A SURVEY ON SECURITY ENHANCEMENT IN CLOUD BY CIPHERTEXT-POLICY ATTRIBUTE BASED ENCRYPTION

Dr. P. Boobalan[1], A. Shahin Nisha[2]
[1]Associate Professor
Department of Information Technology, Pondicherry Engineering College, Pondicherry
[2]Master of Technology, Department of Information Technology
Pondicherry Engineering College, Pondicherry

ABSTRACT
In cloud computing, security remains a primary concern for businesses contemplating cloud adoption especially public cloud adoption. Public cloud service providers share their underlying hardware infrastructure between numerous customers, as public cloud is a multi-tenant environment. This environment demands copious isolation between logical compute resources. At the same time, access to public cloud storage and computing resources is guarded by account login credentials. These details of the cloud users are considered as attributes in Attribute Based Encryption (ABE). Attribute-based encryption is a relatively recent approach that reconsiders the concept of public-key cryptography in cloud. In traditional public-key cryptography, a message is encrypted for a specific receiver using the receiver's public-key. Identity-based cryptography and in particular identity-based encryption (IBE) changed the traditional understanding of public-key cryptography by allowing the public-key to be an arbitrary string, e.g., the email address of the receiver. ABE goes one step further and defines the identity not atomic but as a set of attributes, e.g., roles, and messages can be encrypted with respect to subsets of attributes (key-policy ABE (KP-ABE) or policies defined over a set of attributes (ciphertext-policy ABE - CP-ABE). The key issue is, that someone should only be able to decrypt a ciphertext if the person holds a key for "matching attributes" (more below) where user keys are always issued by some trusted party. In this paper, we focus on the various types of attribute based encryption to address different security issues in cloud.

Keywords: Attribute Based Encryption(ABE), Access Control, Key Policy Attribute Based Encryption(KP-ABE), Ciphertext-Policy Attribute Based Encryption(CP-ABE), User Access Structure.

I. INTRODUCTION TO CLOUD COMPUTING

Cloud computing is a computing paradigm, where a large pool of systems are connected in private or public networks, to provide dynamically scalable infrastructure for application, data and file storage. With the advent of this technology, the cost of computation, application hosting, content storage and delivery is reduced significantly. Cloud computing is a practical approach to experience direct cost benefits and it has the potential to transform a data center from a capital-intensive set up to a variable priced environment. The idea of cloud computing is based on a very fundamental principal of reusability of IT capabilities'.

Several computing paradigms such as Grid computing have promised to deliver this utility computing vision. Cloud computing is the most recent emerging paradigm promising to turn the vision of "computing utilities" into a reality. It is based on the concept of dynamic provisioning, which is applied not only to services, but also to compute capability, storage, networking, and Information Technology (IT) infrastructure in general.

a. Benefits

The following are some of the possible benefits for those who offer cloud computing-based services and applications:

• Cost Savings

Companies can reduce their capital expenditures and use operational expenditures for increasing their computing capabilities. This is a lower barrier to entry and also requires fewer in-house IT resources to provide system support.

• Scalability/Flexibility

Companies can start with a small deployment and grow to a large deployment fairly rapidly, and then scale back if necessary. Also, the flexibility of cloud computing allows companies to use extra resources at peak times, enabling them to satisfy consumer demands.

• Reliability

Services using multiple redundant sites can support business continuity and disaster recovery.

• Maintenance

Cloud service providers do the system maintenance, and access is through APIs that do not require application installations onto PCs, thus further reducing maintenance requirements.

• Mobile Accessibility

Mobile workers have increased productivity due to systems accessible in an infrastructure available from anywhere.

b. Challenges

The following are some of the notable challenges associated with cloud computing, and although some of these may cause a slowdown when delivering more services in the cloud, most also can provide opportunities, if resolved with due care and attention in the planning stages.

• Security and Privacy

Perhaps two of the more "hot button" issues surrounding cloud computing relate to storing and securing data, and monitoring the use of the cloud by the service providers. These issues are generally attributed to slowing the deployment of cloud services. These challenges can be addressed, for example, by storing the information internal to the organization, but allowing it to be used in the cloud. For this to occur, though, the security mechanisms between organization and the cloud need to be robust and a Hybrid cloud could support such a deployment.

