systems by combining oblivious transfer and anonymous credentials.

The "provable data possession" (PDP) model for ensuring possession of file on untrusted storages was defined by Ateniese et al [6]. Their scheme utilized public key based homomorphic tags for auditing the data file, thus providing public verifiability. However, their scheme requires sufficient computation overhead that can be expensive for an entire file. Later in their subsequent work during 2008, described a PDP scheme that uses only symmetric key cryptography. This method has lower-overhead than their previous scheme and allows for block updates, deletions and appends to the stored file, which has also been supported in our work. However, their scheme focuses on single server scenario and does not address small data corruptions, leaving both the distributed scenario and data error recovery issue unexplored.

XML-based denial of service (X-DoS) attacks. A Denial of Service (DoS) is where an attacker attempts to deprive legitimate users of their resources (Rogers, 2009). An X-DoS attack, according to Padmanabhuni et al. (2007) is where a network is flooded with XML messages instead of packets in order to prevent legitimate users to access network communications. Further, if the attacker floods the web server with XML requests, it will affect the availability of these web services. Attackers can also manipulate the message content, in order to cause the web server to crash as shown in Jensen et al. (2007). To adapt X-DoS into a Distributed Denial of Service paradigm, called Distributed XML based Denial of Service (DX-DoS); the attacker uses multiple hosts to attack the victim with X-DoS attacks

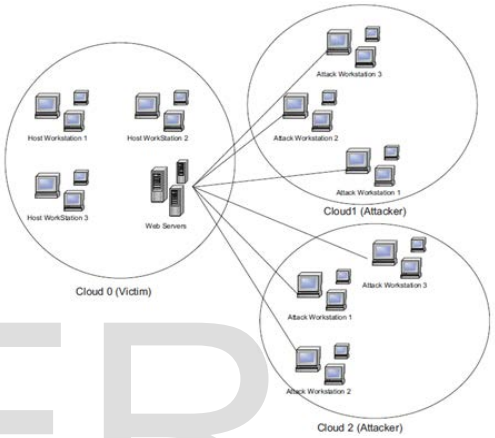

Figure 3: Distributed XML-based Denial of Service attack

Swift-based architectures A Swift-based object storage architecture is composed by two networks: the internal (private) network that consists of storage nodes, and the external (public) network that consists of a proxy server and (optionally) an authentication server. The proxy server accepts HTTP requests and processes them using a Web Server Gateway Interface. The parameters used in each request are encoded as HTTP headers. Each request is pipelined through a number of add-ons, each of which may transform it, forward it, or respond on behalf of the system to the user. Objects stored in a Swift-based architecture are organized in a three level hierarchy. The topmost level of this hierarchy is the accounts level, followed by the containers level (second level) and the objects level (third level). The accounts level contains user accounts. Each user account is associated with many containers from the containers level. A container is used for organizing objects, therefore a container is associated with many objects from the objects level. An object may be a file or a folder (that contains other objects). Every object within a container is identified by a container-unique name. Each request for an operation over an object contains a URI that denotes the account, the container and the name of the object in question.

Single Sign-On (SSO) systems–such as Kerberos and, more recently, OpenID 2.0 [7] and OAuth 2.0 [8]–have similar goals with our scheme. In these systems, user identity management is performed by a separate trusted entity. Kerberos has been widely used

for providing access control to network resources. In a Kerberos system a Ticket Granting Service (TGS) provides a "ticket" to an authenticated user that enables her to use a resource. The TGS and the resource, however, have to belong to the same administration realm, or they should be preconfigured with a shared secret. In this system there is no restriction on the administrative domains in which the various entities should belong to. Moreover there is no secret with which an entity has to be pre-configured. OpenID is an identity management system that allows third parties to delegate identity management to an Identity Provider (IdP) trusted by the user. In an OpenID system, the IdP is responsible for authenticating the user and for providing a token that proves that a user is authenticated. This token is unique per user, therefore it enables the third party to track user activity. Nunez et al. [9] used OpenID in conjunction with proxy re-encryption in order to provide Cloud based identity management services, whereas Khan et al. [10] have implemented OpenID based authentication mechanisms for the OpenStack platform. OpenID provides only user authentication; in an OpenID-based access control system, the Cloud provider is responsible for evaluating the access control policies. In this system tokens are ephemeral; therefore they cannot be used to track the long term activity of a specific user. In addition, system's access control policy is evaluated by a third trusted party and not by the Cloud provider.

