DECENTRALIZED ACCESS CONTROL WITH ANONYMOUS AUTHENTICATION OF DATA STORED IN CLOUDS

PROJECT REFERENCE NO.: 39S_BE_1554

COLLEGE: YENEPOYA INSTITUTE OF TECHNOLOGY (FORMERLY DR.MVSIT), MOODABIDRI
BRANCH: DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING
GUIDE: MRS. MAMATHA SALIAN
STUDENTS: MS. MAITHRY
           MS. PRIYANKA
           MS. SMITHA M
           MS. TENIMOL THOMAS
ABSTRACT

A new decentralized access control scheme for secure data storage in clouds, which supports anonymous authentication. In the proposed scheme, the cloud verifies the authenticity of the user without knowing the user’s identity before storing the data. This has the added feature of access control in which only valid users are able to decrypt the stored information. The scheme prevents replay attacks and supports creation, modification and reading data stored in the cloud. And also addresses user revocation. It is a decentralized, robust and access control schemes designed for cloud. The communication, computation, and storage overheads are comparable to centralized approaches.
Chapter 1

INTRODUCTION

Cloud computing is a rising computing standard in which assets of the computing framework are given as a service over the Internet. As guaranteeing as it may be, this standard additionally delivers a lot of people new challenges for data security and access control when clients outsource sensitive data for offering on cloud servers, which are not inside the same trusted dominion as data possessors. In any case these results unavoidably present a substantial processing overhead on the data possessor for key distribution and data administration when fine-grained data access control is in demand, and subsequently don't scale well. The issue of at the same time accomplishing fine-graininess, scalability, and data confidentiality of access control really still remains uncertain. This issues can be solved by implementing access policies based on data qualities, and, then again, permitting the data owner to representative the majority of the calculation undertakings included in fine-grained data access control to un-trusted cloud servers without unveiling the underlying data substance

Research in cloud computing is receiving a lot of attention from both academic and industrial worlds. In cloud computing, users can outsource their computation and storage to servers using Internet. This frees users from the hassles of maintaining resources on-site. Clouds can provide several types of services like applications, infrastructures, and platforms to help developers write applications. Much of the data stored in clouds is highly sensitive, for example, medical records and social networks. Security and privacy are thus very important issues in cloud computing. In one hand, the user should authenticate itself before initiating any transaction, and on the other hand, it must be ensured that the cloud does not tamper with the data that is outsourced.

User privacy is also required so that the cloud or other users do not know the identity of the user. The cloud can hold the user accountable for the data it outsources, and likewise, the cloud is itself accountable for the services it provides. The validity of the user who stores the data is also verified. Apart from the technical solutions to ensure security and privacy, there is also a need for law enforcement. Cloud computing, also known as 'on-demand computing', is a kind of Internet-based computing, where shared resources, data and information are provided to computers and other devices on-demand. Cloud computing and
Storage solutions provide users and enterprises with various capabilities to store and process their data in third-party data centers. It relies on sharing of resources to achieve coherence and economies of scale similar to a utility. The foundation of cloud computing is the broader concept of converged infrastructure and shared services.

1.1 Cloud Computing

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. Proponents claim that cloud computing allows companies to avoid upfront infrastructure costs, and focus on projects that differentiate their businesses instead of on infrastructure. Proponents also claim that cloud computing allows enterprises to get their applications up and running faster, with improved manageability and less maintenance, and enables IT to more rapidly adjust resources to meet fluctuating and unpredictable business demand. 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.

Fig.1.1 Architecture of cloud computing
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. Companies can scale up as computing needs increase and then scale down again as demands decrease. 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 vendors are experiencing growth rates of 50% per annum. But due to being in a stage of infancy, it still has some pitfalls which need to be given proper attention to make cloud computing services more reliable and user friendly. Cloud computing is the result of the evolution and adoption of existing technologies and paradigms. The goal of cloud computing is to allow users to take benefit from all of these technologies, without the need for deep knowledge about or expertise with each one of them. The cloud aims to cut costs, and helps the users focus on their core business instead of being impeded by IT obstacles.

The main enabling technology for cloud computing is virtualization. Virtualization software separates a physical computing device into one or more "virtual" devices, each of which can be easily used and managed to perform computing tasks. With operating system–level virtualization essentially creating a scalable system of multiple independent computing devices, idle computing resources can be allocated and used more efficiently. Virtualization provides the agility required to speed up IT operations, and reduces cost by increasing infrastructure utilization. Autonomic computing automates the process through which the user can provision resources on-demand. By minimizing user involvement, automation speeds up the process, reduces labor costs and reduces the possibility of human errors.