• Lack of Standards

Clouds have documented interfaces; however, no standards are associated with these, and thus it is unlikely that most clouds will be interoperable. The Open Grid Forum is developing an Open Cloud Computing Interface to resolve this issue and the Open Cloud Consortium is working on cloud computing standards and practices. The findings of these groups will need to mature, but it is not known whether they will address the needs of the people deploying the services and the specific interfaces these services need. However, keeping up to date on the latest standards as they evolve will allow them to be leveraged, if applicable.

• Continuously Evolving

User requirements are continuously evolving, as are the requirements for interfaces, networking, and storage. This means that a "cloud," especially a public one, does not remain static and is also continuously evolving.

• Compliance Concerns

The Sarbanes-Oxley Act (SOX) in the US and Data Protection directives in the EU are just two among many compliance issues affecting cloud computing, based on the type of data and application for which the cloud is being used. The EU has a legislative backing for data protection across all member states, but in the US data protection is different and can vary from state to state. As with security and privacy mentioned previously, these typically result in Hybrid cloud deployment with one cloud storing the data internal to the organization.

II. ATTRIBUTE BASED ENCRYPTION

Attribute-based encryption is a type of public-key encryption in which the secret key of a user and the ciphertext are dependent upon attributes (e.g. the country he lives, or the kind of subscription he has). In such a system, the decryption of a ciphertext is possible only if the set of attributes of the user key matches the attributes of the ciphertext. A crucial security feature of Attribute-Based Encryption is collusion-resistance: An adversary that holds multiple keys should only be able to access data if at least one individual key grants access.

ABE uses a tree-based access structure which must be satisfied with a given set of attributes in order to decrypt the data. It uses operators such as AND, OR and k-of-n. AND is usually known as 'n of n' and OR is known as '1 of n'. For example, if A wants to encrypt a data P such that only someone with the attributes friend AND colleague or the attribute family can decrypt it, the tree-based access structure would look like Figure 1.

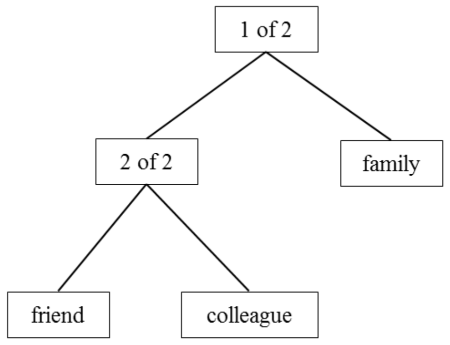

Figure 1: ABE Access Structure

III. TYPES OF ATTRIBUTE BASED ENCRYPTION

a. Key-Policy ABE
b. Ciphertext-Policy ABE

a. Key Policy Attribute Based Encryption

KP-ABE is the dual to CP-ABE in the sense that an access policy is encoded into the users secret key, e.g., (A∧C)∨D(A∧C)∨D, and a ciphertext is computed with respect to a set of attributes, e.g., {A,B}{A,B}. In this example the user would not be able to decrypt the ciphertext but would for instance be able to decrypt a ciphertext with respect to {A,C}{A,C}.An important property which has to be achieved by both, CP-ABE and KP-ABE is called collusion resistance. This basically means that it should not be possible for distinct users to "pool" their secret keys such that they could together decrypt a ciphertext that neither of them could decrypt on their own (which is achieved by independently randomizing users' secret keys).

