American Journal of Information Systems, 2016, Vol. 4, No. 2, 17-31
Available online at http://pubs.sciepub.com/ajis/4/2/1
© Science and Education Publishing
DOI:10.12691/ajis-4-2-1

ASP: Advanced Security Protocol for Security and Privacy in Cloud Computing

Shyam Nandan Kumar[1,*], Amit Vajpayee[2]

[1]M.Tech-Computer Science and Engineering, Lakshmi Narain College of Technology-Indore (RGPV, Bhopal), MP, India
[2]Department of Computer Science and Engineering, Lakshmi Narain College of Technology-Indore (RGPV, Bhopal), MP, India
*Corresponding author: shyamnandan.mec@gmail.com

Abstract Security concern has become the biggest obstacle to adoption of cloud because all information and data are completely under the control of cloud service providers. To provide optimal services on cloud, this paper introduces a new distributed and scalable data sharing scheme for data owners in clouds that supports anonymous authentication. Proposed ASP (Advanced Security Protocol) protocol is a cryptographic access control protocol based on key-updating scheme referred to as Advanced Key Update (AKU). The main advantage of the AKU scheme its support for efficient delegation and revocation of privileges in hierarchies without requiring complex cryptographic data structures. Proposed ASP protocol also includes a new digital signature scheme that enables cloud providers to ensure that requests are submitted by authorized end-users, without learning their identities. User Revocation facility is also supported by proposed ASP. In this paper various existing approaches and issues related to data encryption and message authentications are also discussed. At last, experiment results are analyzed and performances are evaluated. The main aim of the paper is to provide more visibility and control to the end-users and close the gap between capabilities of existing solutions and new requirements of cloud-based systems.

Keywords: *cloud computing, data sharing, decryption, encryption, concurrent access, distributed system, web, message signing and verification, data confidentiality, message authentication, cloud security*

Cite This Article: Shyam Nandan Kumar, and Amit Vajpayee, "ASP: Advanced Security Protocol for Security and Privacy in Cloud Computing." *American Journal of Information Systems*, vol. 4, no. 2 (2016): 17-31. doi: 10.12691/ajis-4-2-1.

1. Introduction

Cloud computing is emerging from recent advances in technologies such as hardware virtualization, Web services, distributed computing, utility computing and system automation. It is continuously evolving and showing consistent growth in the field of computing [1]. With virtualization, one or more physical servers can be configured and partitioned into multiple independent "virtual" servers, all functioning independently and appearing to the user to be a single physical device. Such virtual servers are in essence disassociated from their physical server, and with this added flexibility, they can be moved around and scaled up or down on the fly without affecting the end user. The difference with cloud computing is that the computing process may run on one or many connected computers at the same time, utilizing the concept of virtualization [1]. With multiple users from different organizations contributing to data in the Cloud, the time and cost will be much less compared to having to manually exchange data and hence creating a clutter of redundant and possibly out-of-date documents. With social networking services such as Facebook, the benefits of sharing data are numerous such as the ability to share photos, videos, information and events. Google Docs provides data sharing capabilities as groups of students or teams working on a project can share documents and can collaborate with each other effectively. This allows higher productivity compared to previous methods of continually sending updated versions of a document to members of the group via email attachments. Also in modern healthcare environments, healthcare providers are willing to store and share electronic medical records via the Cloud and hence remove the geographical dependence between healthcare provider and patient. Due to this need of Cloud Mining [5] and security over web [4] is gaining popularity.

Cloud computing providers offer their services according to several fundamental models [6]: infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS) where IaaS is the most basic and each higher model abstracts from the details of the lower models.

Layered architecture of cloud computing requires different levels of security considerations. In this work we are mainly concerned with the problem of identity management and access control in application and service level. In SaaS, users are provided access to application software and databases. SaaS users have less control over security among the three fundamental delivery models in the cloud [1]. In the PaaS models, cloud providers deliver a computing platform, typically including operating system, programming language execution environment, database, and web server. Application developers can develop and run their software solutions on a cloud platform without the cost and complexity of buying and managing the underlying hardware and software layers. PaaS application security comprises two software layers: Security of the PaaS platform itself (i.e., runtime engine), and Security of customer applications deployed on a PaaS platform [7]. PaaS providers are responsible for securing the platform software stack that includes the runtime engine that runs the customer applications. PaaS model aims to protect data, which is especially important in case of storage as a service. In case of congestion, there is the problem of outage from a cloud environment. Thus the need for security against outage is important to ensure load balanced service. The data needs to be encrypted when hosted on a platform for security reasons. Cloud providers manage the infrastructure and platforms that run the applications [1]. IaaS refers to the sharing of hardware resources for executing services, typically using virtualization technology. Potentially, with IaaS approach, multiple users use available resources. Unlike PaaS and SaaS, IaaS customers are primarily responsible for securing the hosts provisioned in the cloud. Customers of IaaS have full access to the virtualized guest VMs that are hosted and isolated from each other by hypervisor technology. Hence customers are responsible for securing and ongoing security management of the guest virtual machine (VM) [1]. However, finding an efficient and secure way to share partial data in cloud storage is not trivial. In a shared-tenancy cloud computing environment, things become even worse. Data from different clients can be hosted on separate virtual machines (VMs) but reside on a single physical machine. Data in a target VM could be stolen by instantiating another VM co-resident with the target one [2].

The fundamental factor defining the success of any new computing technology is the level of security it provides [1]. The three basic requirements of security: confidentiality, integrity and availability are required to protect data throughout its lifecycle. Data must be protected during the various stages of creation, sharing, archiving, processing etc. However, situations become more complicated in case of a public cloud where we do not have any control over the service provider's security practices [8].

To enable data access control in the Cloud, it is imperative that only authorized users are able to get access to data stored in the Cloud. Various access control models are in use, including the most common Mandatory Access Control (MAC), Discretionary Access Control (DAC) and Role Based Access Control (RBAC). All these models are known as identity based access control models. In all these access control models, user (subjects) and resources (objects) are identified by unique names. Identification may be done directly or through roles assigned to the subjects. These access control methods are effective in unchangeable distributed system, where there are only a set of Users with a known set of services [1,2]. In DAC, information may be accessed by unauthorized users because there is no control on copies of objects. MAC deals with information flow and solves this problem by attaching security levels on both users and objects. All users are required to obtain certain clearance to access objects. Security labels propagate to derivative objects, including copies. However, the policies in DAC and MAC are fixed and there is no room for flexible access control. RBAC emerged due to increasing practitioner dissatisfaction with the then dominant DAC and MAC paradigms, inspiring academic research on RBAC. Since then RBAC has become the dominant form of access control in practice. In enterprise settings, we see the rise in demand for data outsourcing, which assists in the strategic management of corporate data [1,2,3]. It is also used as a core technology behind many online services for personal applications.

Cloud security is an evolving sub-domain of computer security, network security, and, more broadly, information security. It refers to a broad set of policies, technologies, and controls deployed to protect data, applications, and the associated infrastructure of cloud computing. Most Cloud service provider's provide basic key encryption schemes for protecting data or may leave it to the user to encrypt their own data. Both encryption and key management are very important to help secure applications and data stored in the Cloud [1,3]. The stored data must be protected against unauthorized access. Also, both the data and the access to data need to be protected from cloud storage service providers (e.g., cloud system administrators). In these scenarios, relying on password and other access control mechanisms is insufficient. Cryptographic encryption mechanisms [2,3,4] are typically employed. However, simply having encryption and decryption implemented in the cloud database systems is insufficient. In order to support both challenges, data should be encrypted first by users before it is outsourced to a remote cloud storage service and both data security and data access privacy should be protected such that cloud storage service providers have no abilities to decrypt the data, and when the user wants to search some parts of the whole data, the cloud storage system will provide the accessibility without knowing what the portion of the encrypted data returned to the user is about [1,2].

The Cloud however is susceptible to many privacy and security attacks. The biggest obstacle hindering the progress and the wide adoption of the Cloud is the privacy and security issues associated with it. Evidently, many privacy and security attacks occur from within the Cloud provider themselves as they usually have direct access to stored data and steal the data to sell to third parties in order to gain profit [1,2,3]. Care should be taken to ensure access control of the sensitive information. Performance of sharing and accessing applications should be improved.

The main aim of the paper includes:

- To provide more visibility and control to the end-users and close the gap between capabilities of existing solutions and new requirements of cloud-based systems.
- To introduce a new scalable and secure key-updating scheme for access hierarchies.
- To design and implement a scalable and privacy-preserving access control framework for existing untrusted cloud services. Proposed framework supports lazy revocation and access hierarchies.
- To present a signature scheme for Key-Policy Attribute-Based Encryption [15]. Using proposed signature scheme, users can prove that they own a key that its policy satisfies with a set of attributes, without revealing their identity or credentials.

The paper is organized as follows. Security issue with cloud model is given in Section 2. Literature Review is presented in Section 3. The mathematical background, Access policies and assumptions are detailed in Section 4. We present our privacy preserving access control scheme ASP in Section 5. Section 6 has the idea about Implementation and Operation of ASP protocol. The security is analyzed and computation complexity is discussed in Section 7. Conclusion and future work is provided in Section 8.

2. Security issue with Cloud Model

As cloud computing is achieving increased popularity, concerns are being voiced about the security issues introduced through adoption of this new model. The relative security of cloud computing services is a contentious issue that may be delaying its adoption [1]. Security issues have been categorized into sensitive data access, data segregation, privacy, bug exploitation, recovery, accountability, malicious insiders, management console security, account control, and multi-tenancy issues. Solutions to various cloud security issues vary, from cryptography, particularly public key infrastructure (PKI), to use of multiple cloud providers, standardization of APIs, and improving virtual machine support and legal support [1].

