Novel Design and Implementation of Decentralized Access Control with Anonymous Authentication of Data Stored in Grid extended Clouds

P.NAGAMANI
Research Scholar CSE,
Rayalaseema University Kurnool
A.P. India
nagamani.koli@gmail.com

DR. Y.RAMADEVI
Professor and HOD CSE,
C.B.I.T., Affiliated to O.U.
Hyderabad, T.S. India
yrdcse.cbit@gmail.com

DR.M.UPENDRA KUMARI
Associate Professor CSE
MGIT JNTU H Hyderabad T.S.
India
uppi_shravani@rediffmail.com

Abstract:
The novel research area of a middleware technology framework for data management and distribution in Grid Computing, had now evolved into niche integrated areas of Cloud Computing, Big Data and Internet of Things for Security and Privacy. This paper is a novel design and implementation of Decentralized access control with anonymous authentication of data stored in Grid extended Clouds.

Index Terms—Grid Computing, Middle ware Framework, Data Management and Distribution, Security and Privacy, Cloud Computing

I. INTRODUCTION TO GRID COMPUTING EXTENDED CLOUDS

A distributed system consists of a set of computational nodes connected by a communication network that cooperate to accomplish common tasks. we will review the benefits of using a distributed system, and the challenges facing the designers.

Distributed System:

The basic requirements from a distributed system are that the nodes would be autonomous so that they can work independently; the network should be connected, that is any node should have a communication link directly or indirectly to any other node; and there should be a coordination mechanism of the nodes to cooperate to achieve common goals.

There are a number of benefits to be gained by utilizing distributed systems. One of the obvious advantages of using a distributed system is resoure sharing. Access to a central resource has two disadvantages as this central site becomes a bottleneck for communications and also is a single point of failure. Distributing the resource such as the datable and peripherals over a network overcomes these problems.

Resources and computation can be replicated at various sites providing fault tolerance as a replica may be substituted in the case of the disfunctioning of a node. This type of fault tolerance is an important reason to employ distributed systems. It is also possible for the application to be inherently distributed such as bank transaction systems and airline transaction systems where employment of distributed systems is inevitable

A distributed system can be modeled as a graph G(V,E) conveniently where V is the set of vertices and E is the set of edges of G. The computing nodes of the distributed system are represented by the vertices of the graph, and an edge exists between the nodes of there is a communication link between them. The first thing that may be noticed is that the graph is connected, providing a communication path between any pair of nodes. Many nodes are not directly connected to each other; therefore, they have to rely on their neighbor nodes to communicate with the other nodes of the network we will use graphs to represent distributed system and show the execution of a distributed algorithm in these graphs frequently .we will first describe platforms and models for distributed computing in sects. Distributed Computing Platforms

Due to the recent technological advancements, in the last few decades, we have witnessed diverse distributed system platforms such as the Grid,The Cloud, mobile adhoc networks, and wireless sensor networks that are described below.

II LITERATURE SURVEY

The Grid

The Grid consists of loosely coupled, heterogeneous, and geographically dispersed

computing elements that are connected by a network acting together to perform large tasks [3]. These computationally intensive scientific tasks may include various applications such as seismic analysis, drug discovery, and bioinformatics problems. Grid computing provides effective usage of the unused processing power and results in decreased completion time for a task due to parallelization.

The size of a grid varies from a small network of workstations in a corporation to thousands of nodes across many networks and nations. Grids require general software libraries called the middleware to accomplish coordination among a large number of nodes that comprise them. Resource discovery is the process of finding the location of the required resources such as the database tables in the Grid [2]. Resource allocation process, on the other hand, tries to map these resources to the application requirements for the best performance. Both resource discovery and resource allocation are active research areas for the grids. An important problem with the grids is that nodes may abort due to faults that may be difficult to find and take necessary action due to the lack of central control. For this reason, fault tolerance and also load balancing is another important research area in the grids [8]. Lack of central control and the need to provide access to a large number of users requires protection due to possible risks. The European Grid Infrastructure [EGI) is a grid for high-energy physics, earth observation, and biology application [6], and in the United States, the National Grid (USNG) [9] is prototyping a computational grid for infrastructure and an access grid for people and effective usage of the unused processing power and results in decreased completion time for a task due to parallelization.

