Hierarchical Attribute-based Access Control Solutions for Access Control in Cloud Computing: A Survey

Gunjan Batra
PhD IT, Rutgers Business School

Abstract – *Cloud computing, also known as on-demand computing is one of the biggest paradigms for the IT industry in the coming years. It enables the users to enjoy a reduced cost of ownership of resources along with the benefit of scalability services on demand. Especially for small organizations who cannot afford their own resources, cloud-based services are help achieve increase in productivity with high savings. At the same time, the Cloud Service provider needs to be cautious of the security and privacy of the data and maintain the trust of users. To achieve this, we require a well-defined, flexible, scalable, fine-grained and high performance access control mechanism that enables full delegation. Several schemes using Attribute based encryption (ABE) scheme have been proposed for secure access control of data in cloud computing. We survey the various approaches for Hierarchical Attribute based access control for Access Control in a cloud environment.*

1. Introduction

Cloud technology has revolutionized the way data is handled by the corporates. It has centralized the position of storage and computing resources and has reduced data-management costs significantly. A number of commercial cloud computing systems have been built at different levels, e.g., IBM's Blue Cloud [10], Amazon's EC2 [8] and Amazon's S3 [9], and are IaaS systems, while Yahoo Pig , and Google App Engine are representative PaaS systems, and Google's Apps [12] and Salesforce's Customer Relation Management (CRM) System [13] are SaaS systems.

Having said that, it is imperative to adopt an efficient access control technique for data outsourced to cloud. Cipher-text Policy Based Encryption(CP-ABE) proposed by Bethencourt et al [24], emerged as one of the promising encryption techniques that allow encryption of data by specifying access control policy over attributes, such that only the users with a specified set of attributes can decrypt the data. However CP-ABE could not solve the purpose due to many reasons. Cloud users have very high performance expectations which is not possible to achieve with CPABE. Also, a full delegation mechanism, (that is, a delegation mechanism between attribute authorities (AAs), which independently make decisions on the structure and semantics of their attributes) in the generation of keys inside an enterprise is needed. Only some CP-ABE schemes are able to support delegation between users, which enables a user to generate attribute secret keys containing a subset of his own attribute secret keys for other users, however that is partial. Third, a scalable revocation mechanism is required in case of a large-scale industry. The existing CP-ABE schemes usually demand users to heavily depend on AAs and maintain a large amount of secret keys storage, which lacks flexibility and scalability.

Our research problem is to design a secure cloud storage service to which cloud users can share confidential data on cloud servers by simultaneously achieving fine-grained access control, high

performance, practicability, and scalability. We will be surveying the extensions of CP-ABE to achieve the desired goal.

2. Approaches Proposed :

1. G. Wang, Q. Liu, J. Wu "Hierarchical Attribute-Based Encryption for fine Grained Access Control in Cloud Storage Services", ACM, 2010 [1]
- This research paper achieves efficient access control in cloud by using Hierarchical Identity Based encryption (HIBE) system and the Ciphertext-Policy Attribute Based Encryption (CP-ABE) system.
2. Z. Wan, J. Liu, R. Deng. "HASBE: A Hierarchical Attribute-Based Solution for Flexible and Scalable Access Control in Cloud Computing", IEEE Transactions on Information Forensics and Security, Vol.7 April 2012 [2]
- This research paper proposes hierarchical attribute-set-based encryption (HASBE) by extending ciphertext-policy attribute-set-based-encryption (ASBE) with a hierarchical structure of users.
3. X. Liu, Y. Xia, S. Jiang, F. Xia, Y. Wang. "Hierarchical Attribute –based Access Control with Authentication for Outsources Data in Cloud Computing", 2013 12th IEEE International Conference on Trust, Security and privacy in Computing and Communications [3]
- This research paper proposes a hierarchical attribute based access control scheme by extending ciphertext policy attribute based encryption (CP-ABE) with a hierarchical structure of multi authorities and exploiting attribute based signature (ABS). They also propose to decouple the task of policy management from security enforcement by using the extensible access control markup language (XACML) framework.

2.1. Hierarchical Attribute-Based Encryption (HABE) for fine Grained Access Control in Cloud Storage Services [1]

This was the first model using Hierarchical Attribute based Encryption (HABE) system for access control in cloud storage services. The authors propose the model by combining HIBE system and a CP-ABE system and hence provide fine-grained access control and full delegation.

Model – The HABE model consists of :

a. Root Master (RM) - This corresponds to Trusted Third Party (TTP). Its role is similar to that of root private key generator (PKG) of HIBE system. It is responsible for generation and distribution of system parameters and domain keys.
b. Domain Masters (DMs) - The top level DMs correspond to multiple enterprise users. Their role has both the tasks of the domain PKG in HIBE system and Attribute Authority (AA) in CP-ABE system. They are responsible for delegating keys to the DMs and distributing keys to the users. They enable the leftmost DM at second level to administer the users in a domain and other DMs administer an arbitrary number of disjoint attributes.
c. Users – They correspond to all the personnel in an enterprise.

All DMs and Attributes are marked with a unique identifier (ID) and the users are marked with both Unique ID and set of descriptive attributes. Further, using the concept of Gentry et

al [4], they have described the system such that an entity's secret key is obtained from the DM administering it, and an entity's public key is an ID tuple consisting of the public key of the DM administering it and its ID.
For example, the public key of DM_i with ID_i is in the form of $(PK_{i-1}; ID_i)$, the public key of user U with ID_u is in the form of $(PK_{\square}; ID_u)$, and the public key of attribute a with ID_a is in the form of $(PK_i; ID_a)$, where PK_{i-1}, $PK_{\square}$, and PK_i are assumed to be the public keys of the DMs that administer DM_i, U, and a, respectively.