In a public cloud enabling a shared multi-tenant environment, as the number of users increase, security risks get more intensified and diverse. It is necessary to identify the attack surfaces which are prone to security attacks and mechanisms ensuring successful client-side and server-side protection [1,3,10]. Because of the multifarious security issues in a public cloud, adopting a private cloud solution is more secure with an option to move to a public cloud in future, if needed [1,13]. A few of the key security issues in a public cloud include:

- In case of a public cloud, the same infrastructure is shared between multiple tenants and the chances of data leakage between these tenants are very high. However, most of the service providers run a multitenant infrastructure. Proper investigations at the time of choosing the service provider must be done in order to avoid any such risk [1,8,46].
- The three basic requirements of security: confidentiality, integrity and availability are required to protect data throughout its lifecycle. Data must be protected during the various stages of creation, sharing, archiving, processing etc. However, situations become more complicated in case of a public cloud where we do not have any control over the service provider's security practices [1,8].

In a private cloud, customers have total control over the network. Private cloud provides the flexibility to the customer to implement any traditional network perimeter security practice. Although the security architecture is more reliable in a private cloud, yet there are issues/risks that need to be considered: A few of the key security issues in a public cloud include [1]:

- In a private cloud, users are facilitated with an option to be able to manage portions of the cloud, and access to the infrastructure is provided through a web interface or an HTTP end point. There are two ways of implementing a web-interface, either by writing a whole application stack or by using a standard applicative stack, to develop the web interface using common languages such as Java, PHP, and Python etc. As part of screening process, Eucalyptus web interface has been found to have a bug, allowing any user to perform internal port scanning or HTTP requests through the management node which he should not be allowed to do. In the nutshell, interfaces need to be properly developed and standard web application security techniques need to be deployed to protect the diverse HTTP requests being performed [1,47].
- Virtualization techniques are quite popular in private clouds. In such a scenario, risks to the hypervisor should be carefully analyzed. There have been instances when a guest operating system has been able to run processes on other guest VMs or host. In a virtual environment it may happen that virtual machines are able to communicate with all the VMs including the ones who they are not supposed to. To ensure that they only communicate with the ones which they are supposed to, proper authentication and encryption techniques such as IPsec [IP level Security] etc. should be implemented [1,48].

Private clouds are considered safer in comparison to public clouds; still they have multiple issues which if unattended may lead to major security loopholes. Hybrid cloud model is a combination of both public and private cloud and hence the security issues discussed with respect to both are applicable in case of hybrid cloud.

Various types of Attack on Cloud are increasing day by day. Some of the common attack can be consider as follows:

Cross Site Scripting (*XSS*) attacks: Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts (also commonly referred to as a malicious payload) into a legitimate website or web application. XSS is amongst the most rampant of web application vulnerabilities and occurs when a web application makes use of un-validated or un-encoded user input within the output it generates. In order for an XSS attack to take place the vulnerable website needs to directly include user input in its pages. An attacker can then insert a string that will be used within the web page and treated as code by the victim's browser [1,2,26].

XML Signature Wrapping Attacks: Using different kinds of XML signature wrapping attacks, one can completely take over the administrative rights of the Cloud user and create, delete, modify images as well as create instances [2].

Data Stealing Attacks: A term used to describe the stealing of a user account and password by any means such as through brute-force attacks or over-the-shoulder techniques. The privacy and confidentiality of user's data will be severely breached. A common mechanism to prevent such attacks is to include an extra value when authenticating. This value can be distributed to the right user by SMS and hence mitigate the likelihood of data confidentiality issues [2].

Flooding Attacks: A malicious user can send requests to the Cloud; he/she can then easily overload the server by creating bogus data requests to the Cloud. The attempt is to increase the workload of the Cloud servers by consuming lots of resources needlessly [2].

Passive Attacks: This type of attacks includes observation or monitoring of communication. A passive attack attempts to learn or make use of information from the system but does not affect system resources. The goal of the opponent is to obtain information that is being transmitted [3]. Types of passive attacks includes: *Traffic Analysis* and *Release of Message Contents.*

Cloud computing security issues include preserving confidentiality and privacy of data. Only encryption or authentication cannot give suitable security service. They having individual feature [1]. Confidentiality assures that private or confidential information is not made available or disclosed to unauthorized individuals over the clouds. A loss of confidentiality is the unauthorized disclosure of information. Message authentication assures that data received are exactly as sent (i.e., contain no modification, insertion, deletion, or replay). In many cases, there is a requirement that the authentication mechanism assures that purported identity of the sender is valid. It verifies the integrity of message [1].

To achieve confidentiality, integrity and authentication of data, there should be encryption and decryption along with message signature and verification. Data Confidentiality and Message Authentication together will give better security than single encryption or single authentication during data processing over the cloud. The data objects should never be updated by unauthorized clients and in order to achieve this limitation the system ensures that only correct and authorized client are able to perform the updates [1]. For optimal authentication, signing and verifying of message is need. Message authentication may also verify sequencing and timeliness.

3. Literature Review

When sensitive information is stored in cloud servers, which is out of user‘s control in most cases, risks would rise dramatically. Unauthorized users may also be able to intercept someone‘s data (e.g. server compromise).

Sahai and Waters proposed a new type of IBE – Fuzzy Identity-Based Encryption [14]. It is also known as Attribute-Based Encryption (ABE). In their work, an identity is viewed as a set of descriptive attributes. Different from the IBE, where the receiver could decrypt the message if and only if his identity is exactly the same as what specified by the sender, this fuzzy IBE enables the decryption if there are identity overlaps‘ exceeding a pre-set threshold between the one specified by sender and the one belongs to receiver. However, this kind of threshold-based scheme was limited for designing more general [1].

In Key-policy ABE or KP-ABE (Goyal et al. [15]), the sender has an access policy to encrypt data. Cipher-text is associated with a set of attributes, which partially represents the cipher-text‘s encryption policy. A writer whose attributes and keys have been revoked cannot write back stale information. The receiver receives attributes and secret keys from the attribute authority and is able to decrypt information if it has matching attributes. Unfortunately, with a drawback that the access policy is built into the secret key, the data owner in a KP-ABE scheme cannot decide the one who can decrypt the cipher text, and he can only choose a set of attributes to control the access of cipher texts. Besides, the access structure is a monotonic access structure which cannot express the negative attribute to exclude the participants with whom the data owner does not want to share data. Subsequently, Ostrovsky et al. [16] proposed a scheme with a non-monotonic access structure where the secret keys are labeled with a set of attributes including positive and negative attributes [1].

In 2007, using a monotonic access tree as access structure, Bethencourt et al. [17] proposed the first CP-ABE construction. Their scheme can support flexible access control policies like the KP-ABE [15] scheme, but the security proof is in the generic group model. Cheung and Newport [18] provided a provably secure CP-ABE scheme which is proved to be secure under the standard model and their scheme supports AND gate on positive and negative attributes as its access policy. In 2011, Waters [19] proposed a new methodology for realizing CP-ABE under concrete and non-interactive cryptographic assumptions in the standard model. He expressed access control by a linear secret sharing scheme (LSSS) matrix over the attributes in the system (previously used structures can be expressed succinctly in terms of an LSSS). In this most efficient scheme, the cipher text size and the encryption/decryption overheads increase linearly with the complexity of the access formula. As a result, his scheme achieves the same performance and functionality as Bethencourt et al.’s [17]. Finally, Lewko et al. [20] recently leveraged the encoding technique from Waters’s scheme [19] to propose an ABE scheme that achieves adaptive (nonselective) security. Their scheme is based on the Composite order groups, which results in some loss of practical efficiency when compared with Water’s. Emura et al. [21] improved the efficiency and achieved hidden policies [1].

Multi-authority ABE schemes [22,23] can be divided into two types. One needs a central authority (CA, for short) which is used to guarantee the proper decryption and can also decrypt all cipher texts, such as schemes [22,24], while the other does not need a CA, such as schemes [25,26]. Paper [55] proposes the threshold-based key generation approach (TKGA) for ciphertext-policy attribute-based encryption (CP-ABE). TKGA is a multi-authority approach which utilize the technologies of functional encryption and (n, k)-secret sharing. TKGA could efficiently impede collusion attacks because no single authority can directly generate secret keys.

In 2009, Attrapadung and Imai [27] presented a new ABE scheme called the Dual-Policy ABE. Basically, it is a conjunctively combined scheme of Goyal et al.’s KP-ABE scheme [15] and Waters’ CP-ABE scheme [19]. It allows simultaneously two access control mechanisms over encrypted data. One involves policies over objective attributes ascribed to data and the other involves policies over subjective attributes ascribed to user credentials. These two access control mechanisms can only allow either functionality above one at a time. What is more, the security proof is based on decisional bilinear Diffie-Hellman exponent (DBDHE) assumption [1].

To achieve the hierarchical access control and improve update efficiency, the revocable attribute based encryption scheme with hierarchical revocation based on multi-linear maps is proposed in [57]. Hierarchical attribute-based encryption scheme (HABE) [28] by combining a hierarchical identity-based encryption (HIBE) system and a cipher text-policy attribute-based encryption (CP-ABE) system, so as to provide not only fine-grained access control, but also full delegation and high performance. It supports a scalable revocation scheme by applying proxy re-encryption (PRE) and lazy re-encryption (LRE) to the HABE scheme, so as to efficiently revoke access rights from users. Based on the key-policy attribute-based encryption (KP-ABE), combined with the idea of hierarchical ID-Based encryption (HIBE), a hierarchical authority key-policy attribute-based encryption (HA-KP-ABE) scheme is presented in [58]. It uses hierarchical multi-authority to distribute private keys to users. Here private keys are computed for users according to random polynomials.

To make data sharing more efficient, proxy re-encryption (PRE) is proposed. Introduced by Mambo and Okamoto [29] and first defined by Blaze et al. [30], PRE extends the traditional public key encryption (PKE) to support the delegation of decryption rights. It allows a semi-trusted party called proxy to transform a cipher text encrypted under Alice’s public key into another cipher text of the same plaintext intended for Bob. The proxy, however, learns neither the decryption key nor the underlying plaintext [1]. Paper [56] paper presents a novel cipher text-policy attribute-based multi-use unidirectional proxy re-encryption scheme. In this scheme, the tree access policy can be used to handle and (¡Ä), or (¡Å) and threshold (of) operators.