In shared-memory models, processes communicate by reading and writing to shared memory. Synchronization is an important issue also in shared-memory systems. Distributed shared-memory systems implement shared memory model over the message passing model to use the available shared memory software modules conveniently.

Software Architecture

The software modules at a node of a distributed computing system consist of the distributed algorithm that is the application software: the local operating system, the middleware, and the protocol stack. The operating system at each node is mainly responsible for resource management tasks such as file and memory management and local synchronization among local tasks. T distributed operating system, on the other hand, aims to provide global resource management, synchronization, and services to the users so that the users are not aware of the location of the service.

Instead of designing and implementing a distributed operating system from scratch, its tasks are usually handled by special software modules called the middleware targeting at the specific task at hand. The middleware layer is between the local operating system and the application software, and a software module in this layer performs at specific function that may be required by a number of applications[5]. For example, a synchronizer is a middleware module that provides synchronization among application level processes, and any application that needs synchronization may use this module by invoking its interface routines.

The protocol stack is responsible for the correct and timely delivery of messages between the nodes of the distributed system. Distributed systems do not have a common clock and therefore require synchronization at the hardware, operating system, middleware, or the application (algorithm) level. Synchronization is key to the correct coordination of the distributed algorithms. In general, there is no shared memory in a distributed system; therefore, all data transfers must be performed by message passing between then nodes.

Design Issues

Design issues and challenges in a distributed system may be broadly classified as in the area of system software and the distributed algorithms. Communication, synchronization, and the security problems are the key issues in the system software development side. Problems to be solved in distributed algorithms are numerous ranging from fault tolerance algorithms to load balancing to leader election in distributed

cooperating and synchronizing by other distribute algorithms running at other nodes of the distributed system to achieve a common goal. A symmetric distributed algorithm is executed on all nodes of the distributed system, whereas nodes may be running different components of an asymmetric distributed algorithm

Synchronization

A fundamental problem in a distributed system is time synchronization, which aims at keeping the clocks of the nodes of the system in synchrony. As in a single processor system, access to share resources must be monitored. In this so-called mutual exclusion problem, a number of algorithms were developed to provide mutual exclusion in distributed systems. Deadlocks in distributed systems may occur as in a single-processor system where nodes of the distributed system wait for each other indefinitely, and no progress can be achieved. Precautions should be taken to prevent deadlocks. The analysis of distributed algorithms should provide proofs of deadlock-free executions. Leader elections another common problem where it is required to elect one of the all nodes or a group of nodes in the system to perform special tasks.

Load Balancing

It is a general requirement to distribute load that consists of code and data evenly to the nodes of the distributed system. The code and data of process may need to be migrated from a heavily loaded node to a node with less load. The response time, which is the time taken from registering the input to providing a response to it, and throughput, which is the number of tasks finished in a given time, are two important metric of performance in aadistributed system. Load balancing aims to reduce or average response time and increase throughput in a distributed system.

While balancing the load, real-time, requirements of the task should also be considered. A hard real-time task, such as a military application or a process control task requires to be executed before a given deadline, and failure to do so may result in irreversible losses, whereas missing deadlines in a soft read-time system such as a banking system results in degraded performance.

Fault Tolerance

The aim of fault tolerance in distributed systems is to handle fault such as the crash of a computing node or a line connecting two nodes or a software module running at a node. Tolerance of faults is imperative in applications such as plant control or military applications. One way of achieving fault tolerance is by replicating code and data so that the replica may continue to work in the case of faults. The correct nodes reach agreement using consensus algorithms, which is another area of research in fault tolerant computing. Check-pointing and recovery procedures record the state of the software periodically on a secondary storage, and in case of faults, the system may be started from the last recorded state. These algorithms require significant synchronization in distributed systems.