Users routinely face difficult business problems. 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. Such metrics are at the core of the public cloud pay-per-use models. In addition, measured services are an essential part of the feedback loop in autonomic computing, allowing services to scale on-demand and to perform automatic failure recovery. 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.

1.2 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.

1.2.1 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.

1.2.2 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.

1.2.3 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.

1.3 Cloud Computing Types

1.3.1 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.

1.3.2 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
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.

1.3.3 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. It allows one to extend either the capacity or the capability of a cloud service, by aggregation, integration or customization with another cloud service.

1.4 Security in Cloud

According to the Cloud Security Alliance, the top three threats in the cloud are "Insecure Interfaces and API's", "Data Loss & Leakage", and "Hardware Failure". In a cloud provider platform being shared by different users there may be a possibility that information belonging to different customers resides on same data server. Information leakage may arise by mistake when information for one customer is given to other. Eugene Schultz, chief technology officer at Imagined Security, said that hackers are spending substantial time and effort looking for ways to as have this data be indexed by search engines (making the information public) penetrate the cloud. The data from hundreds or thousands of companies can be stored on large cloud servers, hackers can theoretically gain control of huge stores of information through a single attack. A process called "hyper jacking". Some examples of this include the Dropbox security breach, and Cloud 2014 leak. Dropbox had been breached in October 2014, having over 7 million of its user’s passwords stolen by hackers in an effort to get monetary value from it by Bit coins (BTC). By having these passwords, they are able to read private data as well.

There is the problem of legal ownership of the data. Physical control of the computer equipment (private cloud) is more secure than having the equipment off site and under someone else's control (public cloud). This delivers great incentive to public cloud computing service providers to prioritize building and maintaining strong management of
secure services. Some small businesses that don't have expertise in IT security could find that it's more secure for them to use a public cloud. There is the risk that end users don't understand the issues involved when signing on to a cloud service (persons sometimes don't read the many pages of the terms of service agreement, and just click "Accept" without reading). This is important now that cloud computing is becoming popular and required for some services to work, for example for an intelligent personal assistant.

Fundamentally private cloud is seen as more secure with higher levels of control for the owner, however public cloud is seen to be more flexible and requires less time and money investment from the user. Efficient search on encrypted data is also an important concern in clouds. The clouds should not know the query but should be able to return the records that satisfy the query. This is achieved by means of searchable encryption. The keywords are sent to the cloud encrypted, and the cloud returns the result without knowing the actual keyword for the search. 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.

Security and privacy protection in clouds are being explored by many researchers. Many holomorphic encryption techniques have been suggested to ensure that the cloud is not able to read the data while performing computations on them. Using holomorphic encryption, the cloud receives cipher text of the data and performs computations on the cipher text and returns the encoded value of the result. The user is able to decode the result, but the cloud does not know what data it has operated on. In such circumstances, it must be possible for the user to verify that the cloud returns correct results.

1.5 Access Control in Cloud

Access control in clouds is gaining attention because it is important that only authorized users have access to valid service. A huge amount of information is being stored in the cloud, and much of this is sensitive information. Care should be taken to ensure access control of this sensitive information which can often be related to health, important documents (as in Google Docs or Dropbox) or even personal information (as in social networking).

There are broadly three types of access control:

- User Based Access Control (UBAC),
Decentralized access control with anonymous authentication of data stored in clouds

- Role Based Access Control (RBAC),
- Attribute Based Access Control (ABAC).

In UBAC, the access control list (ACL) contains the list of users who are authorized to access data. This is not feasible in clouds where there are many users. In RBAC, users are classified based on their individual roles. Data can be accessed by users who have matching roles. The roles are defined by the system. For example, only faculty members and senior secretaries might have access to data but not the junior secretaries. ABAC is more extended in scope, in which users are given attributes, and the data has attached access policy. Only users with valid set of attributes, satisfying the access policy, can access the data. For instance, in the above example certain records might be accessible by faculty members with more than 10 years of research experience or by senior secretaries with more than 8 years’ experience. There has been some work on ABAC in clouds all these work use a cryptographic primitive known as Attribute Based Encryption (ABE). The extensible Access Control Markup Language (XACML) has been proposed for ABAC in clouds. An area where access control is widely being used is healthcare. Clouds are being used to store sensitive information about patients to enable access to medical professionals, hospital staff, researchers, and policy makers. It is important to control the access of data so that only authorized users can access the data.