System Construction – The authors have constructed the HABE scheme using the bilinear map [4]. To achieve better performance, they have:
1) Sacrificed the expressivity of access structure like Muller et al [5]
2) Used disjunctive normal form (DNF) policy assuming that all attributes in one conjunctive clause are administered by the same DM.

The HABE scheme implementation [1] consists of the following five algorithms: (The notations used in these algorithms have been derived from [5])

1. Setup(K) → (params,MK_0) : This algorithm generates the system parameters, params=(q;G_1;G_2; ^e; n; P_0;Q_0;H_1;H_2) that will be available to the public and master key, $MK_0 = (mk_0)$ that will be kept secret
   As described in [1], the RM first picks $mk_0 \in Z_q$, and then chooses groups G1 and G2 of order q, a bilinear map $\hat{e} : G_1 \times G_1 \rightarrow G_2$, two random oracles H_1: {0; 1}* → G_1 and H_2: $G_2 \rightarrow \{0; 1\}^n$ for some n, and a random generator $P_0 \in G_1$. Let $Q_0 = mk_0P_0 \in G_1$.

2. CreateDM (params,MK_i, PK_{i+1}) → (MK_{i+1}) : This algorithm generates the Master Key for DM_{i+1} with PK_{i+1}.
   As described in [1], for this, the RM or DM_i first picks a random element $mk_{i+1} \in Z_q$, and then computes $SK_{i+1} = SK_i + mk_iP_{i+1}$ where $P_{i+1} = H_1(PK_{i+1}) \in G_1$, and $Q_{i+1} = mk_{i+1}P_0 \in G_1$, finally sets $MK_{i+1} = (mk_{i+1};SK_{i+1}; Q\text{-tuple}_{i+1})$ where $Q\text{-tuple}_{i+1} = (Q\text{-tuple}_i; Q_{i+1})$, and gives the random oracle H_A : {0; 1}→ Z_q that is chosen by the RM and shared in a domain. They assume that SK_0 is an identity element of G_1, and $Q\text{-tuple}_0 = (Q_0)$.

3. CreateUser(params,MK_i, PK_u, PK_a) → ($SK_{i,u}$,$SK_{i,u,a}$) : This algorithm generates a secret key for user U with PK_u on attribute a with PK_a.
   As described in [1], DM_i first checks whether U is eligible for a, and a is administered by itself. If so, it first sets $mk_u = H_A(PK_u) \in Z_q$, $SK_{i;u} = mk_imk_uP_0 \in G1$, and $SK_{i;u;a} = SK_i + mk_imk_uP_a \in G_1$, where $P_a = H_1(PK_a) \in G_1$, and then gives $Q\text{-tuple}_i$. Otherwise, it outputs "NULL".

4. Encrypt(params,A, $\{PKa_{ij} | 1 \leq i \leq N; 1 \leq j \leq n_i\}$; f) → (CT) : As described in [1], this algorithm computes the Cipher Text given a DNF accesss control Policy A
5. Decrypt(params,CT,$SK_{it_i,u,}$, $\left\{SK_{it_i,u,a_{ij}} \middle| 1 \leq j \leq n_i\right\}$, $Q - Tuple_{i(t_i-1)}$ → (f)
   As described in [1],The user U whose attribute satisfy the CC_i is able to decrypt.

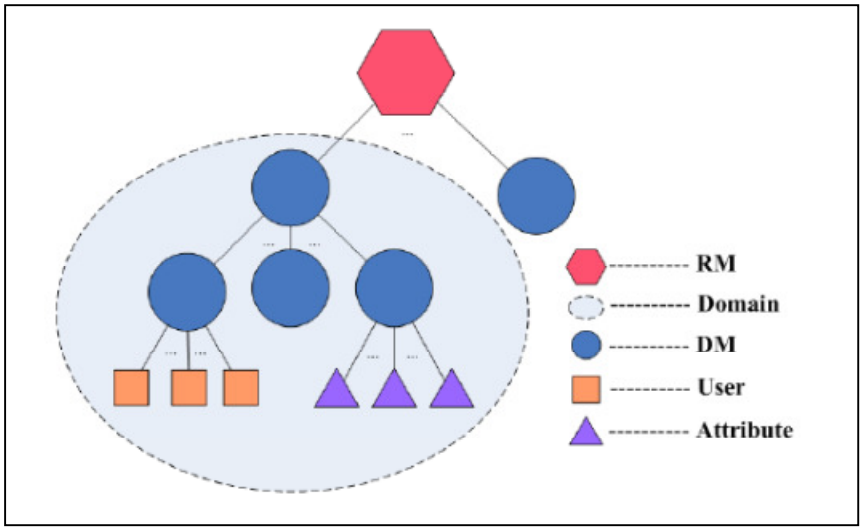

Fig1: Three Level HABE Model [1]

The authors have further applied proxy re-encryption and lazy re-encryption also to the scheme. Overall, they were able to construct a scheme with high performance, fine grained access control, scalability, and full delegation. However there is room for a more expressive in the scheme that can incorporate security and better performance.