Self-stabilizing algorithms aim at reaching a stable state in the presence of faults starting from any arbitrary initial condition. These algorithms should achieve a stable state in a bounded number of steps.

From Grids to the Future Internet

The transportation of massive amounts of data is likely to congest current inks (or certain critical points) for very longperiods without necessarily guaranteeing an effective and correct transfer of all the data. Certain sessions that are too long can fail and simultaneously prevent the circulation of short and urgent messages. Communications taking place on a computing grid also intensively use reliable data distribution and collective-operations mechanisms that we do not know how to efficiently process with traditional point-to-point protocols. This is added to by the instability of the load and availability. For example, a user that requires a large number of computational resources might have to contact several different resource providers in order to satisfy her requirements. When the pool of resources is finally delivered, it is often heterogeneous, making the task of performance profiling and efficient use of the resources difficult. While some users have the expertise required to exploit resource heterogeneity, many prefer an environment where resource hardware, software stacks, and programming environments are uniform. Such uniformity makes the task of

large-scale application development and deployment more accessible.

Recently, a number of systems have arisen that attempt to convert what is essentially a manual large-scale resource provisioning and programming problem into a more abstract notion commonly referred to as elastic, utility, or cloud computing (we use the term "cloud computing" to refer to these systems in the remainder of this work). As the number and scale of cloud-computing systems continues to grow, significant study is required to determine directions we can pursue toward the goal of making
future cloud computing platforms successful.

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort. Cloud providers typically use a "pay as you go" model. This can lead to unexpectedly high charges if administrators do not adapt to the cloud pricing model.The present availability of high-capacity networks, low-cost computers and storage devices as well as the widespread adoption of hardware virtualization, service-oriented architecture, and autonomic and utility computing have led to a growth in cloud computing. Cloud computing has now become a highly demanded service or utility due to the advantages of high computing power, cheap cost of services, high performance, scalability, accessibility as well as availability.

Cloud computing adopts concepts from Service-oriented Architecture (SOA) that can help the user break these problems into services that can be integrated to provide a solution. Cloud computing provides all of its resources as services, and makes use of the well-established standards and best practices gained in the domain of SOA to allow global and easy access to cloud services in a standardized way.

III RELATED WORKS

Cloud computing providers have setup several data centers at different geographical locations over the Internet in order to optimally serve needs of their customers around the world. However, existing systems do not support mechanisms and policies for dynamically coordinating load distribution among different Cloud-based data centers in order to determine optimal location for hosting application services to achieve reasonable QoS levels. Further, the Cloud computing providers are unable to predict geographic distribution of users consuming their services, hence the load coordination must happen automatically, and distribution of services must change in response to changes in the load.

An entity a to an entity b is allowed only if the security class of b is equal to or higher than a. The performance is improved by protecting user's data integrity and secrecy in Decentralized IFC. Decentralized IFC model was designed to meet the changing needs of systems from global, static, hierarchical security levels to a more fluid system. It is able to capture the needs of different application [2]. in the load.

Cloud computing is a kind of grid computing; it has evolved by addressing the QoS (quality of service) and reliability problems. Cloud computing provides the tools and technologies to build data/compute intensive parallel applications with much more affordable prices compared to traditional parallel computing techniques.

3.1.Service Models

Service-oriented architecture advocates "everything as a service" (with the acronyms EaaS or XaaS or simply aas), cloud-computing providers offer their "services" according to different models, which happen to form a stack: infrastructure-, platform- and software-as-a-service.

3.1.1.Infrastructure As A Service (IAAS)

In the most basic cloud-service model and according to the IETF (Internet Engineering Task Force) providers of IaaS offer computers physical or (more often) virtual machines and other resources. IaaS refers to online services that abstract user from the detail of infrastructure like physical computing resources, location, data partitioning, scaling, security, backup etc. A hypervisors, such as Xen, Oracle Virtual Box, KVM, VMware ESX/ ESXi, or Hyper-V runs the virtual machines as guests. Pools of hypervisors within the cloud operational system can support

large numbers of virtual machines and the ability to scale services up and down according to customers' varying requirements. IaaS clouds often offer additional resources such as a virtual-machine disk-image library, raw block storage, file or object storage, firewalls, load balancers, IP addresses, virtual local area networks (VLANs), and software bundles. IaaS-cloud providers supply these resources on-demand from their large pools of equipment installed in data centers. For wide-area connectivity, customers can use either the Internet or carrier clouds. To deploy their applications, cloud users install operating-system images and their application software on the cloud infrastructure. In this model, the cloud user patches and maintains the operating systems and the application software.