Using ABE, the records are encrypted under some access policy and stored in the cloud. Users are given sets of attributes and corresponding keys. Only when the users have matching set of attributes, can they decrypt the information stored in the cloud. Access control is also gaining importance in online social networking where users (members) store their personal information, pictures, videos and share them with selected groups of users or communities they belong to. A colossal measure of data is constantly archived in the cloud, and much of this is sensitive data. Utilizing Attribute Based Encryption (ABE), the records are encrypted under a few access strategy furthermore saved in the cloud. Clients are given sets of traits and corresponding keys. Just when the clients have matching set of attributes, would they be able to decrypt the data saved in the cloud. Studied the access control in health care. Access control is likewise gaining imperativeness in online social networking where users store their personal data, pictures, films and shares them with selected group of users they belong. The work done by gives privacy preserving authenticated access control in cloud. Nonetheless, the researchers take a centralized methodology where a single key
distribution center (KDC) disperses secret keys and attributes to all clients. Unfortunately, a single KDC is not just a single point of failure however troublesome to uphold due to the vast number of clients that are upheld in a nature's domain. Multi-authority ABE principle was concentrated on in, which obliged no trusted power which requires each client to have characteristics from at all the KDCs. Access control in online social networking has been studied in such data are being stored in clouds. It is very important that only the authorized users are given access to those information. A similar situation arises when data is stored in clouds, for example in Dropbox, and shared with certain groups of people. It is just not enough to store the contents securely in the cloud but it might also be necessary to ensure anonymity of the user. For example, a user would like to store some sensitive information but does not want to be recognized. The user might want to post a comment on an article, but does not want his/her identity to be disclosed. The user should be able to prove to the other users that he/she is a valid user who stored the information without revealing the identity. There are cryptographic protocols like ring signatures, mesh signatures, group signatures, which can be used in these situations. Ring signature is not a feasible option for clouds where there are a large number of users. Group signatures assume the pre-existence of a group which might not be possible in clouds. Mesh signatures do not ensure if the message is from a single user or many users colluding together. For these reasons, a new protocol known as Attribute Based Signature (ABS) has been applied.

In ABS, users have a claim predicate associated with a message. The claim predicate helps to identify the user as an authorized one, without revealing its identity. Other users or the cloud can verify the user and the validity of the message stored. ABS can be combined with ABE to achieve authenticated access control without disclosing the identity of the user to the cloud. A key-policy (KP) ABE scheme that allows for threshold policies. Key-policy means that the encryption only gets to label a cipher text with a set of attributes. The authority chooses a policy for each user that determines which cipher texts he can decrypt. In a multi-authority ABE system, we have many attribute authorities, and many users. A user can choose to go to an attribute authority, prove that it is entitled to some of the attribute handled by that authority, and request the corresponding decryption keys. Any party can also choose to encrypt a message, in which case he uses the public parameters together with an attribute set of his choice to form the cipher text. Any user who has decryption keys corresponding to an appropriate attribute set can use them for decryption.
Chapter 2

PROBLEM STATEMENT

Existing work on access control in cloud are centralized in nature, and these schemes use Attribute based encryption (ABE). The centralized scheme uses a symmetric key approach and it does not support authentication. All the approaches take a centralized approach and allow only one KDC, which is a single point of failure. In a multi-authority ABE, in which there are several KDC authorities (coordinated by a trusted authority) which distribute attributes and secret keys to users. Multi authority ABE protocol required no trusted authority in which every user to have attributes from at all the KDCs.

In the Recently proposed fully decentralized ABE where users could have zero or more attributes from each authority and did not require a trusted server. In all these cases, decryption at user’s end is computation intensive. So, this technique might be inefficient when users access using their mobile devices. To get over this problem, they have outsource the decryption task to a proxy server, so that the user can compute with minimum resources. The presence of one proxy and one key distribution center makes it less robust than decentralized approaches. Both these approaches had no way to authenticate users, anonymously, users, who want to remain anonymous while accessing the cloud.

To ensure anonymous user authentication Attribute Based Signatures, it takes a decentralized approach and provides authentication without disclosing the identity of the users, as mentioned earlier in the previous section it is prone to replay attack. It provides privacy preserving authenticated access control in cloud. The authors take a centralized approach where a single key distribution center (KDC) distributes secret keys and attributes to all users.

Disadvantages of existing system:

- The scheme in uses asymmetric key approach and does not support authentication.
- Difficult to maintain because of the large number of users that are supported in a cloud environment.