OAuth 2.0 is an IETF standard for authorizing access to resources over HTTP. OAuth 2.0 requires the resource owner to be online during the third party authorization procedure (Section 1.2 of [8]), and requires implicitly the development of a communication protocol between the resource server and the authorization server in order to be able to exchange an access token whose form–as mentioned in Section 1.4 of [8]–is not specified. The latter limitation raises obstacles to implementations in which the resource server and the authorization server belong to different administrative domains. An approach for providing access control using OAuth 2.0 is the following: the data owner defines an access control policy using attributes that can be provided by an authorization server (e.g., user age, as provided by a social network), these attributes are regarded as resources and they are accessed by the Cloud provider using OAuth 2.0; the Cloud provider uses these attributes and evaluates the access control policy. In this scenario, user credentials are protected. However, the Cloud provider learns some information about the user (in this example his age), and has to understand the authorization server specific attributes in order to evaluate the access control policy. In our system the Cloud provider learns nothing about the user and does not have to understand any authorization server-specific semantics.

3. SYSTEM DESIGN

In this section we present our system design. We begin with a high level overview of our scheme and present our goals.
Then we detail the functionality of our system.

A. Scheme overview

In the system the four basic roles are considered: the data owner (owner), the data consumer (user), the Cloud provider (CP), and the access control provider (ACPr). The goal of an owner is to store some data in a CP and allow authorized consumers to perform operations over this data. The data is protected using an encryption and access control policy. An access control policy is regarded as a function executed in an ACPr. This function accepts as input a consumer's identification data and outputs either an error message if the user cannot be authorized, or an integer number that denotes the access level of the consumer. The access level of a consumer indicates which operations she can perform over the data that is protected by the corresponding access control policy.
In our scheme, the following trust relationships are considered: the owner trusts the ACPr to authorize a consumer, and the owner and the consumer trust the CP to respect the decision of the ACPr. The first trust relationship type can be trivially established if the ACPr belongs to the owner (e.g., a leveraged enterprise user management system). The second trust relationship is a relaxed form of the currently existing trust relationship between an owner and a Cloud provider: currently, in the best case, an owner trusts a Cloud provider to securely store the owner's business logic, to execute it correctly and to enforce its outcome.

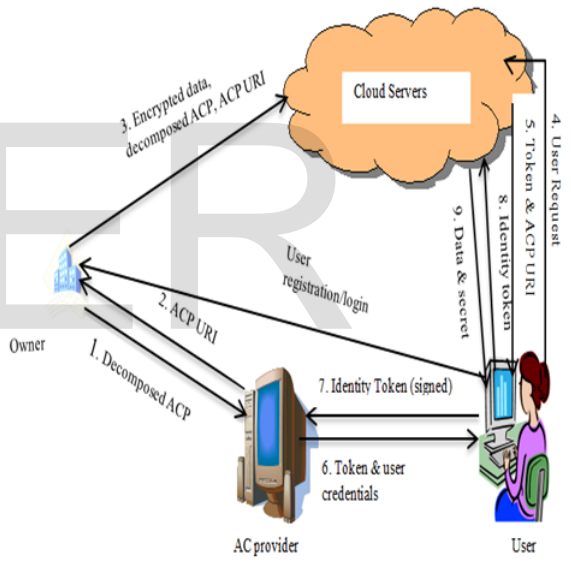

Figure 4: Proposed system architecture

B. Detailed system description

A high-level view of the interactions between the system entities is illustrated in Figure 3. An execution round of the system includes the following steps. Initially the owner stores access control policy in the ACPr and obtains a URI for that policy. As a next step she communicates the obtained URI, as well as the data it protects, to a CP, specifying at the same time the required access level(s) for each operation. When a consumer tries to perform an operation over some protected data for the first time, she receives as a response from the CP a token and the URI of the access control policy that protects the data item requested, and she is being redirected to the appropriate ACP. Then, the consumer authenticates herself to the

ACPr, by providing some form of identification data, and requests authorization, based on the access control policy that corresponds to the obtained URI. The ACPr checks if the consumer satisfies the stored access control policy; if this is true, the ACPr signs the token, including in the signature the consumer's access level and also provide the secrets for creation of decryption key. The signed token can now be used by the consumer in order to perform the desired operation.