Digital content is easily spread out in the era of cloud computing. [53] Proposed a novel identity-based access control approach for digital content based on ciphertext-policy attribute-based encryption (iDAC). In iDAC, the access control still works even the digital content is duplicated to another content server. Moreover, only one copy of encrypted digital content is required to share with multiple users. This could efficiently reduce the overhead of content servers.

In [54], for achieving access control and keeping data confidential, the data owners could adopt attribute-based encryption to encrypt the stored data. Users with limited computing power are however more likely to delegate the mask of the decryption task to the cloud servers to reduce the computing cost. This scheme achieves security against chosen-plaintext attacks under the k-multi-linear Decisional Diffie-Hellman assumption.

4. Background Work

In this section, Access Policies, Mathematical Background, assumptions and KP-ABE [15] scheme are presented.

4.1. Assumptions

Following assumptions are made [2]:

1) The cloud is honest-but-curious, which means that the cloud administrators can be interested in viewing user‘s content, but cannot modify it. This is a valid assumption that has been made in [2,34]. Honest-but-curious model of adversaries do not tamper with data so that they can keep the system functioning normally and remain undetected.
2) Users can have either read or write or both accesses to a file stored in the cloud.
3) All communications between users/clouds are secured by Secure Shell Protocol, SSH.

4.2. Formats of Access Policies

Access policies can be in any of the following formats: 1) Boolean functions of attributes, 2) Linear Secret Sharing Scheme (LSSS) matrix, or 3) Monotone span programs. Any access structure can be converted into a Boolean function [2,35]. An example of a Boolean function is $((a_1 \wedge a_2 \wedge a_3) \vee (a_4 \wedge a_5)) \wedge (a_6 \vee a_7))$, where $a_1, a_2, ..., a_7$ are attributes. Consider an access structure for which there exists a linear secret-sharing scheme that realizes it. It is known that for every LSSS realizable access structure, there exist a monotone span program (MSP) that computes the corresponding Boolean functions and vice versa. Thus, such an access structure can be represented by a monotone span program.

Secret-Sharing Schemes: Secret-sharing schemes (SSS) are used to divide a secret among a number of parties. The information given to a party is called the share (of the secret) for that party. Every SSS realizes some access structure that defines the sets of parties who should be able to reconstruct the secret by using their shares.

Let $Y:\{0,1\}^n \to \{0,1\}$ be a monotone Boolean function [2,36]. A monotone span program for Y over a field F is an $l \times t$ matrix M with entries in F, along with a labeling function $a:[1] \to [n]$ that associates each row of M with an input variable of Y, such that, for every $(x_1, x_2 ..., x_n) \in \{0,1\}^n$, the following condition is satisfied:

$$\begin{aligned} &Y(x_1,\ x_2,\ ...,\ x_n)\ =1 \\ &\Leftrightarrow \exists v \in F^{1\times l} : vM = [1,\ 0,\ 0,\ ...,\ 0] \\ &\text{and } (\forall i:\ x_{a(i)} =\ 0 \Rightarrow v_i =\ 0) \end{aligned}$$

In other words, $Y(x_1, x_2, ..., x_n) = 1$ if and only if the rows of M indexed by $\{i \mid x_{a(i)} = 1\}$ span the vector $[1, 0, 0, ..., 0]$.

4.2.1. 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 $0 < k_x \le num_x$. 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 $k_x = 1$.

Here the parent of the node x in the tree is denoted 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 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. Here the index values are uniquely assigned to nodes in the access structure for a given key in an arbitrary manner.

4.3. Mathematical Background

Bilinear pairings on elliptic curves is used. Let *G* be a cyclic group of prime order *q* generated by *g*. Let G_2 be a group of order *q*. We can define the map $e: G_1 \times G_1 \to G_2$. The map satisfies the following properties [3]:

1) $e(P^a, Q^b) = e(P,Q)^{ab}$ for all $P, Q \in G_1$ and $a, b \in Z_q$, $Z_q = \{0,1,2,...,q-1\}$.

2) Non-degenerate: $e(g,g) \neq 1$.

Bilinear pairing on elliptic curves groups is used. The choice of curve is an important consideration because it determines the complexity of pairing operations.

4.4. Key Policy-Attribute Based Encryption (KP-ABE)

Key Policy - Attribute Based Encryption [15] scheme consists of four algorithms, proceeds as follows:

4.4.1. System Setup

This is a randomized algorithm that takes no input other than the implicit security parameter. It outputs the public parameters PK and a master key MK.

Let G_1 be a bilinear group of prime order *p*, and let *g* be a generator of G_1. In addition, let $e: G_1 \times G_1 \to G_2$ denote the bilinear map. A security parameter, *k*, will determine the size of the groups. We also define the Lagrange coefficient $\Delta_{i,s}$ for $i \in Z_q$ and a set, *S*, of elements in $Z_q : \Delta_{i,s}(x) = \Pi_{j \in s, j \neq i}(x-j)/(i-j)$. We will associate each attribute with a unique element in Z_q^*.

Consider T be an access tree with root r. Consider T_x as 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, it can be denoted as $T_x(\gamma) = 1$. $T_x(\gamma)$ can be computed 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 $\text{Tx}(\gamma)$ returns 1 if and only if $\text{att}(x) \in \gamma$.

Define the universe of attributes $u = \{1, 2, \ldots\ldots, \text{n}\}$. Now, for each attribute $\text{i} \in u$, choose a number t_i uniformly at random from Z_q. Finally, choose *y* uniformly at random in Z_q. The published public parameters PK are

$$T_1 = g^{t1}, \; \ldots.., T_{|u|} = g^{\text{t}|u|}, Y = e(g,\, g)^y$$

The master key MK is:

$$t_1, \ldots\ldots., t_{|u|}, y.$$

4.4.2. Encryption (M, γ, PK)

This is a randomized algorithm that takes as input a message *M*, a set of attributes γ, and the public parameters PK. It outputs the cipher text *E*.

To encrypt a message $M \in G_2$ under a set of attributes γ, choose a random value $s \in Z_q$ and publish the cipher text as:

$$E = (\gamma, E' = MY^{\text{s}}, \{E_i = \text{T}^{\text{s}}{}_{\text{i}}\}_{i \in \gamma}).$$

4.4.3. Key Generation (*T*, MK, PK)

This is a randomized algorithm that takes as input – Access Tree *T* (an access structure *A*), the master key MK and the public parameters PK. It outputs a decryption key *D*.

The algorithm outputs a key that enables the user to decrypt a message encrypted under a set of attributes γ if and only if $T(\gamma) = 1$. The algorithm proceeds as follows. First choose 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 top-down 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, $d_x = k_x - 1$. Now, for the root node r, set $q_r(0) = y$ and d_r other points of the polynomial q_r randomly to define it completely. For any other node x, set $q_x(0) = q_{parent(x)}(\text{index}(x))$ and choose d_x other points randomly to completely define q_x.

Once the polynomials have been decided, for each leaf node x, we give the following secret value to the user:

$$D_x = g^{qx(0)/ti}, \text{ where } i = \text{att}(x)$$

The set of above secret values is the decryption key *D*.

4.4.4. Decryption (*E, D, PK*)

This algorithm takes as input - the cipher text *E* that was encrypted under the set γ of attributes, the decryption key D for access tree T (access control structure *A)* and the public parameters PK. It outputs the message M if $\gamma \in A$.

Decryption procedure is specified as a recursive algorithm. For ease of exposition, the simplest form of the decryption algorithm is presented. Let consider a recursive algorithm *DecryptNode*(*E, D, x*) that takes as input the cipher text $E = (\gamma, E', \{E_i\}_{i \in \gamma})$, the private key D (we assume the access tree *T* is embedded in the private key), and a node x in the tree. It outputs a group element of G_2 or $\perp$.

Consider *i* = att(*x*). If the node x is a leaf node then:

$$DecryptNode(E, D, x) = \text{Either } e(D_x, E_i)$$
$$= e(g^{qx(0)/ti}, g^{s.ti}) = e(g,\, g)^{s.qx(0)} \text{ if } i \in \gamma$$
$$\text{Or } \perp \text{ otherwise}$$

Now consider the recursive case when *x* is a non-leaf node. The algorithm *DecryptNode*(*E, D, x*) then proceeds as follows:

For all nodes z that are children of x, it calls *DecryptNode*(*E*, *D*, *x*) 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 $\perp$.

Otherwise, compute following and return the result:

$F_x = \Pi_{z \in sx} F_z^{\Delta i,s'x(0)}$, where $i = \text{index(z)}$ and $s'_x = \{\text{index(z)} : z \in s_x\}$

$= \Pi_{z \in sx}(e(g,g)^{s.qz(0)})^{\Delta i,s'x(0)}$

$= \Pi_{z \in sx}(e(g,g)^{s.qparrent(z)(index(z))})^{\Delta i,s'x(0)}$ (by construction)

$= \Pi_{z \in sx}(e(g,g)^{s.qx(i)})^{\Delta i,s'x(0)}$

$= e(g,g)^{s.qx(0)}$ (using polynomial interpolation)

The decryption algorithm simply calls the function on the root of the tree. It can be observed that $DecryptNode(E,D,x) = e(g,g)^{ys} = Y^s$ if and only if the cipher text satisfies the tree. Since, $E' = MY^s$ the decryption algorithm simply divides out Y^s and recovers the message M.

5. Proposed Methodology

In this section, ASP (Advanced Security Protocol) is presented which is a privacy-preserving cryptographic access control Protocol that enables end-users to securely store, share, and manage their sensitive data in untrusted cloud storage anonymously. ASP is scalable and supports lazy revocation. It can be easily implemented on top of existing cloud services and APIs. Its prototype can be demonstrated based on Amazon S3 [37] API.