2.2. HASBE : Hierarchical Attribute based solution for Flexible and Scalable Access Control in Cloud Computing [2]

HASBE is an extension of ASBE algorithm with a hierarchical structure. HASBE is based on ASBE by Bobba et al [25] which is implemented by the following algorithms (these are an extension of CPABE to support recursive key structure)

1. Setup (d): This algorithm takes as input d, the depth of key structure and outputs a public key PK and master secret key MK.
2. KeyGen (MK,u,A): This algorithm takes as input the master secret key MK, the identity of user u , and a key structure A. It outputs a secret key for user SK_u.
3. Encrypt(PK, M, T): This algorithm takes as input the public key PK, a message M, and an access tree T. It outputs a ciphertext CT.
4. Decrypt(CT, SK_u): Take as input a ciphertext CT and a secret key for user SK_u. It outputs a message m.

If the key structure A associated with the secret key SK_u satisfies the access tree T, associated with the ciphertext CT, then m is the original correct message M. Otherwise,m is Null.
is null.
ASBE missed the delegation algorithm, which is used in the proposed solution to construct the hierarchical structure

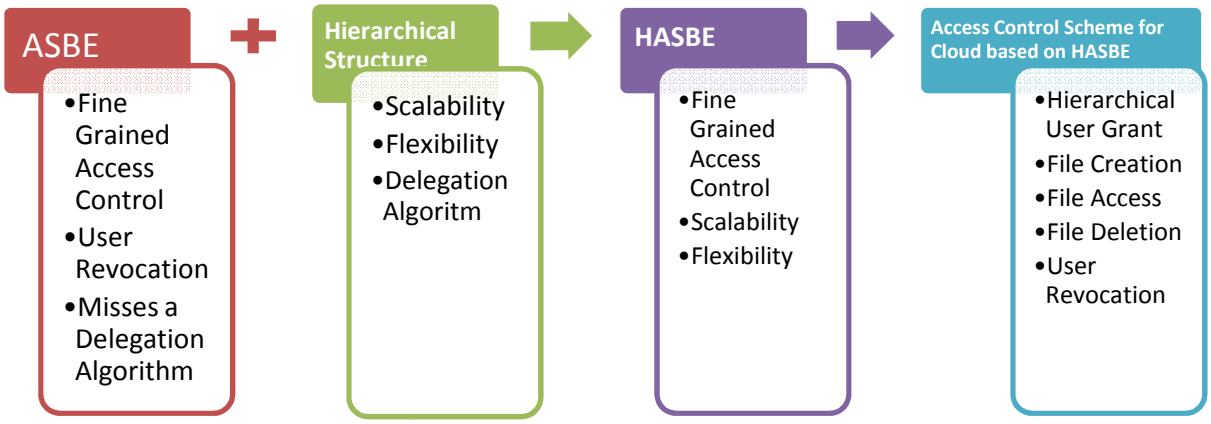

Fig 2: HASBE Model Development

System Model

The system has been made in a hierarchical structure and has five parties:

1. CSP (Cloud Service Provider) – Manages the Cloud services
2. Data Owners(Employes) – Encrypt their data files and store them in the cloud to share with data consumers
3. Data Consumers(Employees) – Download encrypted files and decrypt them to get access to the data
4. Domain Authorities(Federated Enterprise/its affiliated company) – Managed by its parent domain authority or trusted authority, manages the domain authorities at the next level and or data owners/ consumers in its domain
5. Trusted Authority(Root Authority) - Manages top level domain authorities

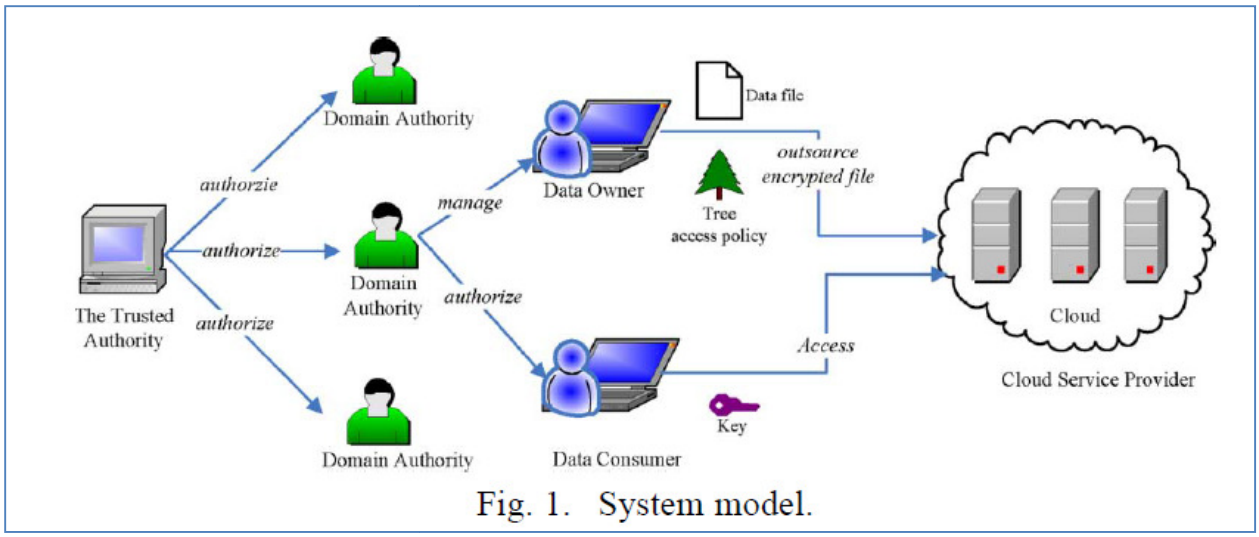

Fig 3: HASBE Model [2]

Assumptions

1. CSP is untrusted – It may collude with other malicious users to get the file contents
2. Each party has a public key and private key(kept secret)
3. Trusted authority is root of trust
4. Domain authority is trusted by subordinates
5. Domain authority tries to get keys of users outside its domain
6. Users try to access files within/outside their scope of access
7. Communication channel between all parties are secure(using security protocols eg SSL)

System Construction

In HASBE, the access structure for cipher text is specified by the data encryptor while encrypting the data. This is referred to as cipher text policy.
Only those users can decrypt the cipher text who have the decryption keys whose associated attributes specified in key structure satisfy the access structure.