4. DECOMPOSITION ALGORITHM

An important issue in the TLE approach is how to distribute the encryptions between the Owner and the Cloud. There are two possible extremes. The first approach is for the Owner to encrypt all data items using a single symmetric key and let the Cloud perform the complete access control related encryption. The second approach is for the Owner and the Cloud to perform the complete access control related encryption twice. The first approach has the least overhead for the Owner, but it has the highest information exposure risk due to collusions between Users and the Cloud. Further, IEL updates require re-encrypting all data items. The second approach has the least information exposure risk due to collusions, but it has the highest overhead on the Owner as the Owner has to perform the same task initially as in the SLE approach and, further, needs to manage all identity attributes. An alternative solution is based on decomposing ACPs so that the information exposure risk and key management overhead are balanced. The problem is then how to decompose the ACPs such that the Owner has to manage the minimum number of attributes while delegating as much access control enforcement as possible to the Cloud without allowing it to decrypt the data. In what follow we propose such an approach to decompose and we also show that the policy decomposition problem is hard.
Policy decomposition Algorithm takes the ACP as input and produces the two sets of ACPs ACPOwner and ACPCloud that are to be enforced at the Owner and the Cloud respectively. It first converts each policy into DNF (disjunctive normal form) and decomposes each conjunctive term into two conjunctive terms such that one conjunctive term has only those ACs in ACPi and the other term may or may not have the ACs in ACPi. It can be easily shown that the policy decomposition is consistent.
Policy decomposition algorithm

Input: access control policies.
Output: decomposed policies.
Method:-
1. First ACPO = null and ACPC= null.
2. Convert the given ACPi into DNF to form an expression of attributes.
3. if (any conjunctive term appears into the expression) Then decompose that term into c1 and c2
Such that c1=ACPi (owner) and c2=ACPi(cloud). And c1 AND c2=c.
4. if (multiple conjunctive terms are appeared) Then repeat step 3, and Add ACPi (owner) to ACPO and Add ACPi (cloud) to ACPC, until all conjunctive terms are decomposed and added to ACPC and ACPO
5. Stop.

5. ANALYSIS

A. Comparing Approaches

In this section the comparison between SLE and TLE approaches was made in term of time used to generate keys with respect to the algorithms used in each of the approaches, figure 5 shows BGKM used in the new approach is more effective, as it uses less time to derive a good number of keys. The TLE approach reduces the overhead incurred by the Owner during the initial encryption as well as subsequent re-encryptions. In this approach, the Owner handles only the minimal set of attribute conditions and most of the key management tasks are performed by the Cloud. Further, when identity attributes are added or removed, or the Owner updates the Cloud's ACPs, the Owner does not have to re-encrypt the data as the Cloud performs the necessary re-encryptions to enforce the ACPs. Therefore, the TLE approach reduces the communication and computation overhead at the Owner.

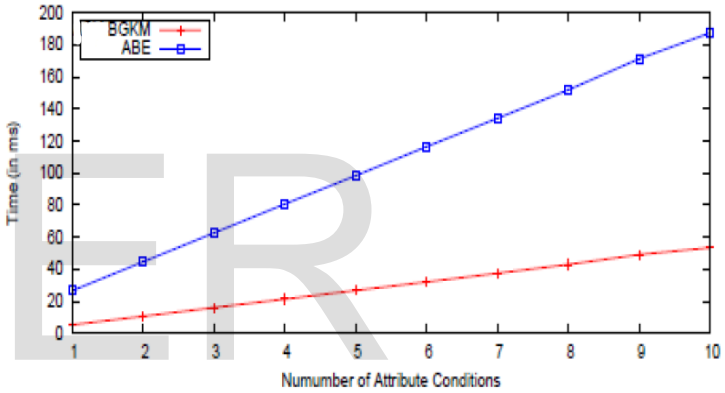

Figure 5: Average time to generate keys for different number of attributes

.