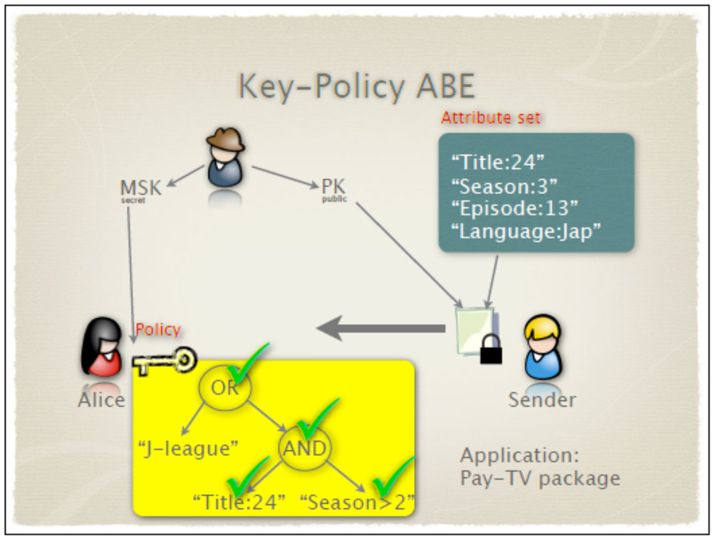

Figure 2: Key-Policy ABE

It is the modified form of classical model of ABE. Users are assigned with an access structure (AS) over the data attributes[5]. To reflect the access structure the secret key of the user is defined. Cipher texts are labeled with sets of attribute and private keys are associated with monotonic access structure that control which cipher texts a user is able to decrypt. Key policy Attribute Based Encryption (KP-ABE) scheme is designed for one-to-many communications.
Algorithm takes input K as a security parameter and returns PK as public key and the system master secret key MK. PK is used by message senders for encryption.MK is used to generate user secret keys and is known only to the authority. For encryption algorithm takes a message M, the public key (PK), and a set of attribute as input. It outputs the cipher text (CT). Key generation algorithm takes as input an access structure (AS) and the master secret key MK. It outputs as a secret key SK that enables the user to decrypt the message encrypted under a set of attributes if and only if matches Access Tree [8]. Decryption is possible only if the attribute set satisfies the user's access structure. The KP-ABE scheme can achieve secured access control and more flexibility to control users than ABE scheme[6].

Drawbacks

The problem with KP-ABE scheme is encryptor cannot decide who can decrypt the encrypted data. It can only choose descriptive attributes for the data, it is unsuitable in some application because a data owner has to trust the key issuer.

b. Ciphertext-Policy Attribute Based Encryption

In ciphertext-policy attribute-based encryption (CP-ABE) a user's private-key is associated with a set of attributes and a ciphertext specifies an access policy over a defined universe of attributes within the system. A user will be ale to decrypt a ciphertext, if and only if his attributes satisfy the policy of the respective ciphertext. Policies may be defined over attributes using conjunctions, disjunctions and (k,n)(k,n-1)threshold gates, i.e., k out of nattributes have to be present (there may also be non-monotone access policies with additional negations and meanwhile there are also constructions for policies defined as arbitrary circuits). For instance, let us assume that the universe of attributes is defined to be {A,B,C,D}{A,B,C,D} and user 1 receives a key to attributes {A,B}{A,B} and user 2 to attribute {D}{D}. If a ciphertext is encrypted with respect to the policy (A∧C)∨D(A∧C)∨D, then user 2 will be able to decrypt, while user 1 will not be able to decrypt.

Ciphertext-policy attribute based encryption (CPABE) is becoming very important in distributed computing environment, because it makes easier to protect, broadcast and control the access of information, especially over the cloud server [5]. In CP-ABE every plaintext is encrypted under an access structure, defined on the user's attribute and users have given private keys in advance from the trusted and reliable authority. If the user's attributes satisfy the access structure then only user can decrypt the ciphertext using his/her private keys. But, there is one privacy issue in available CP-ABE schemes, owner sends the access structure along with the ciphertext and everyone can learn the access policy.