1. Key Structure:

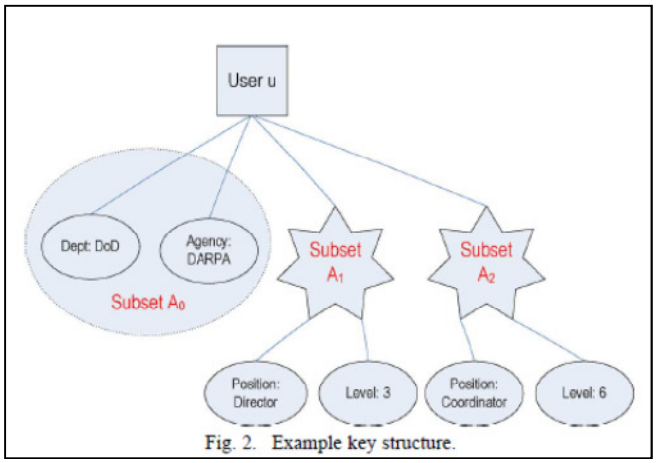

Figure 4: Example Key Structure [2]

The authors have used a recursive set based key structure as in [25]. Here each element of the set is either a set or an element corresponding to an attribute.
Depth of the key structure = level of recursions in the recursive set (analogous to depth of a tree)
In example shown in Fig. the key structure is {Dept:DoD, Agency:DARPA, Position:Director,Level:3}, {Position: Coordinator, Level : 6} is a key structure of Depth 2 and represents attributes of a person who is both a direcot of level 3 for a unit and a coordinator of level 6 for another unit in the Defense Advanced Research Projects Agency(DARPA) of the Department of Defense(DoD).
As mentioned in [2] , if there are m sets at depth 2 then a unique index where is assigned to each set. The set at depth 1 is referred to as set 0. Using this convention, a key structure of depth 2 can be represented as $A = \{A_0, A_1, \ldots, A_m\}$, where A_0 is the set at depth 1 while A_i is the ith set at depth 2, for $1 \leq i \leq m$.
For example in Fig., the key structure corresponds to {Dept:DoD, Agency:DARPA} corresponds to A_0, { Position:Director,Level:3} and { Position: Coordinator, Level : 6}correspond to A_1 and

A_2, respectively. Individual attributes inherit the label of the set they are contained in and are uniquely defined by the combination of their name and their inherited label.

2. Access Structure:

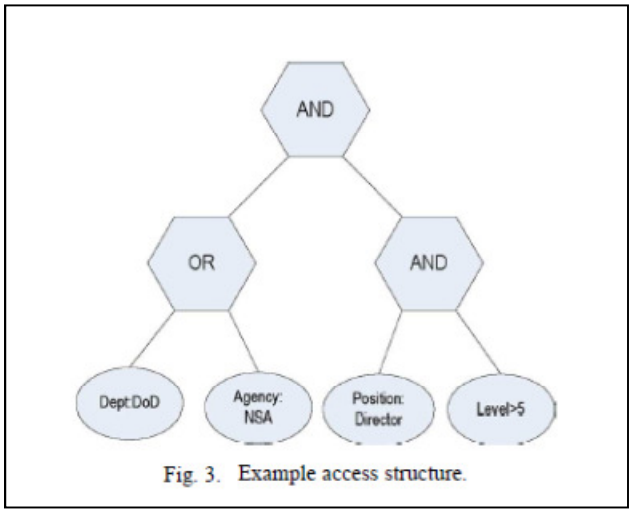

Fig. 3. Example access structure.

Figure 5: Example access Structure [2]

Same as [25]. It is a tree access structure having :

1. Leaf nodes that are attributes
2. Non Leaf nodes that are threshold gates, defined by its children and threshold value
   Num_x = Number of Children ; K_x= threshold value of node x
   Access structure[2] in Fig. demands that only a director in DoD or NSA of level larger than 5 can access the data files protected by the access policy; threshold values of AND and OR are 2 and 1 respectively.

As defined in [2],assuming T_x as the access structure rooted at node x and T be the access structure rooted at the root node R. Without loss of generality, considering key structure of depth 2, A = $\{A_0,A_1,\ldots,A_m\}$, where A_i $(0 \leq i \leq m)$ is the i th attribute set and i is the label.They that A satisfies T if and only if a function T(A) returns a nonempty set S of labels. The function T(A) is computed recursively.