B. Security and Privacy

The existing approach correctly enforces the ACPs through encryption, although the data owner performs the attribute based encryption based on ACPs. The newly introduced key management scheme (AB-GKM) scheme creates an environment such that only users that satisfies the ACP are able to derive the encryption keys, thus only users with privilege are capable to access the data. The TLE approach correctly enforces the ACPs through two encryptions, where by each ACP is decomposed into two ACPs such that the conjunction of them is equivalent to the original ACP. The Owner enforces one part of the decomposed ACPs that has the minimum number of attributes through attribute based encryption. The Cloud enforces the counterparts of the decomposed ACPs through another attribute based encryption. Only users that are capable of decrypting both encryptions can access the data. As the AB-GKM scheme makes sure that only those users who satisfy these decomposed policies can derive the corresponding keys, a user can access a data item by decrypting twice only if it satisfies the two parts of the decomposed ACPs, that is, the original ACPs.

In both approaches, the privacy of the identity attributes of users is assured. From AB-GKM scheme the algorithm SecGen issues secrets to users based on the identity tokens which hide the identity attributes. Further, at the end of the algorithm neither the Owner nor the Cloud knows if a user satisfies a given attribute condition. Therefore, neither the Owner nor the Cloud learns the identity attributes of users.

6. CONCLUSION AND FUTURE WORK

Current trends in cloud computing and associated services are further pushing publishing functions to third-party providers to achieve flexibility and economies of scale. However, recent surveys have found that one of the key resistance factors for organizations to move to the cloud is represented by data privacy and security concerns. The proposed approach with two layers encryption and cloudTraceBack/cloudprotector provides the required privacy and security and also delegates much of the access control enforcement responsibilities to the cloud while minimizing information exposure risks due to colluding users and cloud, on top of all that it protects the cloud from XDoS attacks by detecting malicious traffic and stopping them from entering the cloud network while monitoring the performance of the cloud network. Policy decomposition which is very instrumental in the proposed approach was handles with the policy decomposition algorithm so that the Owner has to handle a minimum number of attribute conditions while hiding the content from the Cloud and access control policy sub set for the cloud is kept very carefully in the trusted access control provider. The approach is based on a privacy preserving attribute based key management scheme that protects the privacy of users while enforcing attribute based ACPs. As future work, the plan is to investigate the alternative choices for the TLE approach further.