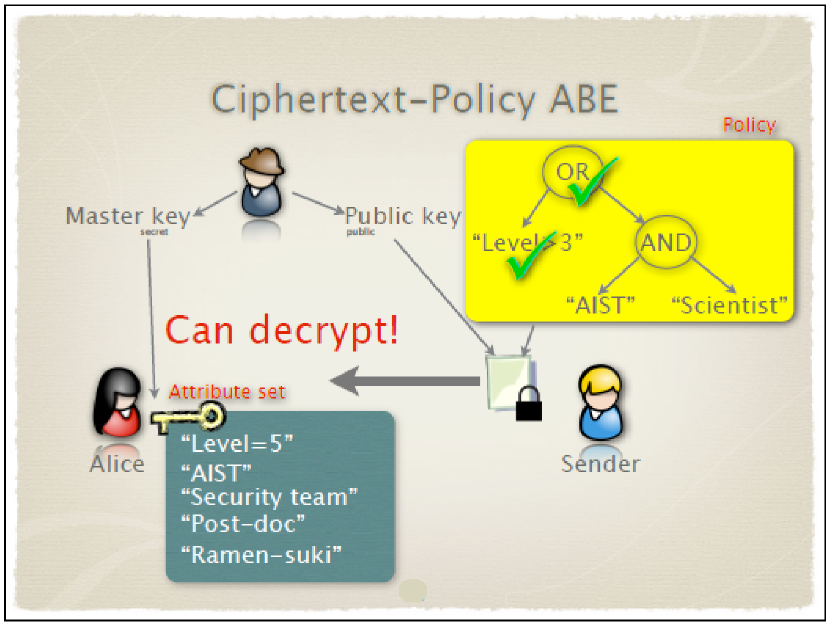

Figure 3: Ciphertext-Policy ABE

CP-ABE thus allows to realize implicit authorization, i.e., authorization is included into the encrypted data and only people who satisfy the associated policy can decrypt data. Another nice features is, that users can obtain their private keys after data has been encrypted with respect to policies. So data can be encrypted without knowledge of the actual set of users that will be able to decrypt, but only specifying the policy which allows to decrypt. Any future users that will be given a key with respect to attributes such that the policy can be satisfied will then be able to decrypt the data.

CP-ABE is the modified form of KP-ABE. In a CP-ABE scheme, every cipher text is associated with an access policy on attributes, and every user's private key is associated with a set of attributes[3]. A user is able to decrypt a cipher text only if the set of attributes associated with the user's private key satisfies the access policy associated with the cipher text. CP-ABE works in the reverse way of KP-ABE.

As in Figure 3, the algorithm takes as input a security parameter K and returns the public key PK as well as a system master secret key MK. PK is used by message senders for encryption.MK is used to generate user secret keys and is known only to the authority. For encryption of data algorithm takes as input the public parameter PK, a message M, and an access structure AS. it outputs the cipher text C. Key-Generation this algorithm takes as input a set of attribute associated with the user and the master key MK. It outputs a secret key SK that enables the user to decrypt a message encrypted under an access tree structure T if and only if matches T. Decryption of the data only if satisfies the access structure associated with the cipher text CT. It improves the disadvantage of KP-ABE that the encrypted data cannot choose who can decrypt. It can support the access control in the real environment. In addition, the user's private key is in this scheme, a combination of a set of attributes, so an user only use this set of attribute to satisfy in the encrypted data.

Drawbacks

Drawbacks of the most existing CP-ABE schemes are still not fulfilling the enterprise requirements of access control which require considerable flexibility and efficiency. CP-ABE has limitation in terms of specifying policies and managing user attributes. In a CP-ABE scheme, decryption keys only support user attributes that are organized logically as a single set, so the user can only use all possible combinations of attributes in a single set issued in their keys to satisfy policies.

IV. ALGORITHMS

Aciphertext-policy attribute based encryption scheme consists of four fundamental algorithms: Setup, Encrypt, KeyGen, and Decrypt. In addition, we allow for the option of a fifth algorithm Delegate.

i. Setup

The setup algorithm takes no input other than the implicit security parameter. It outputs the public parameters PK and a master key MK.

The setup algorithm will choose a bilinear group G_0 of prime order p with generator g. Next it will choose two random exponents $\alpha, \beta \in Z_p$.

The public key is published as:

$PK = \{G_0, g, h\} = g^{\beta}$,

$f = g^{1/\beta}, e(g,g)^{\alpha}$ and the master key MK.

ii. Encrypt(PK,M, A)

The encryption algorithm takes as input the public parameters PK, a message M, and an access structure A over the universe of attributes. The algorithm will encrypt M and produce a ciphertext CT such that only a user that possesses a set of attributes that satisfies the access structure will be able to decrypt the message. We will assume that the ciphertext implicitly contains A.