3.1.2.**Platform As A Service (PAAS)**

Cloud providers typically bill IaaS services on a utility computing basis: cost reflects the amount of resources allocated and consumed PaaS vendors offer a development environment to application developers. The provider typically develops toolkit and standards for development and channels for distribution and payment. 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. With some PaaS offers like Microsoft Azure and Google App Engine, the underlying computer and storage resources scale automatically to match application demand so that the cloud user does not have to allocate resources manually. The latter has also been proposed by an architecture aiming to facilitate real-time in cloud environments. They need quotation to verify even more specific application types can be provided via PaaS, such as media encoding as provided by services like bitcodin.com or media.

3.1.3.Software As A Service (SAAS)

The users gain access to application software and databases. Cloud providers manage the infrastructure and platforms that run the applications. SaaS is sometimes referred to as "on-demand software" and is usually priced on a pay-per-use basis or using a subscription fee. In the SaaS model, cloud providers install and operate application software in the cloud and cloud users access the software from cloud clients. Cloud users do not manage the cloud infrastructure and platform where the application runs. This eliminates the need to install and run the application on the cloud user's own computers, which simplifies maintenance and support.

3.2 **Cloud Computing Types**

3.2.1.Private Cloud

Private cloud is cloud infrastructure operated solely for a single organization, whether managed internally or by a third-party, and hosted either internally or externally. Undertaking a private cloud project requires a significant level and degree of engagement to virtualize the business environment, and requires the organization to reevaluate decisions about existing resources. When done right, it can improve business, but every step in the project raises security issues that must be addressed to prevent serious vulnerabilities. Self-run data centers are generally capital intensive.

3.2.2. Public Cloud

A cloud is called a "public cloud" when the services are rendered over a network that is open for public use. Public cloud services may be free. Technically there may be little or Decentralized access control with anonymous authentication of data stored in clouds Dept of ISE,YIT,Moodbidri Page 6 no difference between public and private cloud architecture, security consideration may be substantially different for services that are made available by a service provider for a public audience and when communication is effected over a non-trusted network.

3.2.3.Hybrid Cloud

Hybrid cloud is a composition of two or more clouds that remain distinct entities but are bound together, offering the benefits of multiple deployment models. Hybrid cloud can also mean the ability to connect collocation, managed and/or dedicated services with cloud resources. A hybrid cloud service crosses isolation and provider boundaries so that it can't be simply put in one

category of private, public, or community cloud service[15]. It allows one to extend either the capacity or the capability of a cloud service, by aggregation, integration or customization with another cloud service.

3.4.DECENTRALIZED ACCESS CONTROL WITH ANONYMOUS AUTHENTICATION OF DATA STORED IN CLOUDS

Data Security for Cloud Environment with Semi-Trusted Third Party (DaSCE) protocol provides key management, access control, file assured deletion for cloud computing.. DaSCE scheme enhances the security of keys and authentication process [4]. For overcoming uncertainty and avoiding potential risks a new flexible method i.e Trust & reputation based access control can be used. In Trust & reputation based access control method trust evaluated by the data owner and/or reputations generated by a number of reputation centers in a flexible manner by applying Attribute-Based Encryption (ABE) and Proxy Re-Encryption(PRE)[16]. This scheme is one of the first scheme to flexibly control cloud data access in an efficient way by integrating the concept of trust and reputation evaluation into a cryptographic system [5]

KDC provide access to particular fields in all records. Single keys Decentralized access control with anonymous authentication of data stored in clouds separates the data and the data owners, using this technique the user own the data by having the attribute it had, and this can be retrieved only if the attribute matches the data. The Author apply the attribute based encryption (ABE) based on bilinear pairings on elliptic curves. This scheme is collusion secure in which two users cannot together decode any data, that no one has individual right to access.