Advanced Security Protocol (ASP) supports cryptographic key-updating scheme, referred to as AKU (Advanced Key Update) as well as Authentication and data Integrity scheme, referred to as AB-SIGN. The main advantage of the AKU scheme its support for efficient delegation and revocation of privileges in hierarchies without requiring complex cryptographic data structures. Authentication Scheme is attribute based which enables the verifier to ensure that a signature is produced by a sender/creator/writer whose access policy is satisfied by a set of attributes without learning the signer's identity.

First, a formal definition for secure key-updating schemes for hierarchical access is provided. Then, we give a concrete construction of a key-updating scheme based on ABE scheme. It supports both key revocation and hierarchical delegation of secret access keys.

5.1. Hierarchical KU Scheme: Model and Definition

Let *T = (V, E, O)* be a tree that represent a hierarchical access structure. More general access class hierarchies in which partially ordered access classes are represented by a DAG are studied in [34]. In our work, we are only interested in a special case where DAG is a tree. Each vertex v_i in $V = \{v_0, v_1, v_2, \ldots, v_n\}$ corresponds to an access class. v_0 is the root and an edge $e = (v_i, v_j) \in E$ implies that v_i class is the parent of class v_j. *O* is a set of sensitive data objects, each object o is associated with exactly one access class $V(o)$. In this model, any subject that can assume access rights at class v_i is also permitted to access any object assigned to a vertex that is a descendant of v_i.

Definition 1 The local time at vertex v_i is an integer t_i that increases (elapses) every time access rights of a subject to that class is revoked.

Definition 2 The global time associated with node v_i is a vector $T_i = (t_0, \ldots, t_i, \ldots, t_j)$ where t_j is the local time of j^{th} vertex on the path from root to vertex v_i on the access tree T.

Two instances of global time are comparable only if the vertices that they belong to are identical or one of them is the ancestor of the other one; We say $T_i < T_j$ if and only if T_i and T_j are comparable and all common components of T_i are less than the corresponding components in T_j. Similarly, we define comparative operators =,>,≤, and ≥.

Definition 3 A Hierarchical Key-Updating (*HKU*) Scheme consists of a root user and end users. An end user may be a reader, a writer, or both. There are five polynomial time algorithms *HKU = (Init, Derive, Encrypt, Decrypt, Update)* defined as follows:

- **Init** $(1^k, T)$ is a randomized process run by the root user which takes as input a security parameter k and an access hierarchy tree T and then generates and publishes a set of public parameters Pub and outputs the root key K_{v0}, $\perp$. It also initializes the state parameters including the value of local time at each vertex.
- **Derive** $(T, K_{(vi,Ti)}, v_j)$ is a randomized process run by the root user, reader or writer which using the private key $K_{(vi,Ti)}$ of v_i at time T_i derives a private key of target class v_j at its current global time T_j according to *T*. Derive computes the requested key only if v_i is an ancestor of v_j and $T_j = T_i$; otherwise, it outputs null ($\perp$).
- **Revoke** (T, v_i) run by the root user, reader or writer, increments the local time t_i of v_i by one, updates other state variables, and returns the updated tree T'.
- **Encrypt** (T, o_k) is a randomized algorithm called by writer that encrypts the data object o_k and returns the encrypted object C.
- **Decrypt** $(K_{(vi,Ti)}, C)$ is a deterministic process run by reader which takes a key and an encrypted object as inputs and returns the corresponding object in plaintext. This function can decrypt C only if it belongs to the same or a descendant of the access class that the key belongs to and the time that ok is encrypted at is less than or equal to T_i ; otherwise, it outputs null ($\perp$).

Definition 3 is a generalization of the definition of key-updating schemes in [38] and the definition of key allocation schemes for hierarchies in [39]. If we assign to *T* a tree of depth 1 where its leaves are a set of groups (i.e., remove hierarchies), our definition reduces to a key-updating scheme defined in [38] and if we remove the update process and the time dimension, our scheme reduces to key allocation scheme for hierarchies defined in [39]. Intuitively, a hierarchical key-updating scheme is secure if all polynomial time adversaries have at most a negligible advantage to break the cipher-text encrypted with the current-time key of a target class, assuming that the adversaries do not belong to higher (ancestor) target classes in the hierarchy, or possess keys for earlier time periods. In this model the adversary chooses her target at the beginning of the game and then adaptively queries the scheme.

We define the security model of hierarchical key-updating schemes as follows:

Definition 4 A hierarchical key-updating schemes is secure if no polynomial time adversary A has a non-negligible advantage (in the security parameter k) against the challenger in the following game (HKU game):

Choosing target: The adversary declares an access object $\tilde{v}$ and a time instance $T_{\tilde{v}}$ that she wishes to guess its corresponding private key (i.e. $K'(v',T')$).

Setup: The challenger runs Init$(1^k, T)$, and gives the resulting public parameters Pub and T to the adversary.

Key-Extraction Query: The adversary adaptively queries the private keys of polynomial number of vertices at any time that she wishes subject to the restriction that either the queried vertices are not an ancestor of (or equal to) $\tilde{v}$ or the time instance that they are being queried at is earlier than or equal to $T_{\tilde{v}}$.

Challenge: The adversary submits two equal length objects o_0 and o_1 belonging to the access class $\tilde{v}$. The challenger flips a random coin b, and encrypts o_b for time $T_{\tilde{v}}$ and submits the result to the adversary.

The adversary issues more Key-Extraction queries.

Guess: The adversary outputs a guess b' of b.

Adversary's advantage is the probability that her guess is correct: $Adv_A = Pr[b' = b]$. The HKU scheme is secure if the adversary's probability compared to random guessing (1/2) is negligible.

5.2. AKU: Confidentiality Scheme

In this section, a concrete construction for HKU scheme called Advanced Key Update (AKU) is presented. This scheme is based on the use of bilinear map and the difficulty of the Decisional Bilinear Diffie-Hellman problem. Our solution is realized on top of the Key-Policy Attribute-Based Encryption scheme (KP-ABE) [15] and invokes KP-ABE operations including Setup ABE, KeyGen ABE for private key generation, Encrypt_ABE for data encryption, and Decrypt_ABE for decryption.

5.2.1. Init(1^k, T)

The root user runs the Setup Attribute Based Encryption process with 1^k as security parameter to generate ABE public parameters and the master key MK. Publishes the ABE public parameters as Pub$_{abe}$.

Invoke KeyGen_ABE procedure using MK as the secret key and "$L_0 = v_0$" as its policy. Outputs the result as the root key ($K_{(v0,\perp)}$ = Key- Gen_ABE(MK, $L_0 = v_0$)).

To each vertex in T adds a local time variable t_i initialized to zero.

5.2.2. Derive(T, $K(v_i, T_i)$, v_j)

It is run by a user (root user, reader, or writer) with secret key $K_{(vi,Ti)}$ at time T_i to obtain the private key for node v_j.

If class v_j is not a descendant of class v_i, or the time T_i is not equal to current time T_j associated with v_j, then return null. Otherwise, denote $(u_1, u_2, \ldots, u_n)$ as the list of vertices in the path from v_i to v_j ; denote $(t_{u1}, t_{u2}, \ldots\ldots, t_{un}, t_{vj})$ on T as the list of current local time values of intermediate vertices (including v_j); and let d represent the depth of v_i.

The user performs the following operations:

1) Construct the access tree T' which corresponds to the following Boolean expression: ($L_d.v$ attribute represents vertex in d-th level, $L_d.t$ represents its current local time and ^ is conjunction operator.).

$$(L_{(d+1)}.v = u_1{}^\wedge .....{}^\wedge L_{(d+n)}.v = u_n$$
$${}^\wedge L_{(d+n+1)}.v = vj)^\wedge$$
$$(L_{(d+1)}.t = T_{u1}{}^\wedge .....{}^\wedge L_{(d+n)}.t = Tun$$
$${}^\wedge L_{(vj)}.t \le Tvj)$$

This Boolean expression restricts access to objects that belong to node vj or its descendants and are created at current time or before.

2) Denote the access tree of $K_{(vi,Ti)}$ by T. Using the procedure for delegation of private key in [15], add the access tree T' to the root of $K_{(vi,Ti)}$, increase its threshold by one, update and calculate the private parameters associated to the root according the protocol. In implementation section we provide more details on this procedure.

3) Output the resulting access tree and parameters as a private key $K_{(vj,Tj)}$ for v_j.

5.2.3. Encrypt(T, o_k)

Encryption of data is performed using key. Denote v_i as the access class that object o_k belongs to. $(v_i = V(o_k))$. Denote $(v_0, u_1, u_2, \ldots, u_n, v_i)$ as v_i's path and $T_i = (t_{v0}, t_{u1}, t_{u2}, \ldots.., t_{un}, t_{vi})$ as its current time according to T. A writer encrypts o_k as follows:

The attribute set is used as the public key for encryption. Set the attribute set γ as follows:

$$\gamma = \{L_0.v = v_0, \ldots\ldots, L_n.v = v_n, L_{n+1}.v = v_i;$$
$$L_0.t = t_{v0}, \ldots\ldots., L_n.t = t_{un}, L_{n+1}.t = t_{vi}\}$$

Use ABE encryption procedure to encrypt o_k with attribute set γ and return the resulting encrypted object. $(C = Encrypt\_ABE(Pubabe, \gamma, o_k))$.

5.2.4. Decrypt($K_{(vi, Ti)}$, C)

After encryption of the data using key, we get cipher text that is transmitted to receiver end. Receiver or reader decrypts the cipher text as follows:

- If the encrypted object C does not belong to the same access class vi as the key $K_{(vi,\ Ti)}$ or one of its descendants, or the time when C is encrypted is later than the time T_i when the key is generated, then return null ($\perp$).
- Otherwise, run ABE decryption procedure and return its result as output $(o_k = \text{Decrypt\_ABE}(K(_{(vi,\ Ti)}, C))$.

5.2.5. Revoke(T, v_i)

It is run by a user to increment the local time of v_i by one and then returns the updated tree T'.)

The correctness of AKU scheme follows the correctness of the key policy ABE scheme [15].

Theorem 1 Assuming the hardness of the Decisional BDH, AKU is a secure hierarchical key-updating scheme.