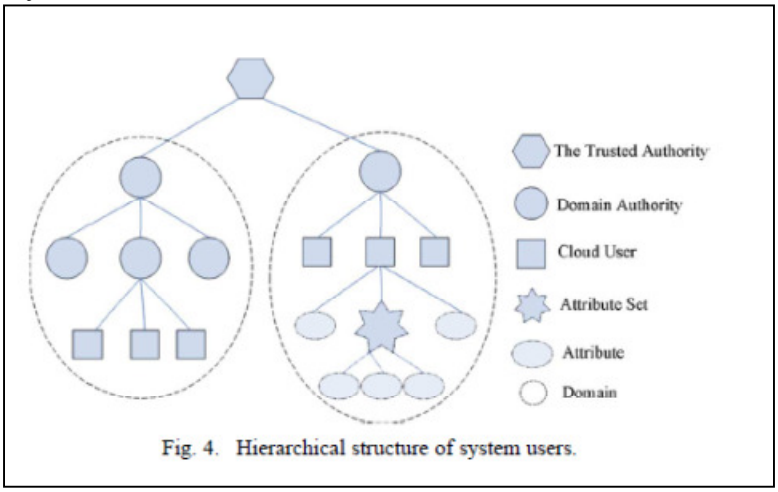

Fig. 4. Hierarchical structure of system users.

Figure 6: Hierarchical Structure of System Users [2]

HASBE Algorithms

Main operations of HASBE system:

1. System Setup: The trusted authority calls the algorithm to create system public parameters PK and master key $MK_{0\ while}$ taking the depth of the system d=2 as the input. PK will be made public to other parties and MK_0 will be kept secret.
   HASBE scheme can be extended to any depth.
2. Top-level Domain Authority Grant: Whenever a new top-level domain authority, i.e.,DA_i , wants to join the system, the trusted authority verifies if it is a valid domain authority and if it is, the trusted authority calls CreateDA to generate the master key for DA_i . Now, DA can authorize the next level domain authorities or users in its domain.

As mentioned in [2] , a domain authority is associated with a unique ID and a recursive attribute set $A = \{A_0,A_1,A_2,\ldots A_m\}$ where $A_i = \{a_{i,1}, a_{i,2}, \ldots a_{i,n}\}$ with $a_{i,j}$ being the jth attribute in A_i and n_i being the number of attributes in A_i.

3. CreateDA (PK,MK_0, A) : This algorithm creates the master key, MK_i for top-level DA_i . It also selects a unique number $r^{\{u\}}$ for the domain authority, which is also for the set A_0, and selects m random numbers $r^{\{u\}}_i \in Z_p$ one for each set, $a_i \in A$. It also computes $r^{\{u\}}_{i,j}$ for each $a_{i,j}$
4. New Domain Authority/User Grant: When a new user u or a new subordinate domain authority DA_{i+1} , wants to join the system, the administrating domain authority, DA_i, will first verify the validity of the new entity.
   If true, DA assigns the new entity a key structure $\tilde{A}$ corresponding to its role and a unique ID.$\tilde{A}$ is a subset of A, and A is the key structure of DA_i

For a new user u, DA_i calls CreateUser(MK_i, u, $\tilde{A}$) to generate the secret key for this user.
For a new domain authority, DA_{i+1}, DA_i calls CreateUser(MK_i, DA_{i+1}, $\tilde{A}$) to generate the master key for DA . Then DA can authorize the other lower level DAs or users in its domain.

5. CreateUser(MK_i, DA_{i+1}, $\tilde{A}$) : This algorithm inputs the Master key of DA_i i.e. key for access structure A, and new key structure $\tilde{A}$. It computes the new secret key for access structure $\tilde{A}$, SK_u or MK_{i+1}
6. New File Creation: In this algorithm, the data owner encrypts the file using symmetric key DEK as in [27], and then encrypts DEK using HASBE.
   Before uploading to the cloud, a data file is processed by the data owner as follows:
   - Assign a unique ID for this data file.
   - Encrypt the data file using randomly chosen DEK
   - Define an tree access structure, T for the file and encrypt DEK with T using algorithm Encrypt.

The format of the encrypted data file on the cloud is shown in Fig. 7

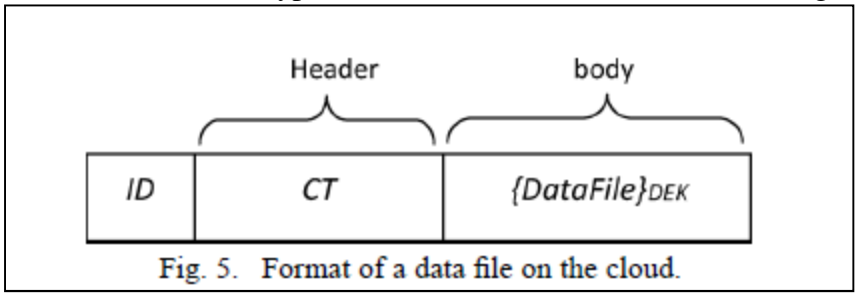

Fig. 5. Format of a data file on the cloud.

Figure 7: Format of a data file on the cloud [2]

7. File Access - When a user requests for file stored on the cloud, the cloud sends the corresponding CTs to the user. The user calls Decrypt (CT,SK_u) to get DEK and decrypt data files
8. Decrypt(CT, SK_u) This algorithm takes as input ciphertext CT and user u's key structure. The algorithm calls $\tau(A)$ to verify whether the key structure A in S_{Ku} satisfies the tree access structure τ associated with CT. This algorithm runs a subroutine DecryptNode and outputs the message M
9. File Deletion – This algorithm is executed at the request of the Data Owner of the after the cloud verifies his/her ownership.
10. User Revocation: HASBE adds an attribute expiration_time to a users's key which indicates the time till which the key will be considered valid. The policy associated with data files includes a check on the expiration_time attribute.
    Key Update - In this algorithm, the DA just maintains some state information of user's keys and assigns new value for expiration time to a user's existing key when updating it.
    Data Re-encryption – When re-encrypting the data files, the data owner just needs two exponentiations for ciphertext components associated with Expiration time and attribute.