The encryption algorithm encrypts a message M under the tree access structure T. The algorithm first chooses a polynomial q_x for each node x (including the leaves) in the tree T. These polynomials are chosen in the following way in a topdown manner, starting from the root node R. For each node x in the tree, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node, that is, $d_x = kx - 1$. Starting with the root node R the algorithm chooses a random $s \in Z_p$ and sets $q_R(0) = s$. Then, it chooses dR other points of the polynomial q_R randomly to define it completely. For any other node x, it sets $q_x(0) = q_{parent}(x)(index(x))$ and chooses d_x other points randomly to completely define q_x. Let, Y be the set of leaf nodes in T . The ciphertext is then constructed by giving the tree access structure T and computing

$CT = T,\ \tilde{C} = Me(g,g)^{\alpha s},\ C = h^s$,

$\forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)})$.

iii. Key Generation(MK,S)

The key generation algorithm takes as input the master key MK and a set of attributes S that describe the key. It outputs a private key SK.

The algorithm first chooses a random $r \in Z_p$, and then random $r_j \in Z_p$ for each attribute $j \in S$. Then it computes the key as

$SK = (D = g^{(\alpha+r)/\beta}$,

$\forall j \in S : D_j = g^r \cdot H(j)^{r_j}, D'_j = g^{r_j})$.

iv. Decrypt(PK, CT, SK)

The decryption algorithm takes as input the public parameters PK, a ciphertext CT, which contains an access policy A, and a private key SK, which is a private key for a set S of attributes. If the set S of attributes satisfies the access structure A then the algorithm will decrypt the ciphertext and return a message M.

We specify our decryption procedure as a recursive algorithm. For ease of exposition we present the simplest form of the decryption algorithm and discuss potential performance improvements in the next subsection.

We first define a recursive algorithm DecryptNode(CT, SK,x) that takes as input a ciphertext CT = (T , C, C˜ $\forall_y \in$ Y : C_y, C'_y), a private key SK, which is associated with a set S of attributes, and a node x from T . If the node x is a leaf node then we let i = att(x) and define as follows: If i∈ S, then

$$\text{DecryptNode}(CT, SK,x) = e(D_i ,C_x) / e(D'_i ,C'_x) = e(g^r \cdot H(i)^{ri} ,h^{qx(0)} / e(g\ ri ,H(i)\ qx(0)) = e(g,g)^{rqx(0)}$$

If i Φ S, then we define DecryptNode(CT, SK,x) = ⊥.

We now consider the recursive case when x is a non-leaf node. The algorithm DecryptNode(CT, SK,x) then proceeds as follows: For all nodes z that are children of x, it calls DecryptNode(CT, SK,z) and stores the output as F_z. Let S_x be an arbitrary k_x-sized set of child nodes z such that $F_z \neq \perp$. If no such set exists then the node was not satisfied and the function returns ⊥.

v.Delegate(SK, S˜)

The delegate algorithm takes as input a secret key SK for some set of attributes S and a set S˜ ⊆ S. It output a secret key SK for the set of attributes S˜. The secret key is of the form

.

SK = (D, $\forall_j \in$S :D_j ,D'_j).

This algorithm chooses random r˜ and $\tilde{r}_k \forall k \in$ S˜. Then it creates a new secret key as

SK = (D˜ = $Df^{\tilde{r}}$,

$\forall k \in \tilde{S} : \tilde{D}_k = D_k g^{\tilde{r}} H(k)^{\tilde{r}k}, D'k = D'k\, g^{\tilde{r}}k$).

The resulting secret key SK is a secret key for the set S. Since the algorithm re-randomizes the key, a delegated key is equivalent to one received directly from the authority.

V. BUILDING AN ACCESS TREE

Access Tree

Let T be a tree representing an access structure. Each non-leaf node of the tree represents a threshold gate, described by its children and a threshold value. If num_x is the number of children of a node x and k_x is its threshold value, then $0 < k_x \leq num_x$. When $k_x = 1$, the threshold gate is an OR gate and when $k_x = num_x$, it is an AND gate. Each leaf node x of the tree is described by an attribute and a threshold value k_x = 1. To facilitate working with the access trees, we define a few functions