To overcome these a new decentralized access control scheme for secure data storage in clouds that supports anonymous authentication is introduced.
Chapter 3

LITERATURE SURVEY

The Security of storage is not only enough to store the user must also check the anonymity of the user. For example the user wants to post a comment on Article but doesn’t want his/her to disclose. There are three cryptographic protocols such as Ring Signature, Mesh Signature Group Signature. The Ring Signature which mean a large number of users are been involved so it is not feasible. The Mesh Signature which does not ensure whether the message is from the single user or from a group user and it colludes the information. The Group Signature which is not possible because of the pre-existing in the group. For these kind of reasons a new protocol Attribute Based Signature is been introduced in which the users have claim predicate associate with the message. Hence the ABS is combined with ABE for the Authentication Access Control by not showing users identity in the Cloud.

The Author take a centralized approach in where the Single Key Distribution Center (KDC) which distributes the Secret key and the attributes to all user. In this the failure not only occurred at a Single point and it is also difficult to maintain by a large number of user in Cloud, finally the author decided to have a decentralized system to make the work less and to access by the user from various location from the world using many KDC’s. Yang et al proposed, this decentralized access must also need the technique of anonymous of accessing the cloud while authentication. In earlier Ruj et al proposed the distributed access Control and this does not provide the authentication. The other drawback is the Creator can only create and store the file. So that user who reads the Data can have only the read access and doesn’t had the permission to write in the data except by Creator.

3.1 Existing Systems

3.1.1 Dacc: Distributed Access Control In Clouds

S. Ruj, A. Nayak, and I. Stojmenovic[2], proposed a data storage and access in which the multiple encrypted copies of data can be avoided. The main novelty of this paper is producing the key distribution centers where one or more KDCs distribute keys to data owners and users. KDC provide access to particular fields in all records. Single keys
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.

### 3.1.2 Attribute-Based Signatures: Achieving Attribute-Privacy Collusion Resistance

H.K.Maji, M.Prabhakaran, and M. Rosulek[3], proposed 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. 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.

### 3.1.3 Secure and Efficient Access to Outsourced Data

W. Wang, Z. Li, R. Owens, and B. Bhargava[8], proposed by providing secure and efficient access to outsourced data should be must in cloud computing. To encrypt every data block with a different key the flexible cryptography-based access control is used. Through this key derivation methods, the owner should maintain only a few secrets in the storage, and this key derivation procedure is used in hash functions which will introduce very limited computation. Thus to use over-encryption and or lazy revocation to prevent revoked users from getting access to updated data blocks. A Mechanism is used to handle both updates to outsourced data and changes in user access rights. Hence it is investigated in the overhead and safety of the proposed approach an encryptor can choose, for each authority, a number.
do and a set of attributes. Thus this scheme tolerate an arbitrary number of corrupt authorities.

3.1.4 Secured Scheme For Secret Sharing And Key Distribution

A. Beimel[5], proposed the sharing of data, now a days take place in Computer Networks, and the data which is been communicated inside the network may affected through the bad users, to overcome this user users two Cryptographic tools such as Generalized Secret Sharing scheme and Key distribution scheme. This make it possible to store only the secret information in the network such that only good users can access the information, the secret sharing scheme mostly received through the threshold secret sharing schemes, only through the certain threshold the information can accessed and can used by the user. In generalized secret sharing it is capable of arbitrary monotone collection whereas in Key distribution scheme the keys can be used Communication key Distribution scheme does not help in unrestricted scheme on other hand secured and restricted scheme can be accessed only through limits. Linear Secret Sharing Scheme, Monotone Span programs, Secret sharing the public reconstruction computation function of shared secret keys are used.

3.1.5 Cipher text-Policy Attribute-Based Encryption

J. Bethencourt, A. Sahai, and B. Waters [6], proposed 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.

3.1.6 Multi-Authority Attribute Based Encryption

M. Chase[7], proposed identity based encryption the user use the identity to search the data whereas in attribute based encryption involves attribute to search the data. Sahai and water introduced a single authority attribute encryption scheme and left the question whether
the multiple authorities allowed to distribute system. This scheme allows any polynomial number of independent authorities to monitor attributes and distribute secret keys.

### 3.1.7 Privacy Preserving Access Control with Authentication for Securing Data in Clouds

S. Ruj, M. Stojmenovic and A. Nayak [9], proposed a new privacy preserving authenticated access control scheme for securing data in clouds. In the proposed scheme, the cloud verifies the authenticity of the user without knowing the user’s identity before storing information. Our scheme also has the added feature of access control in which only valid users are able to decrypt the stored information. The scheme prevents replay attacks and supports creation, modification, and reading data stored in the cloud. Moreover, our authentication and access control scheme is decentralized and robust, unlike other access control schemes designed for clouds which are centralized. The communication, computation, and storage overheads are comparable to centralized approaches.