Computational Complexity Analysis

The performance of the scheme is analyzed by computation overhead put on each operation. This is described in the table below:

Operation	**Complexity**
System Setup	O (1)
Top Level Domain Authority Grant	O (2N+M)
New User/ Domain Authority Grant	O (2N+M)
New File Creation	O (2|Y| + |X|)
User Revocation	O (1)
File Deletion	O (1)

Table 1 : Computational Complexity [2]

N = Number of attributes in the set of the new user or domain authority
M = Number of sets in A
Y = Leaf Nodes of T
X = Translating Nodes of T

2.3. Hierarchical Attribute –based Access Control with Authentication for Outsources Data in Cloud Computing

In this solution, authors utilize cipher text-policy attribute-based encryption (CP-ABE) with a hierarchical structure of multi-authorities and attribute-based signature (ABS) as authentication mechanism and propose a hierarchical attribute based access control scheme.

In addition, extensible access control markup language(XACML) has been used as a policy descriptive language. The access control policy is at two levels. (1) Coarse-grained access control policy that allows users access to data and (2) Fine-grained privilege management that provides privileges to corresponding users.

System Model

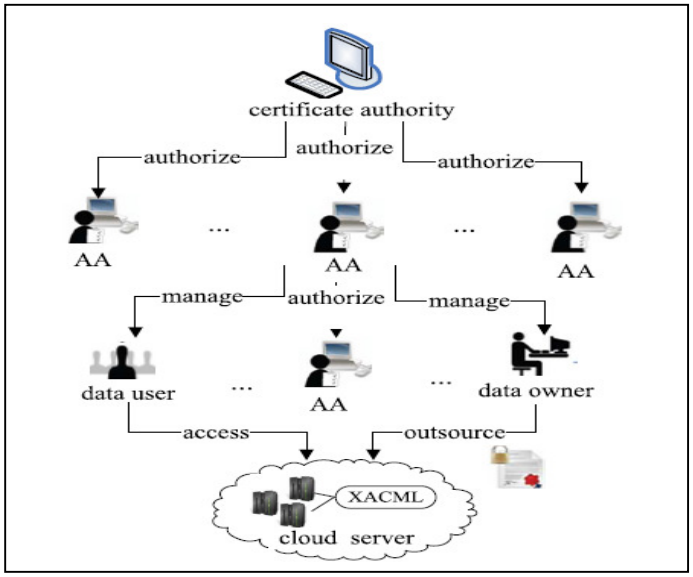

Figure 8 : System Model [3]

The cloud storage system has a hierarchical structure made of multi-authorities. It has five parties:

1. Global certificate authority (CA) – It is responsible for managing top-level attribute authorities. It sets up the system and generates the global secret key and global public key.
2. Attribute authorities (AAs) - Every top-level AA corresponds to a top-level organization, and each lower-level attribute authority corresponds to a lower-level organization.
3. Cloud servers (servers) - The cloud service provider (CSP) manages a cloud to provide data storage service to data owners and data access service to data users.
4. Data owners (owners) - Data owners define access control policies for the data, encrypt data under the policies and generate a signature related to the data before hosting them in the cloud. To access data files , data users download data files from cloud and then decrypt. A user can decrypt the ciphertext only if his/her attributes satisfy the access policy defined in the ciphertext.
5. Data users (users) – To update the data, users need to resign it. A user can update the data in the cloud if his/her signature satisfies the signing policy defined by the data owner.

Algorithms - Notations used in the scheme are mentioned in the table below:

Notation	Description
$\mathrm{PK}_e, \mathrm{PK}_s$	system public key for encryption and signature
$\mathrm{MK}_0, \mathrm{MK}_s$	master key for encryption and signature
$\mathrm{SK}_{u,e}, \mathrm{SK}_{u,s}$	encryption key and signing key for user u
$\mathcal{T}_{sig}, \mathcal{T}_{enc}$	key structure for encryption and signature
MK_i	master key of attribute authorities for encrypion
Y	set of leaf nodes in $\mathcal{T}_{enc}$
σ	the data owner's signature on message

Table 2 : Notations [3]