Proof 1 Sketch. It suffices to show that an adversary, who can play HKU game for AKU with non-negligible advantage, can also win KP-ABE game with a non-negligible probability, and thus break the security of KP-ABE and subsequently the Decisional BDH. Let A be an adversary who can win HKU game with non-negligible advantage $1/2 + €$. She can play KP-ABE Selective-Set model game as follows:

Init: A declares the set of attributes that corresponds to vertex $\tilde{v}$ and time $T_{\tilde{v}}$ as γ, the set of attributes that she wishes to be challenged upon.

Setup: This step is identical to Setup step in HKU game.

Phase 1: In this phase the adversary queries for the private keys for access structures (trees) T_j which correspond to that of keys that she would query in HKU game. Since, according to the protocol of HKU game, these keys belong to vertices that are not an ancestor of $\tilde{v}$ or their time is less than $T_{\tilde{v}}$, their access trees will not satisfy with attributes in γ (γ does not belong to T_j) and therefore they are legitimate queries.

Challenge: Identical to the Challenge step in HKU game.

Phase 2: Repeat Phase 1.

Guess: The adversary guesses b using the same strategy that she uses in HKU game. Since the data is encrypted under the same set of attributes and using the same procedure, she has the same non-negligible advantage to make the correct guess. This concludes our proof.

5.3. AB-SIGN: Authentication and Integrity Scheme

Proposed ASP protocol supports message authentication and data integrity using AB-SIGN scheme. Our design for AB-SIGN is based on the same technique introduced by by Moni Naor (Section 6 of [40]) for Identity Based Encryption and then extended in [41] for HIDS signature scheme. However, the original paper which introduces KP-ABE [15] does not present any signature scheme.

AB-SIGN scheme is an attribute based signature scheme which

1) Enable the readers to verify the integrity of data and ensure that it is produced by an authorized writer,
2) Enable the cloud service providers to validate incoming requests and block unauthorized accesses.

Definition 5 AB-SIGN is a signature scheme for KP-ABE [15] that it’s signing and verification methods are defined as follows. Let's say that the signer has a key K for policy P, and wants to sign message M. The verifier needs to verify that the signature is generated by a signer whose key policy satisfies attribute set *A:*

Signature: From K derive a key (K') which corresponds to a policy which is the concatenation of P and $(@S = M)$ ($@S$ is a reserved attribute for signatures). Send the derived key to the verifier as the signature.

Signature Verification: Generate a random token and encrypt it using the attribute set $A\text{U}\{@S = M\}$ and then decrypt the result using a key which is equal to the signature. If the result is equal to the original token the signature is valid (i.e. the attribute set A satisfies the signer's key policy.)

To prevent an attacker from using the signature method to derive a valid access key, we need to reserve the attribute `$@S$' for signature.

Theorem 2: Assuming the hardness of the Decisional BDH, AB-SIGN is a secure signature scheme.

Proof 2: Enforceability of AB-SIGN scheme follows immediately from the security of KP-ABE scheme. In AB-SIGN, a signature is a derived key from the actual write access key. Therefore, based on the security of KP-ABE derive operation; the only entity who can generate the signature is the owner of the write access key. Moreover, security of derive operation guarantees that the verifier cannot guess the original access key from the derived key.

6. Implementation and Operation

The ASP protocol runs between the root user, end-user (reader or writer), and the cloud providers. The root user may be a system administrator in the data owner's organization, who can specify the access privileges of end-users. The end-users may further delegate their access privileges to other individuals for easy sharing. We achieve the revocation of privilege by encoding the validity period in the private keys of users and advancing time with respect to the target hierarchy or data object. Another advantage of our ASP framework for use in cloud storage is the support of anonymous access.

ASP protocol requires three repositories: *Meta-data Directory, Data Store* and *Key-store* as shown in Figure 1.

Meta-data Directory: All meta-data associated with hierarchies and data objects are maintained in this repository. ASP requires two properties for each object: Read Access Revision (RAR) and Write Access Revision (WAR). These two properties play the role of local time in AKU for read and write access, respectively. In order to compute Read/Write Access Revision Vectors (which correspond to global time instances in ASP), the cloud provider that hosts Meta-data Directory needs to provide an API for querying RAR and WAR values of multiple directories in a single request. All existing cloud-based databases (also known as `NoSQL systems’ or `schema-free database’ such as Amazon SimpleDB [43], Microsoft Azure SQL [42], and Google's AppEngine [44] database

(Bigtable [45])) satisfy this requirement and therefore qualify to host an ASP Meta-data Directory. For our experiments we use Amazon SimpleDB [43].

Data Store: This repository contains the actual content of each data object. Any cloud key-value based storage system such as Amazon S3 [37] can be used as ASP Data Store. In ASP, keys are hierarchical path name of data objects and values are the actual content of corresponding data objects. Cloud key-value storage providers are tuned for high throughput and low storage cost; these features make them a good candidate for ASP Data Store.

Key-store: Key-Store is a secure local repository which having all read/write access keys of end-users. Each key-store contains all public parameters as well as read/write access key entries of all data-objects and categories that the end user has access to. Each access key entry contains the following fields: object path, access type (read/write), granter's identity, and secret key. The Key-store provides an API that given user's credential and a path, returns the first key entry that its path is a prefix of the input path.

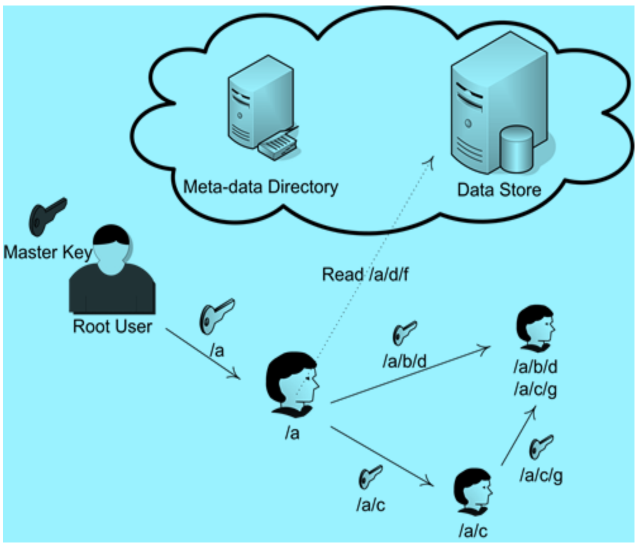

Figure 1. ASP Working Environment

All major participants of ASP protocol are shown in Figure 1. In ASP protocol, end-users can enforce access control on their own data without fully trusting or relying on the cloud providers. Here keys are distributed and managed in a distributed fashion. Solid arrows represent access delegation.

Working with ASP

- To work with ASP, the root user needs to follow the following steps:
- Sign up for cloud services required for hosting Meta-data Directory and Data Store.
- Run ***Init*** procedure according to AKU scheme to generate public parameters and the master key.
- Save the master key and public parameters in the root's Key-store.
- Share the public parameters with the cloud service providers that support ASP request authorization.
- Create an entry in Meta-date Directory that corresponds to the root directory. The WAR and RAR numbers of the root directory entry are initialized to zero.

6.1. ASP Operation

The basic operations supported by ASP include: write, read, delegate, and revoke. Each basic operation leads to calls to Meta-data directory and/or Data Store. Other operations such as create, remove, rename, update for directories and data objects can be defined similarly. AB-SIGN Scheme is needed before performing the basic operations to maintain the authenticity and data integrity. Requirement of AB-SIGN of ASP to perform operations enables cloud providers to block unauthorized request.

6.1.1. Write Operation

Figure 2 shows the idea about write operation using ASP protocol. To write into a specific data object, the end-user needs to perform the following steps:

1) Retrieve the required write access key from the local Key-store.
2) Query Meta-data directory to get read access revision (RAR) vector of the target object.
3) Using AKU scheme, encrypt the data by the retrieved RAR vector and its path.
4) Using AB-SIGN scheme, sign the data by his write access key.
5) Construct a key-value pair where the key is equal to the path of data object and the value is the encrypted data and corresponding signature. Store the pair in Data Store.
6) To prevent destructive writes by unauthorized users, the Data Store can query write access revision (WAR) vector of that object from the Meta-data Directory to validate the signature of request.

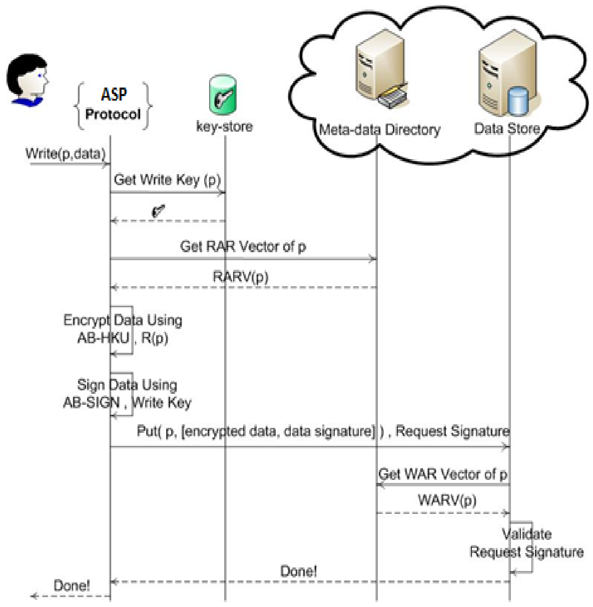
ASP Protocol
key-store
Meta-data Directory
Data Store
Write(p,data)
Get Write Key (p)
Get RAR Vector of p
RARV(p)
Encrypt Data Using AB-HKU , R(p)
Sign Data Using AB-SIGN , Write Key
Put(p, [encrypted data, data signature]) , Request Signature
Get WAR Vector of p
WARV(p)
Validate Request Signature
Done!
Done!