Attribute-Based Signatures: an Attribute based Signature in which the signature attests not to identify the individual of the message by a user instead it claim regarding the attribute that produced by the user. The signature was produced by a single party whose attributes satisfy the claim being made i.e. it is not colluding the all individuals instead it just make the attribute together who pooled it[18]. The author explains the security requirements of ABS as a cryptographic primitive, and then tells that efficient ABS construction based on groups with bilinear pairings. Thus by proving the construction is secure in the generic group model, ABS fills a critical security requirement in attribute-based messaging (ABM) systems. A powerful feature of ABS construction is that unlike many other attribute based cryptographic primitives, it can be readily used in a multi-authority setting, wherein users can make claims involving combinations of attributes issued by independent and mutually distrusting authorities.

Cipher text-Policy Attribute-Based Encryption : In certain distributed system the user can access the data only if the data consist of credential or attributes. Only way of enforcing such data in Cloud can be performed through the trusted server to store the data and accessing the cloud. In this paper the complex access control on the encrypted data is performed in which the Cipher text policy Attribute-Based Encryption is used. By using this scheme the storage data can be kept confidential even when the storage is untrusted, and this method secures against the collusion attack. The Previous Attribute Based Encryption systems used attributes to describe the encrypted data and even to build policies into user's keys; while in our system attributes are used to describe a user's credentials, and a party encrypting data determines a policy for who can decrypt.

Fine grained, system-wide, end-to-end, flow control is provided & allows security contexts to be defined system-wide and guarantees non-interference between them. A system is secure in the (Information flow control)IFC model if and only if all allowed messages are safe, all allowed label changes are safe and all privilege delegation is safe [3].

.

Abstract

- We propose a new decentralized access control scheme for secure data storage in clouds, that supports anonymous authentication.

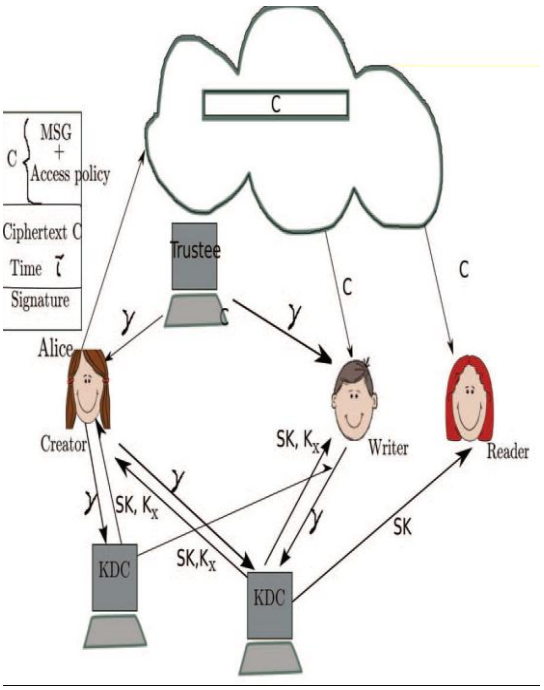
- In the proposed scheme, the cloud verifies the authenticity of the server without knowing the user's identity before storing data.

The scheme prevents replay attacks and supports creation, modification, and reading data stored in the cloud. We also address user revocation

Introduction

- The main aim of this project is that the identity of the user is protected from the cloud during authentication.
- Revoked users cannot access data after they have been revoked.
- To avoid replay attacks in the cloud.
- And also improve the security to the cloud data.

3.5.SYSTEM ARCHITECTURE



User privacy is also required so that the cloud or other users do not know the identity of the user.

The problem here is that the data records should have keywords associated with them to enable the search.

The correct records are returned only when searched with the exact keywords.

3.5.1.Disadvantages of Existing System

Authentication not support in existing schemes.