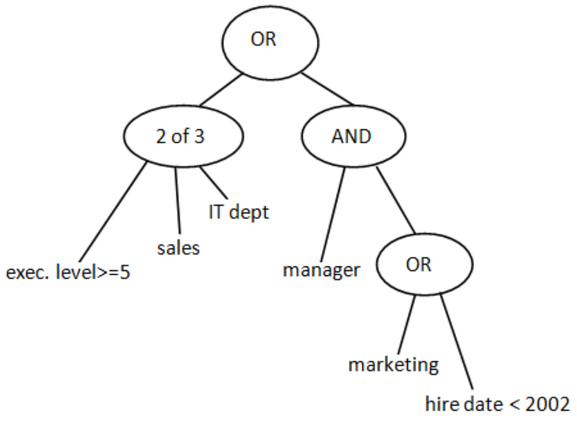

Figure 4: Access Tree with OR and AND gates

We denote the parent of the node x in the tree by parent(x). The function att(x) is defined only if x is a leaf node and denotes the attribute associated with the leaf node x in the tree. The leaf nodes should satisfy the access policies of the parent node i.e, OR gate in Figure 4. The access tree T also defines an ordering between the children of every node, that is, the children of a node are numbered from 1 to num. The function index(x) returns such a number associated with the node x. Where the index values are uniquely assigned to nodes in the access structure for a given key in an arbitrary manner.

Satisfying an access tree

Let T be an access tree with root r. Denote by T_x the subtree of T rooted at the node x. Hence T is the same as T_r. If a set of attributes γ satisfies the access tree T_x, we denote it as $T_x(\gamma) = 1$. We compute $T_x(\gamma)$ recursively as follows. If x is a non-leaf node, evaluate $T_{x'}(\gamma)$ for all children x′ of node x. $T_x(\gamma)$ returns 1 if and only if at least k_x children return 1. If x is a leaf node, then $T_x(\gamma)$ returns 1 if and only if att(x) ∈ γ.

Example

As an example, to give out a private key with the 4-bit attribute “a = 9”, we would instead include “a : 1***”, “a : *0***”, “a : **0*”, and “a : ***1” in the key. We can then use policies of AND and OR gates to implement integer comparisons over such attributes, as shown for “a < 11” in Figure 5.

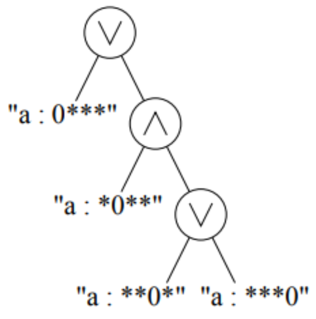

Figure 5: Access tree implementing the integer comparison “a < 11”

There is a direct correspondence between the bits of the constant 11 and the choice of gates. Policies for ≤, >, ≥, and = can be implemented similarly with at most n gates, or possibly fewer depending on the constant. It is also possible to construct comparisons between two numerical attributes (rather than an attribute and a constant) using roughly 3n gates, although it is less clear when this would be useful in practice.

VI. EFFICIENCY

The efficiencies of the key generation and encryption algorithms are both fairly straightforward. The encryption algorithm will require two exponentiations for each leaf in the ciphertext’s access tree[11]. The ciphertext size will include two group elements for each tree leaf. The key generation algorithm requires two exponentiations for every attribute given to the user, and the private key consists of two group elements for every attribute. In its simplest form, the decryption algorithm could require two pairings for every leaf of the access tree that is matched by a private key attribute and (at most2) one exponentiation for each node along a path from such a leaf to the root. However, there might be several ways to satisfy a policy, so a more intelligent algorithm might try to optimize along these lines.

Optimizing the decryption strategy

The recursive algorithm results in two pairings for each leaf node that is matched by a private key attribute, and up to one exponentiation for every node occurring along the path from such a node to the root (not including the root). The final step after the recursive portion adds an additional pairing. Of course, at each internal node with threshold k, the results from all but k of its children are thrown away. By considering ahead of time which leaf nodes are satisfied and picking a subset of them which results in the satisfaction of the entire access tree, we may avoid evaluating DecryptNode where the result will not ultimately be used. More precisely, let M be a subset of the nodes in an access tree T .

We define restrict(T ,M) to be the access tree formed by removing the following nodes from T (while leaving the thresholds unmodified). First, we remove all nodes not in M. Next we remove any node not connected to the original root of T along with any internal node x that now has fewer children than its threshold k_x. This is repeated until no further nodes are removed, and the result is restrict(T ,M). So given an access tree T and a set of attributes γ that satisfies it, the natural problem is to pick a set M such that γ satisfies restrict(T ,M) and the number of leaves in M is minimized (considering pairing to be the most expensive operation). This is easily accomplished with a straightforward recursive algorithm that makes a single traversal of the tree.