Figure 2. Write Operation

6.1.2. Read Operation

Figure 3 shows the idea about read operation using ASP protocol. To ensure the data is produced by an authorized writer, the reader needs to validate the corresponding signature using AB-SIGN signature scheme. Then the reader can decrypt the data using its read access key and AKU scheme. To read a specific data object stored using ASP protocol, the end-user needs to do the following steps:

1) Retrieve the required read access key from the local Key-store.

2) Using AKU scheme and the read access key, decrypt the encrypted data.
3) Using AB-SIGN signature scheme, validate the signature to ensure that data is produced by a user who has the proper write access.
4) Return the decrypted data.

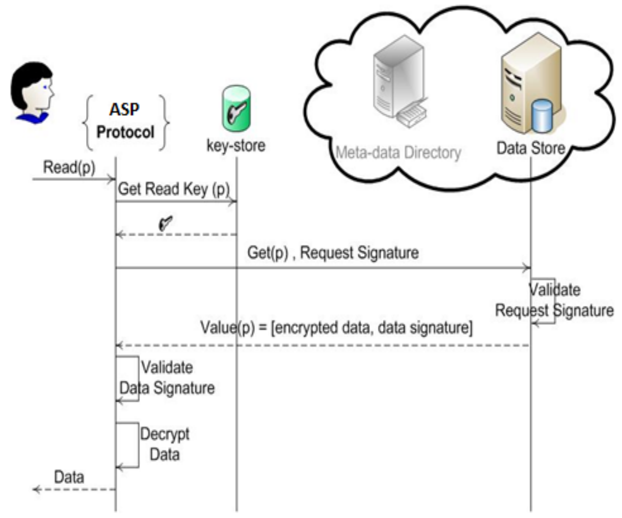

Figure 3. Read Operation

6.1.3. Delegation Operation

Delegation operation can be run by a user to authorize another user a subset of his access privileges as shown in Figure 1. It requires three input parameters: the identity of the delegate, the resource path, and access type (read/write). The steps required for this operation are listed below:

1) From the local Key-store, get the access key that matches the target resource path and access type.
2) Query Meta-date Directory to get the read/write access revision (RAR/WAR) vector of target resource.
3) Run Derive operation, as defined in AKU scheme, to generate the required access key.
4) Send the generated access key to the delegate through a secure communication channel.

6.1.4. Revocation Operation

This facility reduces the overhead on the data center by restricting fake user. To revoke a user's access on a specific directory or data object, the authorized user needs to make a signed request to the Meta-data Directory to increase the corresponding access revision number. To ensure the integrity of access revision numbers, these entries should be signed by the requester.

7. Analysis of Proposed Technique

In this section, some experimental results are provided which show the performance overhead of our ASP protocol.

Pre-computation and caching: As discussed in the previous sections, to overcome the limitation of fixed attributes, we adopted large universe construction of KP-ABE. However, in this construction the process of mapping an attribute to the bilinear group $G(i.e.\{1,0\}^* \rightarrow G)$ is very expensive (on average 22 *ms* per attribute). In our KP-ABE library every bit of a numerical attribute gets translated into a symbolic attribute. For example, a 10-bit representation of the numerical attribute *li* = 352 gets translated into a list of symbolic attributes shown in Table 1.

Table 1. Symbolic representation of attribute *li* = 352

[li@0=1, li@1=1, li@2=0, li@3=0, li@4=0, li@5=1, l i@6=1, li@7=0, li@8=1, li@9=0]

Also, all numerical comparisons get translated into symbolic matching policies. For example, Table policy 2 corresponds to the numerical comparison *li* < 356.

Table 2. KP-ABE policy for *li* < 356

(2 li@9=0 (1 (2 li@7=0 (1 (1 (2 li@4=0 (2 li@3=0 (1 li@1=0 li@2=0))) li@5=0) li@6=0)) li@8=0))

In ASP, every level of an object's path has two numbers associated with it – read access revision number and write access revision number. Therefore, these numerical attributes lead too many symbolic attributes which their mapping cost create a significant over-head. Since the value of each bit is either zero or one, we pre-compute the mapped values of these symbols and during the startup process load them into the framework. Moreover, at runtime, we cache the mapped value of each path segment in a hash table so that it can be reused. Using the described pre-computation and caching techniques, we were able to significantly reduce the computational cost associated with required KP-ABE crypto operations.

Security Process: In ASP the actual content of data is encrypted using either a symmetric-key or asymmetric-key encryption scheme based on user choice. And only the corresponding symmetric/asymmetric keys are encrypted by KP-ABE scheme. By default our framework uses AES (Advanced Encryption Standard) [4,12] for symmetric encryption with the default key length of 128 bits. Similarly, AB-SIGN signature scheme is performed on fixed-length digest of data. Our framework, by default, uses SHA-1 [4,11] as the digest hash function. SHA-1 generates 160-bit message digest of data.

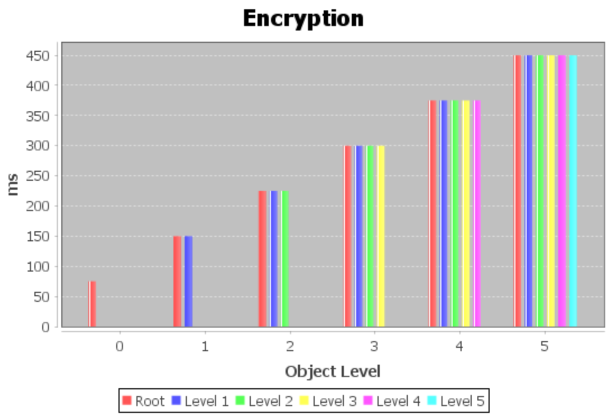

Figure 4. Encryption Analysis

The Figure 4 and Figure 5 show the overhead of encryption and decryption schemes while Figure 6 and Figure 7 show signature and sign verification schemes of ASP protocol on top of underlying symmetric-key encryption and hashing schemes. In our experiments numerical attributes are of size 10 bits.

Figure 4 shows how the cost of ASP encryption relates to the user's access level and hierarchy level of the data object. In KP-ABE the encryption time is only a function of number of attributes, which linearly increases as the object level increases. As a result, ASP encryption cost linearly increases as the hierarchy level of the object increases, but it is independent of the user's access level.

By contrast, as Figure 5 shows, decryption time is just a function of user's access level. That is because in KP-ABE, decryption time is a function of complexity of access structure that linearly increases as user's access level increases. Decryption time is independent of hierarchy level of the encrypted object.

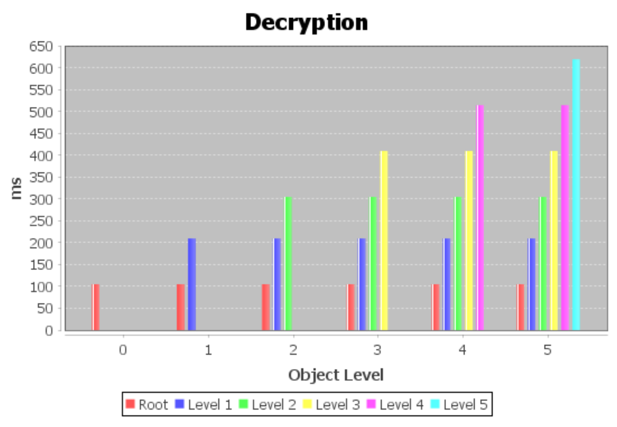

Figure 5. Decryption Analysis

Figure 6 and Figure 7 show the overhead of ASP signing and signature verification on signed objects in different hierarchy levels for users with different access levels respectively. In ASP, as Figure 6 illustrates, signature cost is independent of the hierarchy level of data objects; it only depends on access level of the user. This is because proposed AB-SIGN signature scheme is based on KP-ABE derive operation which its complexity linearly increases as the complexity of the access structures increases.

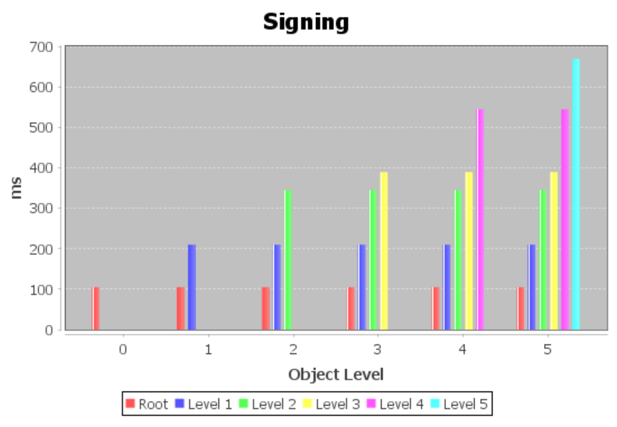

Figure 6. Message Signing Analysis

In AB-SIGN scheme, each signature verification operation requires KP-ABE encryption and decryption, therefore its computational cost depends on the user's access level as well as the hierarchy level of data object. Figure 7 shows how the time required for signature verification increases linearly as the access level of user decrease and the hierarchy level of data object increases.

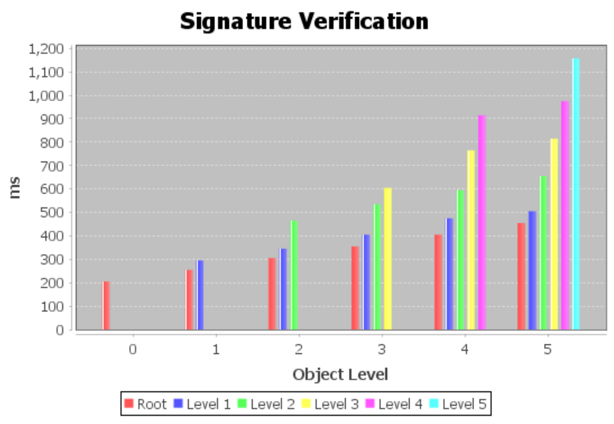

Figure 7. Signature Verification Analysis

Figure 8 and Figure 9 show the combined overhead cost of read and write operations in ASP. To perform ASP write operation, a user needs to encrypt and sign the data objects. The portion of cost below the white indicator is related to encryption and the rest is the cost associated with signature as shown in Figure 8.