A single KDC is not only a single point of failure but difficult to maintain because of the large number of users that are supported in a cloud environment

3.5.2.Proposed System

Our proposed system enables to authenticate the validity of the message without revealing the identity of the user who has stored information in the cloud. We use attribute based signature scheme to achieve authenticity and privacy.

3.5.3.Advantages of Proposed System

The proposed scheme is resilient to replay attacks.

The costs are comparable to the existing centralized approaches, and the expensive operations are mostly done by the cloud.

3.5.4System Configuration

3.5.4.1.Software Requirements:

Operating System : Windows95/98/2000/XP
Front End : Swings & AWT
Scripts : JavaScript.
Database : Mysql
Database Connectivity : JDBC.

3.5.4.2.Hardware Requirements:

Speed -1.1 Ghz
RAM -256 MB (min)
Hard Disk - 20 GB
Floppy Drive -1.44 MB
Key Board - Standard Windows Keyboard
Mouse -Two or Three Button Mouse
Monitor -SVGA

3.5.5.Implementation

We have 3 modules,

1. User Registration.
2. Attribute generation.
3. Verify.

Module Description

3.5.5.1.User Registration:

In this module, User get registered because if user wants to upload and download a file he must login into the cloud. Then only the cloud server will check the user authentication. And also provide access policies after verifying the user authentication details.

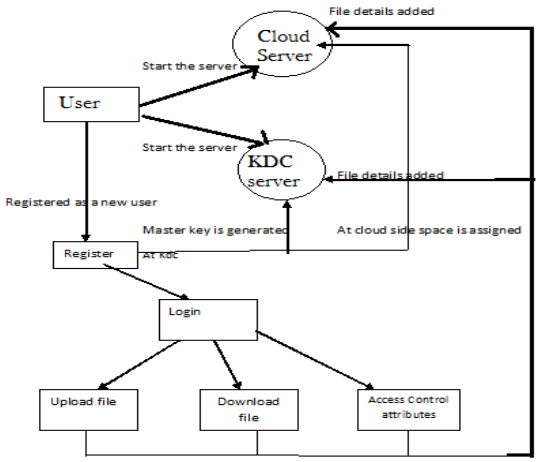
3.5.5.2.Attribute Generation:

In this module, we create the attributes for registered user to verify the tokens when they are share the data in the cloud.

3.5.5.3.Verify Module:

The verification process to the cloud, it relieves the individual users from time consuming verifications. When a reader wants to read some data stored in the cloud, it tries to decrypt it using the secret keys it receives from the KDCS