1. Setup : This algorithm comprises of the algorithms setting up the certification authority (CASetup), top level attribute authority grant and top-level attribute authorities setup (AASetup).
   - The CASetup (d) takes the d=number of top-level attribute authorities as input and outputs two public and master key pairs (PK_e,MK_0) and (PK_s,MK_s).
   - Top Level Attribute Authority Grant – An attribute authority is associated with a unique ID aa and a recursive attribute set A={A_0, A_1,….,A_m}, where A_i= {$a_{i,1}$, $a_{i,2}$, …..,$a_{i,ni}$} with $a_{i,j}$ being the jth attribute in Ai and n_i being the number of attributes in A_i. The certificate authority calls AASetup that generates the Master key for AA_i whenever a new authorized top-level authority wants to join
   - AASetup(PK_e,MK_0,A) takes the global public parameters PK_e for encryption and outputs the attribute authorities' master key MK_i.
2. Key Generation (MK_0, MK_s,$_u$, S): When a new user or subordinate authority AA_{i+1} wants to join the system, the top level authority A_i verifies its validity. Then it assigns an attribute set Ã C A. This algorithm takes as inputs the identity of user u, the master key MK_0, MK_s and a set of attributes S that describe the user. It outputs the user's encryption key $SK_{u,e}$ and signing key $SK_{u,s}$.
3. Data Outsourcing
   - Encrypt (PK_e,M,T_{enc}). This algorithm takes as input the public parameter PK_e, a message M, and an access structure T_{enc}. It outputs a ciphertext CT such that only a user who possesses a set of attributes that satisfies the access structure will be able to decrypt the message.
     After encryption the data owner signs the CT to provide integrity verification to all parties who want to access the files and for specifying write policy to authorized users who could update the file.
   - Sign (PK_s,$SK_{u,s}$,CT,T_{sig}). The owner signs the CT using ABS. this algorithm takes as input the public parameter PK_s and signing key $SK_{u,s}$, a ciphertext CT, and a structure T_{sig}. It outputs a signature σ such that only the user who possesses a set of attributes that satisfies the structure T_{sig} will be able to verify the signature.
   -

The encrypted file with signature is stored in the cloud in the following format shown in Fig

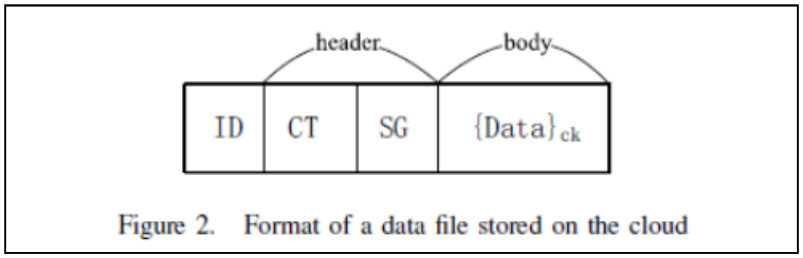

Figure 9: Format of Data File Stored on the Cloud [3]

4. Data Accessing
   - Decrypt ($SK_{u,e}$,CT). This algorithm takes as input a ciphertext CT which contains an access structure T_{enc} and encryption key $SK_{u,e}$ for user u. It outputs a message m such that only the attributes associate with the encryption key $SK_{u,e}$ satisfy the access structure T_{enc} associate with the ciphertext CT.

- Verify (PK_s,CT,T_{sig},σ). This algorithm verifies the signature attached with the data. It takes as input public key for signature PK*s* and the owner's signature on ciphertext CT. It returns true such that only the attributes associate with the signature σ satisfy the structure T_{sig} associate with the ciphertext CT.

5. This indicates the time period until which the key is considered valid. Moreover, the associated policy includes a check on the expiration time attribute. This way, the users' key is updated without entire key regenerating and redistributing at the end of expiration time.
6. Policy Management – This is for management and matching of the user access policies in a systematic and a scalable manner. The data owner specifies the policies in XML format and XACML framework is used for policy enforcement. When the user needs to access the data, he/she provides his/her credentials and certain action on the system of resources such as read or write. This request is translated into a well-formed and standard XACML request, which is evaluated against list of policies. The end user receives and decrypts the cipher text of his/her authorized data.

Computational Complexity Analysis

The performance of the scheme is analyzed by computation overhead put on each operation. This is described in the table below:

Operation	**Complexity**
System Setup	O (2N)
Create a file	O (2|Y| + lt)
Read a file	O (2|Y|)
Update a file	O (2lt)

Table 3 : Computational Complexity [3]

N=Number of attributes in the set of new user or attribute authority
l×t = Size of the Access structure matrix

3. Comparison

On comparison of the three schemes we found out the results displayed in the table below:

Schemes	**Fine Grained Access Control**	**Type of Authority**	**Write/Read Access**	**Policy Management**
[1]	Yes	Hierarchical		No
[2]	Yes	Hierarchical	1-W-M-R	No
[3]	Yes	Hierarchical	M-W-M-R	Yes

Table 4 : Comparison of the three solutions

All three of them have fine grained access control and hierarchical authority structure
The solution in [2] produces a very heavy computation overhead on the data owner for key distribution and data management, so it does not scale well. The approach in [3] is scalable and also combines XACML framework for policy management that helps in security.

4. Conclusion

We surveyed papers that modeled, implemented and analysed the performance of various hierarchical attribute based access control schemes for cloud environment. In the first paper, we briefly described the HABE scheme, that's had achieved high performance, fine grained access control and scalability. In the next paper, we described HASBE scheme which not only has benefits of the previous scheme but also supports compound attributes due to flexible attribute set combinations. In the third paper, we described the scheme that used Attribute based Signature (ABS) along with CP-ABE and integrated policy management with an XML based framework (XACML).
All the three schemes came year after each other and added more features to improve the access control in cloud environment.

5. References

.

[1] G.Wang, Q. Liu, and J.Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in Proc. ACM Conf. Computer and Communications Security (ACM CCS), Chicago, IL, 2010

[2] Z. Wan, J. Liu, and R.H. Deng. Hasbe: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing. IEEE Transactions on Information Forensics and Security, 7(2):743–754, 2012.

[3] Xuejiao Liu, Yingjie Xia, Shasha Jiang, Fubiao Xia and Yanbo Wang, "Hierarchical attribute-based access control with authentication for outsourced data in cloud computing", Proceedings of the 12th IEEE TrustCom, pp. 477-484

[4] C. Gentry and A. Silverberg. Hierarchical ID-Based Cryptography. In Proceedings of ASIACRYPT 2002, pages 548-566.