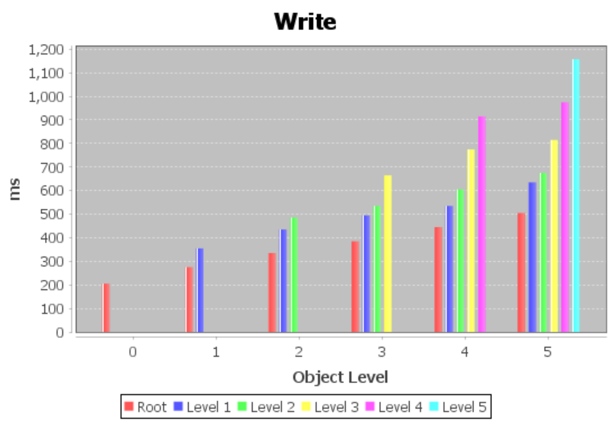

Figure 8. Write Operation Analysis

ASP read operation includes cost of decryption and signature verification. If we divide the graph given in Figure 9 by horizontal line, portion below and above of that line, shows the overhead cost for decryption and signature verification, respectively.

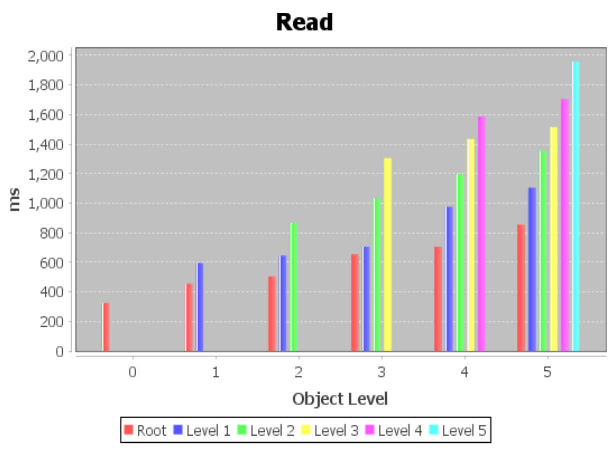

Figure 9. Read Operation Analysis

7.1. Complexity Analysis

In this section, we denote *N* as the number of attribute authorities, *I* as the size of the entire attribute set and *X* as the number of nodes in a tree T_p . Table 3 shows the complexity comparisons of proposed approach with existing approaches proposed in [9] and [26].

Table 3. Complexity Comparisons

Phase	Chase *et al.* [26]	Yu *et al.* [9]	ASP
System Setup	$O(1)$	$O(I)$	$O(1)$
KeyGen	$O(N + I)$	$O(X)$	$O(N + I)$
Encryption	$O(I)$	$O(I)$	$O(X \cdot K)$
Decryption	$O(N \cdot I)$	$O(max(X, I))$	$O(X)$
Revocation		$O(I)$	$O(X \cdot K)$

System Setup:

When the system is setup, ΠY^k is computed by any one of the authorities and sent to others, whose complexity is $O(N)$. Then, secret parameters xk's are calculated within the clusters. The complexity of that calculation is $O(C2 \cdot NC) = O(C \cdot N)$, but C is a constant number, so $O(C \cdot N) = O(N)$. Therefore, the total complexity is *O(N)*. However, since we have N authorities per system, the complexity per authority is *O(1)*.

KeyGen

In the Attribute Key Generation, $g^{\Sigma Vi}$ is computed by N authorities, and $D_i = H(att(i))^{ri}$. $g^{\Sigma Vi}$ is computed for I times by one attribute authority. Therefore, the total complexity of Attribute Key Generation is $O(N2 + I.N)$. In the Aggregation of Two Keys, a user aggregates the I components, thus the computation complexity of this operation is $O(I)$. So, the complexity per authority is $O(N + I)$.

Encryption

In this phase at every non-leaf node, a polynomial is chosen and k_{x-1} numbers are to be found to determine the polynomial, where x is the threshold value. Therefore, denoting the average threshold value to be K , the computation complexity of this process is $O(X.K)$.

Decryption

Decryption is a recursive algorithm, and it is executed exactly once at every node in a Breadth-First-Search manner, therefore the computation complexity of this process is $O(X)$.

Revocation

Revocation operation has the same complexity as the addition of Encryption and Decryption. Its complexity is $O(X.K)$.

The comparison between proposed ASP protocol and the different multi-authority schemes is shown in Tables 4 and 5. By $|U|$, $|A_U|$, and $|A_C|$, we denote the number of the universal attributes, the attributes held by user U, and the attributes required by the cipher text, respectively. I_U and I_C denote the index set of the authorities. By E and P, we denote one exponential and one paring operation, respectively. By L_{G1} and L_{G2}, we denote one element in group G1 and one element in group G2, respectively. N denotes the number of the authorities in the systems. Table 4 shows the ideas about operation cost for various MA-ABE schemes while Table 5 shows the working ideas comparison of existing MA-ABE technologies.

Table 4. Comparison of computational cost

Schemes	Authority setup	KeyGen	Encryption	Decryption
Chase's [22]	$(\|U\|+1)E$	$(\|A_U\|+1)E$	$(\|A_C\|+2)E$	$\|A_C\|E+(\|A_C\|+1)P$
Han et al.'s [23]	$(\|U\|+2N)E$	$(\|A_U\|+3\|I_U\|)E$	$(\|A_C\|+3)E$	$\|A_C\|E+(\|A_C\|+\|I_C\|+1)P$
Chase and Chow [26]	$(\|U\|+2N)E$	$(\|U\|+\|I_U\|^2)E$	$(\|A_C\|+2)E$	$\|A_C\|E+(\|A_C\|+1)P$
Our ASP	$(\|U\|+2N)E$	$(\|U\|+\|I_U\|^2+1)E$	$(\|A_C\|+3)E$	$\|A_C\|E+(\|A_C\|+1)P$

Table 5. Working Idea Comparison for MA-ABE

Scheme	Security Model	Used ABE	Cipher text Length	Central Authority	Authenticity
Chase's [49]	Selective	KP-ABE	$(\|A_C\|+1)L_{G1} + L_{G2}$	Yes	No
Han et al.'s [50]	Selective	KP-ABE	$(\|A_C\|+2)L_{G1} + L_{G2}$	No	Yes
Lin et al.'s [51]	Selective	FIBE	$(\|A_C\|)L_{G1} + L_{G2}$	No	No
Chase and Chow [52]	Selective	KP-ABE	$(\|A_C\|+1)L_{G1} + L_{G2}$	No	No
Our ASP	Selective	KP-ABE	$(\|A_C\|+2)L_{G1} + L_{G2}$	Yes	Yes

7.2. Security Analysis

In this section we state the security guarantees provided by ASP protocol.

Confidentiality: Our solution ensures that only the users who have the most recent version of the access key of the data object or one of its ancestor directories can decrypt it. The confidentiality of stored data is protected under our protocol because writers always encrypt the data objects by their path and most recent read access revision (RAR) vector according to AKU scheme. The cloud provider or other unauthorized users cannot gain any information that helps them to guess the access key of unauthorized data objects.

Integrity: The integrity of stored data is preserved. This guarantee is realized by requiring writers to sign the data by their write access key using AB-SIGN scheme. We require readers to validate writer's signature to ensure that it is produced by an authorized writer (i.e. a user with write access to that data object or on of its parent directories). Because meta-data entries stored in the Meta-data Directory are also required to be signed by the end-users, any unauthorized change in Meta-data Directory is detectable by the reader.

Authenticity and Anonymity: The end users are anonymous to each other and to the cloud providers. The signatures used in proposed authorization do not contain any identify information. During the course of protocol, the end-users do not reveal any information about their credentials. AB-SIGN signatures bound to the data objects and requests; include only attributes related to the location and global time of those objects.

Collusion-resistance: Security of KP-ABE guarantees that unauthorized users and malicious cloud service providers cannot collude to guess access key to an unauthorized data object.

8. Conclusion and Future Work

As people are becoming more concerned about their privacy these days, the privacy-preserving is very important over the cloud. Security issues can be categorized into sensitive data access, data segregation, privacy, bug exploitation, recovery, accountability, malicious insiders, management console security, account control, and multi-tenancy issues. The three basic requirements of security: confidentiality, integrity and availability are required to protect data throughout its lifecycle. Data must be protected during the various stages of creation, sharing, archiving, processing etc. However, situations become more complicated in case of a public cloud.

In this paper, Advanced Security Protocol (ASP) is presented that supports Hierarchical Key-Updating scheme. This protocol illustrates how recent cryptographic schemes can be utilized to develop an effective client-side access control protocol for protecting confidentiality and integrity of data stored in untrusted cloud storage. Proposed ASP protocol also includes an attribute based signature(AB-SIGN) scheme that enables cloud providers to ensure that requests are submitted by authorized end-users, without learning their identities. Using the key-updating and signature schemes, proposed idea is developed, implemented, and evaluated. Presented protocol is a scalable cryptographic access control protocol for hierarchically organized data. Proposed ASP protocol achieves Confidentiality, Data Integrity and Authenticity as well as reduces the overhead on web by restricting fake users.

In future, work can be done on security systems for various web based services.

References

[1] Shyam Nandan Kumar, and Amit Vajpayee, “A Survey on Secure Cloud: Security and Privacy in Cloud Computing.” American Journal of Systems and Software, vol. 4, no. 1 (2016): 14-26.

[2] Shyam Nandan Kumar, “Cryptography during Data Sharing and Accessing Over Cloud.” International Transaction of Electrical and Computer Engineers System, vol. 3, no. 1 (2015): 12-18.

[3] Shyam Nandan Kumar, “DecenCrypto Cloud: Decentralized Cryptography Technique for Secure Communication over the Clouds.” Journal of Computer Sciences and Applications, vol. 3, no. 3 (2015): 73-78.

[4] Shyam Nandan Kumar, “Review on Network Security and Cryptography.” International Transaction of Electrical and Computer Engineers System, vol. 3, no. 1 (2015): 1-11.