4.Data Flow Diagram

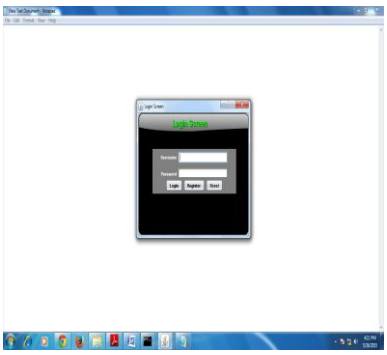


5.SCREENS

Cloud server screen

KDC screen

User screen



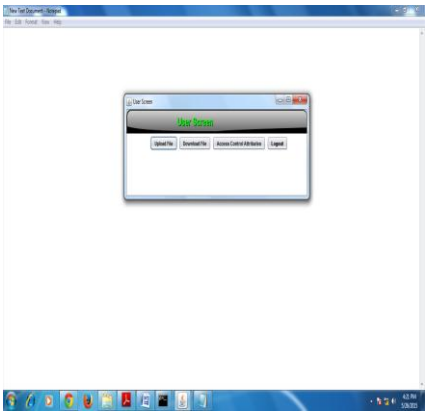
Registration screen

Completion of registration

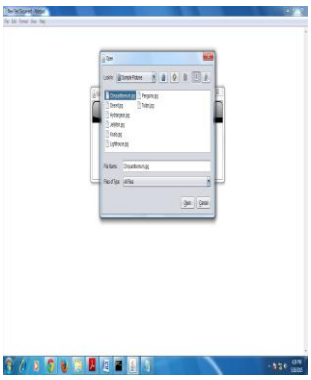
After successful registration

User Login screen

User home screen

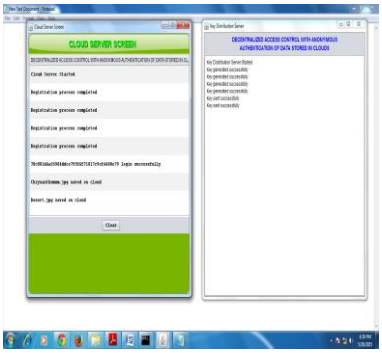


Upload files

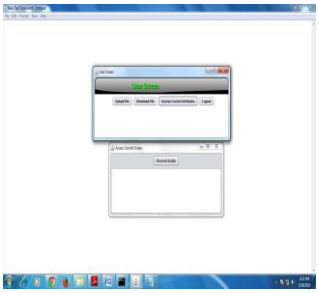


After upload file

After upload file server screen

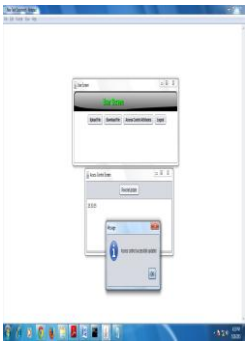


Creating access policies

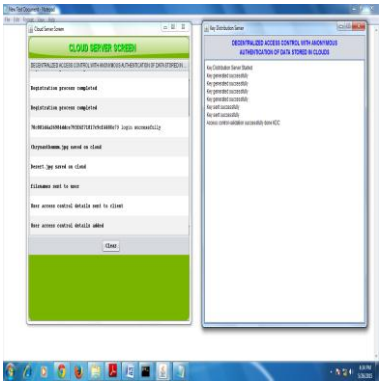


Revoke/update screen

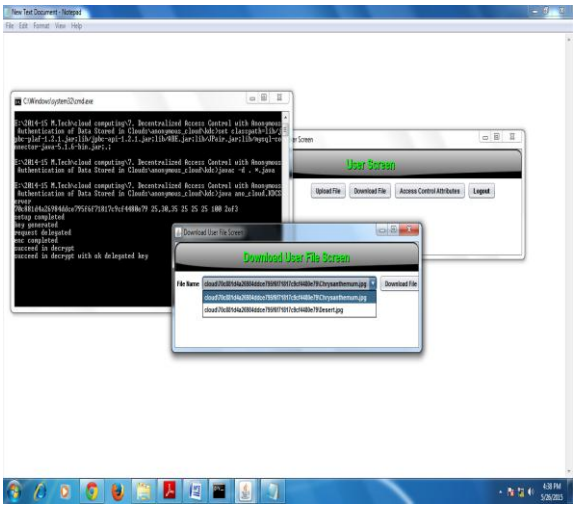
After creating access policy



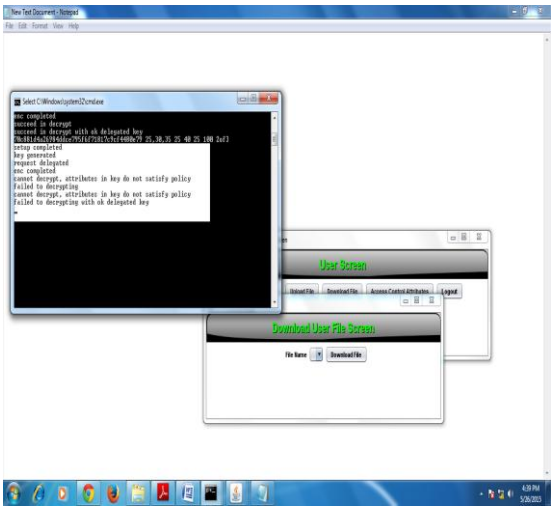
Access policies assigning to users



Who have access policies that user access the data



Who don't have access policies that user failed to decrypt the data



6.CONCLUSION

We have presented a decentralized access control technique with anonymous authentication, which provides user revocation and prevents replay attacks. The cloud does not know the identity of the user who stores information, but only verifies the user's credentials. Key distribution is done in a decentralized way. One limitation is that the cloud knows the access policy for each record stored in the cloud.