Direct computation of DecryptNode

Further improvements may be gained by abandoning the DecryptNode function and making more direct computations. Intuitively, we imagine flattening out the tree of recursive calls to DecryptNode, then combining the exponentiations into one per (used) leaf node.

VII. CONCLUSION AND FUTURE WORK

Although Cloud storage has many advantages, there are still many problems concerning security that need to be resolved. If we can completely eliminate or master this weakness of security, the future is going to be Cloud storage solutions for large as well as small companies. In this paper, we have suggested a solution that allows storage of data in public cloud. Data security is provided by implementing our algorithm. Only the authorized user can access the data. Even if an intruder (unauthorized user) gets the data accidentally or intentionally, he can’t decrypt it and needs two keys from two different parties [12]. We have done our survey upon extensive derivatives of ABE scheme. In future, we use user behaviour pattern as attributes for encryption and decryption to promote ideal security techniques.

VIII. REFERENCES

[1] HumeraAqeel, Syed Taqi Ali, "Directly revocable Attribute Based Encryption scheme under Ciphertext-policy", IEEE

International Conference on Computer, Communication and Electronics (Comptelix), August 2017, ISBN: 978-1-5090-4708-6.

[2] Ilya A.Sukhodolskiy, Sergey V. Zapechnikov, "An access control model for cloud storage using attribute encryption", IEEE Conference of Russian Youth Researchers in EIConRus, February 2017, ISBN: 978-1-5090-4865-6

[3] Qiang Wang; Li Peng; Hu Xiong; Jianfei Sun; Zhiguang Qin. "Ciphertext-Policy Attribute-based Encryption with Delegated Equality Cloud Computing", IEEE Access, Volume:PP ,Year: 2017, ISBN: 978-3-319-40252-9.

[4] Nikhil Chaudhari, Mohit Saini, Ashwin Kumar, G.Priya, "A Review on Attribute Based Encryption", IEEE 8th International Conference on Computational Intelligence and Communication Networks (CICN), 2016, ISBN: 978-1-5090-1144-5

[5] H. Abdul Gaffar,G. S. Tamizharasi; B. Balamurugan. "Privacy preserving ciphertext policy attribute based encryption scheme with efficient and constant ciphertextsize", International Conference on Inventive Computation Technologies (ICICT), Year: 2016, Volume:3, Pages: 1-5, ISBN: 978-1-5090-1285-5.

[6] R. Xu and B. Lang. "A CP-ABE scheme with hidden policy and its application in cloud computing", International Journal of Cloud Computing, Page:279–298, Year: 2015, ISBN: 978-1-4799-3261-0.

[7] U. C. Yadav. "Ciphertext-policy attribute-based encryption with hiding access structure", IEEE International Advance Computing Conference (IACC), pages 6–10, Year: 2015, ISBN: 978-1-4799-8046-8.

[8] Jinguang Han; Willy Susilo; Yi Mu; Jianying Zhou; Man Ho Allen Au. "Improving Privacy and Security in Decentralized Ciphertext-Policy Attribute-Based Encryption", IEEE Transactions on Information Forensics and Security, Year: 2015, Volume: 10, Issue:3, Pages 665-678, ISBN: 978-1-4799-2456-2.

[9] Bobba R., Khurana H., Prabhakaran M. (2009) "Attribute-Sets: A Practically Motivated Enhancement to Attribute-Based Encryption", In: Backes M., Ning P. (eds) Computer Security - ESORICS 2009. Lecture Notes in Computer Science, Volume: 5789. Springer, Berlin, Heidelberg.

[10] J. Bethencournt, A. Sahai, and B. Waters. "Ciphertext-policy attribute-based encryption", In Proceedings of the 2007, IEEE Symposium on Security and Privacy, SP '07, pages 321–334, Year: 2007, ISBN: 0-7695-2848-1.

[11] V. Goyal, O. Pandey, A. Sahai, and B. Waters. “Attribute-Based Encryption for Fine-grained Access Control of Encrypted Data”, In Proc. of CCS’06, Alexandria, Virginia, USA, 2006, ISBN: 978-1-4799-1846-1.