[5] S. Muller, S. Katzenbeisser, and C. Eckert. Distributed Attribute-Based Encryption. In Proceedings of ICISC 2008, pages 20-36

[6] S. Yu, C. Wang, K. Ren, and W. Lou. Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing. In Proceedings of IEEE INFOCOM 2010, pages 534-542.

[7] R. Buyya, C. ShinYeo, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generation Comput. Syst., vol. 25, pp. 599–616, 2009

[8] Amazon Elastic Compute Cloud (Amazon EC2) [Online]. Available: http://aws.amazon.com/ec2/

[9] Amazon Web Services (AWS) [Online]. Available: https://s3.amazonaws.

[10] R. Martin, "IBM brings cloud computing to earth with massive new data centers," *InformationWeek* Aug. 2008 [Online]. Available: http:// www.informationweek.com/news/hardware/data_centers/209901523

[11] Google App Engine [Online]. Available: http://code.google.com/appengine/

[12] K. Barlow and J. Lane, "Like technology from an advanced alien culture:Google apps for education at ASU," in *Proc. ACM SIGUCCS User Services Conf.*, Orlando, FL, 2007.

[13] B. Barbara, "Salesforce.com: Raising the level of networking," *Inf. Today*, vol. 27, pp. 45–45, 2010.

[14] J. Bell, Hosting Enterprise Data in the Cloud—Part 9: Investment Value Zetta, Tech. Rep., 2010.
[15] A. Ross, "Technical perspective: A chilly sense of security," *Commun. ACM*, vol. 52, pp. 90–90, 2009.
[16] D. E. Bell and L. J. LaPadula, Secure Computer Systems: Unified Exposition and Multics Interpretation The MITRE Corporation, Tech. Rep., 1976.
[17] K. J. Biba, Integrity Considerations for Secure Computer Sytems The MITRE Corporation, Tech. Rep., 1977.
[18] H. Harney, A. Colgrove, and P. D. McDaniel, "Principles of policy in secure groups," in *Proc. NDSS*, San Diego, CA, 2001.
[19] P. D. McDaniel and A. Prakash, "Methods and limitations of security policy reconciliation," in *Proc. IEEE Symp. Security and Privacy*, Berkeley, CA, 2002.
[20] T. Yu and M. Winslett, "A unified scheme for resource protection in automated trust negotiation," in *Proc. IEEE Symp. Security and Privacy*, Berkeley, CA, 2003.
[21] J. Li, N. Li, and W. H. Winsborough, "Automated trust negotiation using cryptographic credentials," in *Proc. ACM Conf. Computer and Communications Security (CCS)*, Alexandria, VA, 2005.
[22] V. Goyal, O. Pandey, A. Sahai, and B.Waters, "Attibute-based encryption for fine-grained access control of encrypted data," in *Proc. ACM Conf. Computer and Communications Security (ACM CCS)*, Alexandria, VA, 2006.
[23] S. Yu, C. Wang, K. Ren, and W. Lou, "Achiving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. IEEE INFOCOM 2010*, 2010, pp. 534–542.
[24] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute based encryption," in *Proc. IEEE Symp. Security and Privacy*, Oakland, CA, 2007.
[25] R. Bobba, H. Khurana, and M. Prabhakaran, "Attribute-sets: A practically motivated enhancement to attribute-based encryption," in *Proc. ESORICS*, Saint Malo, France, 2009.
[26] A. Sahai and B. Waters, "Fuzzy identity based encryption," in *Proc. Acvances in Cryptology—Eurocrypt*, 2005, vol. 3494, LNCS, pp.57–473.
[27] Sushmita Ruj, Milos Stojmenovic, and Amiya Nayak. Privacy preserving access control with authentication for securing data in clouds. In *The 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 556–563, 2012.
[28] V. Bozovic, D. Socek, R. Steinwandt, and V.I. Vill´anyi. Multi-authority attribute-based encryption with honest-butcurious central authority. *International Journal of Computer Mathematics*, 89(3):268–283, 2012.
[29] M. Chase. Multi-authority attribute based encryption. *Theory of Cryptography*, pages 515–534, 2007.
[30] Oasis extensible access control markup language (xacml).https://www.oasis-open.org/committees/xacml/.
[31] 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*, pages 89–98, 2006.
[32] J. Hur and D.K. Noh. Attribute-based access control with efficient revocation in data outsourcing systems. *IEEE Transactions on Parallel and Distributed Systems*, 22(7):1214–1221, 2011
[33] H. Maji, M. Prabhakaran, and M. Rosulek. Attribute-based signatures. *Proceedings of the 11th International Conference on Topics in Cryptology*, pages 376–392, 2011.

[34] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology*, pages 41–62, 2001.

[35] W. Wang, Z. Li, R. Owens, and B. Bhargava. Secure and efficient access to outsourced data. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, pages 55–66, 2009.

[36] Yingjie Xia, Li Kuang, and Mingzhe Zhu. A hierarchical access control scheme in cloud using hhecc. *Information Technology Journal*, 9(8):1598–1606, 2010.

[37] S. Yu, C. Wang, K. Ren, and W. Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *Proceedings of IEEE INFOCOM*, pages 1–9, 2010.

[38] S. Yu, C. Wang, K. Ren, and W. Lou. Attribute based data sharing with attribute revocation. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 261–270, 2010.