REFERENCES:

1. Ting Yu, Niresh V. Chawla and Sinees Sinoff, "Computational Intelligent Data Analysis for Sustainable Development", Chepnes & Hall/CRC Press, ISBN: 9778-1-4398-9595-5, 2013.

2. Shao Ying Zhu, Richard Hill, Macello Trouati, "Guide to Security Accurate for Cloud Computing", Springer, ISN: 978-3-319-25988-8,2015.

3. Pascale Vicat-Blose, Sebesties Soudas, Romanic Guillier, Brice Goglis, "Compute Network, From Cluster to Cloud Computing", Johnwilles Sons, ISN; 978-1-84821-286-2, 2011.

4. Gabriel Masricks, "Instant iqgrid", Pack publishing, ISBN 978-1-78928-991-2, 2013

5. Dabu Panda, Arvind Maheshwari," Middle Ware Management with Oracle Enter prize Manager Grid Control", 10gR5, Packb Publishing, ISBN: 978-1-847198-34-1, 2009.

6. Sherali Zeadally, Mohamed Badra, "Privacy is a difficult, Networked world", Springer, ISSBN; 978—3-319-08469-5, 2015

7. Guido Schnutz, Daniel Liebhart, Peter Welkerback, "Service Oriented Architecture: An Integration Blue Print", ISBN: 978-1-849681-04-9, 2010.

8. Maozhes li, Mark Baker," The Grid Core Technologies", ISBN: 978-04-470-09417-4, 2005.

9. Andreas Weip, Dinke Karestoyanava, "Enabling Coupled Multi-scale, Multi-field experiments Through Chore graphics of Data-Drives Scientific Simulations", Institute of Architeture of Applications, Systems, Germany, Springer, 2014.

11. Mohammed Abdur Razzaque, Marija Mihojeuie, Andrei Pelade, "Middle ware for Internet of Theism: a Survey" DOI: 10,1109/JIOT, 315.2498900, 2015.

12. Edoardo Patti, Angeliti Lydia Ateria Syrri Marco Jals, Pierlvisi Mascerlle," Insrastructure for General Purpose Services is smart Grid", IEEE, DOI: 10.1109|TSG.2014-2375197.

13. Enzo Bacarelli, Nicole Cordeschi, A lessadra Mei, Massino Paralle, Mohammed Shoiafar, Jilisde Stefe, "Energy-efficient Dynamic Traffic off loading & Reconstitutes of Networked Data centers for Big data Stream Mobile Computing": Review, Challenges, and a Cese Stides. IEEE : Network, DOI 0890-8044|16,2016.

15. Energy Nikuleteu, Evgesiy Plozhrik, Eles Lukyeschikov, Dmity Biuryukov, "Features Management & Middleware of Hybrid Cloud Infrastructure", IJACSA, International Journal of Advanced Computer Science & Applications, Vol A, No. PP-31-36, 2016.

[16]SushmitaRuj, Member, Ieee, Milos Stojmenovic, Member, Ieee, And Amiya Nayak," Decentralized Access Control With Anonymous Authentication Of Data Stored In Clouds" Ieee Transactions On Parallel And Distributed Systems, Vol. 25, No. 2, February 2015

[17] S. Ruj, A. Nayak, and I. Stojmenovic, "DACC: Distributed access control in clouds," in IEEE TrustCom, 2011

[18] H. K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute based signatures: Achieving attributeprivacy and collusion resistance," IACR CryptologyePrint Archive, 2008.

[19] "Attribute-based signatures," in CT-RSA, ser. Lecture Notes in Computer Science, vol.6558. Springer, pp. 376–392, 2011.

[20] A. Beimel, "Secure Schemes for Secret Sharing and Key Distribution," PhD Thesis. Technion, Haifa, 1996.