REFFERENCES

[1]. J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In SP 2007: Proceedings of the 28th IEEE Symposium on Security and Privacy, 2007.
[2]. M. Nabeel and E. Bertino. Towards attribute based group key management. In CCS 2011: Proceedings of the 18th ACM conference on Computer and communications security, 2011.
[3]. M. Nabeel, N. Shang, and E. Bertino. Privacy preserving policy based content sharing in public clouds. IEEE Transactions on Knowledge and Data Engineering, 99, 2012
[4]. N. Shang, M. Nabeel, F. Paci, and E. Bertino. A privacy-preserving approach to policy-based content dissemination. In ICDE 2010: Proceedings of the 2010 IEEE 26th International Conference on Data Engineering, 2010.
[5]. S. Yu, C. Wang, K. Ren, and W. Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In INFOCOM 2010: Proceedings of the 29th conference on Information communications.
[6]. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in Proc. of CCS'07, Alexandria, VA, October 2007, pp. 598–609.
[7]. D. Recordon and D. Reed, "OpenID 2.0: a platform for user-centric identity management," in Proc. of the 2nd ACM workshop on Digital Identity Management, 2006, pp. 11–16.
[8]. D. Hardt (ed.), "The OAuth 2.0 authorization framework," RFC 6749, October 2012.
[9]. D. Nunez, I. Agudo, and J. Lopez, "Integrating OpenID with proxy re-encryption to enhance privacy in cloud-based identity services," in Proc of the IEEE 4th International Conference on Cloud Computing Technology and Science, 2012.
[10]. R. Khan, J. Ylitalo, and A. Ahmed, "OpenID authentication as a service in OpenStack," in Proc. of the 7th International Conference on Information Assurance and Security, 2011, pp. 372–377.
[11]. Padmanabhuni S; Singh V. Senthil kumar KM, Chatterjee A. Web services, preventing service oriented denial of service (PreSODoS): a proposed approach. In: Proceedings of the ICWS apos;06, international conference on volume , issue, September 2006. p.577–84.
[12]. Jensen M, Gruschka N, Herkenh R, Luttenberger N. SOA and web services: new technologies, new standards – new attacks. In: Proceedings of the fifth European conference on web services, 0-7695-3044-3/07, 2007.
[13] PwC, "Global state of information security survey," 2012.
[14] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," Journal of Network and Computer Applications, vol. 34, no. 1, pp. 1–11, 2011.
[15] R. Richardson. CSI Computer Crime and Security Survey. Technical report, Computer Security Institute, 2008.
[16] Priya Dhawan , "Performance Comparison: Security Design Choices," Microsoft Developer Network October 2002. http://msdn2.microsoft.com/en-us/library/ms978415.aspx
[17] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in Eurocrypt 2005, LNCS 3494. Springer-Verlag, 2005, pp. 457– 473.
[18] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In CRYPTO '01: Proceedings of the 21st Annual International Clyptology Conference on Advances in Clyptology, pages 213-229. Springer-Verlag, 2001
[19] C. Cocks. An identity based encryption scheme based on quadratic residues. In Proceedings of the 8th IMA international Conference on Cryptography and Coding, pages 360-363, London, UK, 2001. Springer-Verlag.
[20] M. Pirretti,P . Traynor,P . McDaniel, and B. Waters. Secure attribute based systems. In CCS '06: Proceedings of the 13th ACM conference on Computer and communications security, pages 99- 1 12, New York, NY, USA, 2006. ACM.
[21] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In CCS '06: Proceedings of the 13th ACM coriference on Computer and communications security, pages 89-98, New York, NY, USA, 2006. ACM.
[22] G. Miklau and D. Suciu. Controlling access to published data using cryptography. In VLDE '2003: Proceedings of the 29th international coriference on VelY large data bases, pages 898-909. VLDB Endowment, 2003.
[23] E. Bertino and E. Ferrari. Secure and selective dissemination of XML documents. ACM Trans. in! Syst. Secw:, 5(3):290-331, 2002.
[24] J. Camenisch, M. Dubovitskaya, and G. Neven. Oblivious transfer with access control. In CCS '09: Proceedings of the 16th

ACM conference on Computer and communications security, pages 13 1- 140, New York, NY, USA, 2009. ACM.
[25] S. Coull, M. Green, and S. Hohenberger. Controlling access to an oblivious database using stateful anonymous credentials. In Irvine: Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography, pages 501-520,B erlin,Heidelberg, 2009. Springer-Verlag.

AUTHOR'S PROFILE

NICHOLAUS GATI Received BSc degree in Computer Engineering and Information Technology from the University Of Dar Es Salaam, Tanzania. He is currently pursuing M.Tech in Computer Networks and Information Security from School of Information Technology Jawaharlal Nehru Technological University, Hyderabad, India. His research interests include cloud computing security and storage, wireless networks security, distributed systems and computing.

K. SURESH BABU has completed his M.Tech. (Computer Science) from Hyderabad Central University (HCU), Hyderabad and presently pursuing his Ph.D. from JNT University Hyderabad in the field of Network Secuirty in MANETs. He has a teachingexperience of 12 years. His subjects of interests are Computer Networks, Network Security, Operating Systems, Wireless Networks, mobile Computing, Ethical Hacking and Wireless & Web Security. He has published serveral papers in both National and international Journals. He also participated and presaented papers in International & National conferences and seminars.

IJSER