[5] Shyam Nandan Kumar, “World towards Advance Web Mining: A Review.” American Journal of Systems and Software, vol. 3, no. 2 (2015): 44-61.

[6] "The NIST Definition of Cloud Computing". National Institute of Standards and Technology. Retrieved 24 July 2011.

[7] Mather T, Kumaraswamy S, Latif S (2009) Cloud Security and Privacy. O’Reilly Media, Inc., Sebastopol, CA.

[8] A. Verma and S. Kaushal, “Cloud Computing Security Issues and Challenges: A Survey”, Proceedings of Advances in Computing and Communications, Vol. 193, pp. 445-454, 2011.

[9] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. “Achieving secure, scalable and fine-grained data access control in cloud computing”. In Proceedings of the 29th conference on Information communications, INFOCOM'10, pp. 534-542, Piscataway, NJ, USA, 2010. IEEE Press.

[10] Wayne Jansen, Timothy Grance, “NIST Guidelines on Security and Privacy in Public Cloud Computing”, Draft Special Publication 800-144, 2011.

[11] RFC 3174, US Secure Hash Algorithm 1 (SHA1) http://www.ietf.org/rfc/rfc3174.txt.

[12] Joan Daemen and Vincent Rijmen. Rijndael/aes. “In Encyclopedia of Cryptography and Security”. 2005.

[13] Jon Marler, “Securing the Cloud: Addressing Cloud Computing Security Concerns with Private Cloud”, Rackspace Knowledge Centre, March 27, 2011, Article Id: 1638.

[14] A. Sahai and B. Waters, “Fuzzy identity-based encryption”, in EUROCRYPT, ser. Lecture Notes in Computer Science, vol. 3494. Springer, pp. 457-473, 2005.

[15] 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 (CCS ’06). ACM, 2006, pp. 89-98.

[16] R. Ostrovsky, A. Sahai, and B. Waters, “Attribute-based encryption with non-monotonic access structures,” in Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07), pp. 195-203, November 2007.

[17] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute-based encryption,” in Proceedings of the IEEE Symposium on Security and Privacy (SP '07), pp. 321-334, May 2007.

[18] L. Cheung and C. Newport, “Provably secure ciphertext policy ABE,” in Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07), pp. 456-465, November 2007.

[19] B. Waters, “Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization,” in Public Key Cryptography (PKC '11), pp. 53-70, Springer, Berlin, Germany, 2011.

[20] A. Lewko, T. Okamoto, A. Sahai, and B. Waters, “Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption,” in Advances in Cryptology: EUROCRYPT 2010, vol. 6110 of Lecture Notes in Computer Science, pp. 62-91, Springer, Berlin, Germany, 2010.

[21] K. Emura, A. Miyaji, K. Omote, A. Nomura, and M. Soshi, “A ciphertext-policy attribute-based encryption scheme with constant ciphertext length,” International Journal of Applied Cryptography, vol. 2, no. 1, pp. 46-59, 2010.

[22] M. Chase, “Multi-authority attribute based encryption,” in Theory of Cryptography, vol. 4392 of Lecture Notes in Computer Science, pp. 515-534, Springer, Berlin, Germany, 2007.

[23] J. Han, W. Susilo, Y. Mu, and J. Yan, “Privacy-preserving decentralized key-policy attribute-based encryption,” IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 11, pp. 2150-2162, 2012.

[24] V. Bozovic, D. Socek, R. Steinwandt, and V. I. Villanyi, “Multi-authority attribute-based encryption with honest-but-curious central authority,” International Journal of Computer Mathematics, vol. 89, no. 3, pp. 268-283, 2012.

[25] H. Lin, Z. Cao, X. Liang, and J. Shao, “Secure threshold multi authority attribute based encryption without a central authority,” Information Sciences, vol. 180, no. 13, pp. 2618-2632, 2010.

[26] M. Chase and S. S. M. Chow, “Improving privacy and security in multi-authority attribute-based encryption,” in Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09), pp. 121-130, Chicago, Ill, USA, November 2009.

[27] N. Attrapadung and H. Imai, “Dual-policy attribute based encryption,” in Applied Cryptography and Network Security, pp. 168-185, Springer, Berlin, Germany, 2009.

[28] Guojun Wang, Qin Liu, Jie Wu and Minyi Guo, “Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers”, 2011.

[29] M. Mambo and E. Okamoto, “Proxy cryptosystems: delegation of the power to decrypt ciphertexts,” IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. 80, no. 1, pp. 54-62, 1997.

[30] M. Blaze, G. Bleumer, and M. Strauss, “Divertible protocols and atomic proxy cryptography,” in Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT '98), pp. 127-144, Espoo, Finland, 1998.

[31] Tatsuaki Okamoto and Katsuyuki Takashima, “Decentralized Attribute-Based Signatures” , Public-Key Cryptography – PKC 2013, Springer Berlin Heidelberg, pp 125-142.

[32] Xiaofeng Chen, Jin Li, Xinyi Huang, Jingwei Li, Yang Xiang and Duncan S. Wong, “Secure Outsourced Attribute-Based Signatures”, pp: 3285-3294, IEEE, vol. 25, (2014).

[33] Wenyi Liu, Uluagac, A.S. and Beyah, R., “MACA: A privacy-preserving multi-factor cloud authentication system utilizing big data”, IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2014, pp. 518-523, Toronto, ON.

[34] S. Yu, C. Wang, K. Ren, and W. Lou, “Attribute based data sharing with attribute revocation” in ACM ASIACCS, pp. 261-270, 2010.

[35] A. B. Lewko and B. Waters, “Decentralizing attribute-based encryption”, in EUROCRYPT, ser. Lecture Notes in Computer Science, vol. 6632. Springer, pp. 568-588, 2011.

[36] H. K. Maji, M. Prabhakaran, and M. Rosulek, “Attribute-based signatures”, in CT-RSA, ser. Lecture Notes in Computer Science, vol. 6558. Springer, pp. 376-392, 2011.

[37] Amazon S3 . http://aws.amazon.com/s3/.

[38] Michael Backes, Christian Cachin, and Alina Oprea. “Secure Key-Updating for Lazy Revocation”,. In Research Report RZ 3627, IBM Research, pages 327-346. Springer, 2005.

[39] Marina Blanton, Nelly Fazio, and Keith B. Frikken. “Dynamic and Efficient Key Management for Access Hierarchies”. In Proceedings of the ACM Conference on Computer and Communications Security, 2005.

[40] Dan Boneh and Matthew Franklin. “Identity-based encryption from the weil pairing”. SIAM J. Comput., 32: 586-615, March 2003.

[41] Craig Gentry and Alice Silverberg. “Hierarchical ID-based cryptography”. In ASI- ACRYPT, pp. 548-566, 2002.

[42] SQL Data Services/Azure Services Platform. http://http://www.windowsazure.com.

[43] Amazon SimpleDB. http://aws.amazon.com/simpledb/.

[44] Google App Engine. http://appengine.google.com.

[45] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: “A distributed storage system for structured data”. In Proceedings of the 7th symposium on Operating systems design and implementation - volume 7, pp. 205-218, 2006.

[46] P. Sharma, S. K. Sood, and S. Kaur, “Security Issues in Cloud Computing”, Proceedings of High Performance Architecture and Grid Computing, Vol. 169, pp. 36-45, 2011.

[47] Alessandro Perilli, Claudio Criscione, “Securing the Private Cloud”, Article on Secure Networks, Virtualization.info. http://virtualization.info/en/security/privatecloud.pdf.

[48] Thomas W. Shinder, “Security Issues in Cloud Deployment models”, TechNet Articles, Wiki, Microsoft, Aug, 2011.

[49] Craig Gentry, A FULLY HOMOMORPHIC ENCRYPTION SCHEME”, PhD Thesis, STANFORD UNIVERSITY, September 2009.

[50] Cloud Security Alliance (2012), “SecaaS implementation guidance, category 1: identity and Access management”. Available: https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_1_IAM_Implementation_Guidance.pdf.

[51] Ron Rivest (2002-10-29). "Lecture Notes 15: Voting, Homomorphic Encryption.

[52] B. R. Kandukuri, P. V. Ramakrishna, and A. Rakshit, “Cloud security issues”, in Proceedings of the IEEE International Conference on Services Computing (SCC '09), pp. 517-520, September 2009.

[53] Win-Bin Huang and Wei-Tsung Su, “Identity-based access control for digital content based on ciphertext-policy attribute-based encryption”, International Conference on Information Networking (ICOIN), IEEE, pp. 87-91, Cambodia, 2015.

[54] Jie Xu, Qiaoyan Wen, Wenmin Li, Zhengping Jin, “Circuit Ciphertext-Policy Attribute-Based Hybrid Encryption with Verifiable Delegation in Cloud Computing”, IEEE Transactions on Parallel and Distributed Systems, vol. 27, issue: 1, pp. 119-129, 2015.

[55] Win-Bin Huang, Wei-Tsung Su, and Chiang-Sheng Liang, “A threshold-based key generation approach for ciphertext-policy attribute-based encryption”, Seventh International Conference on Ubiquitous and Future Networks (ICUFN), IEEE, pp. 908-913, Sapporo, 2015.

[56] Juanjuan Li, Zhenhua Liu, and Longhui Zu, “Chosen-Ciphertext Secure Multi-use Unidirectional Attribute-Based Proxy Re-Encryptions”, Ninth Asia Joint Conference on Information Security (ASIA JCIS), IEEE, pp. 96-103, Wuhan, 2014.

[57] Han Yiliang, Jiang Di , Yang Xiaoyuan, “The Revocable Attribute Based Encryption Scheme for Social Networks”, International Symposium on Security and Privacy in Social Networks and Big Data (SocialSec), IEEE, pp. 44-51, Hangzhou, 2015.

[58] Lin You, and Lijun Wang, “Hierarchical authority key-policy attribute-based encryption”, IEEE 16th International Conference on Communication Technology (ICCT), pp. 868-872, Hangzhou, 2015.