They presented a privacy preserving access control scheme for clouds. Our scheme not only provides fine-grained access control but also authenticates users who store information in the cloud. The cloud however does not know the identity of the user who stores information, but only verify 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. In future, we would like to protect the privacy of user attributes as well.

### 3.1.8 Toward Secure and Dependable Storage Services in Cloud Computing

C. Wang, Q. Wan, K. Ren, N. Cao and W. Lou [10], proposed a Cloud storage which enables users to remotely store their data and enjoy the on demand high quality cloud applications without the burden of local hardware and software management. The benefits are clear, such a service is also relinquishing users’ physical possession of their outsourced data, which inevitably poses new security risks toward the correctness of the data in cloud. In order to address this new problem and further achieve a secure and dependable cloud storage service, we propose in this paper a flexible distributed storage integrity auditing mechanism, utilizing the homomorphic token and distributed erasure coded data. The proposed design allows users to audit the cloud storage with very lightweight communication and
computation cost. The auditing result not only ensures strong cloud storage correctness guarantee, but also simultaneously achieves fast data error localization, i.e., the identification of misbehaving server. Considering the cloud data are dynamic in nature, the proposed design further supports secure and efficient dynamic operations on outsourced data, including block modification, deletion, and append. Analysis shows the proposed scheme is highly efficient and resilient against Byzantine failure, malicious data modification attack, and even server colluding attacks.

To achieve the assurances of cloud data integrity and availability and enforce the quality of dependable cloud storage service for users. An effective and flexible distributed scheme with explicit dynamic data support, including block update, delete, and append. And rely on erasure-correcting code in the file distribution preparation to provide redundancy parity vectors and guarantee the data dependability. By utilizing the homomorphic token with distributed verification of erasure-coded data, our scheme achieves the integration of storage correctness insurance and data error localization, i.e., whenever data corruption has been detected during the storage correctness verification across the distributed servers, we can almost guarantee the simultaneous identification of the misbehaving server(s). Considering the time, computation resources, and even the related online burden of users, we also provide the extension of the proposed main scheme to support third-party auditing, where users can safely delegate the integrity checking tasks to third-party auditors and be worry-free to use the cloud storage services. Through detailed security and extensive experiment results, we show that our scheme is highly efficient and resilient to Byzantine failure, malicious data modification attack, and even server colluding attacks.

3.1.9 Fuzzy Keyword Search over Encrypted Data Using Cloud Computing

J.Li, C. Wang, Q. Wan, K. Ren, N. Cao and W. Lou [11], proposed aCloud computing technology that uses the internet and central remote servers to maintain data and applications. Cloud computing allows consumers and businesses to use applications without installation and access their personal files at any computer with internet access. This technology allows for much more efficient computing by centralizing storage, memory, processing and bandwidth. Perhaps the biggest concerns about cloud computing are security and privacy. If a client can log in from any location to access data and applications, it's possible the client's privacy could be compromised. There are many searching technique which were implemented in the cloud these technique supports only exact keyword search.
Chapter 4

PROPOSED SYSTEM

The proposed decentralized approach does not authenticate users, who want to remain anonymous while accessing the cloud. And uses attribute based signature scheme to achieve authenticity and privacy. Decentralized scheme has the added feature of access control in which only valid users are able to decrypt the stored information. It prevents replay attacks and supports creation, modification, and reading data stored in the cloud.

4.1 Objectives

- Distributed access control of data stored in cloud so that only authorized users with valid attributes can access them.
- Authentication of users who store and modify their data on the cloud.
- The identity of the user is protected from the cloud during authentication.
- The architecture is decentralized, meaning that there can be several KDCs for key management.
- The access control and authentication are both collusion resistant, meaning that no two users can collude and access data or authenticate themselves, if they are individually not authorized.
- The proposed scheme is resilient to replay attacks. A writer whose attributes and keys have been revoked cannot write back stale information.
- The protocol supports multiple read and write on the data stored in the cloud.
- The costs are comparable to the existing centralized approaches, and the expensive operations are mostly done by the cloud and modify appropriately. Our scheme also allows writing multiple times which was not permitted in our earlier work.

Advantages of proposed system:

- Distributed access control of data stored in cloud so that only authorized users with valid attributes can access them.
- Authentication of users who store and modify their data on the cloud.
- The identity of the user is protected from the cloud during authentication.
Chapter 5

SYSTEM REQUIREMENT SPECIFICATION

5.1 Introduction

A System Requirements Specification (SRS) is a complete description of the behavior of the system to be developed. SRS is a document that completely describes what the proposed software should do without describing how the software will do it. It is two-way policy that assures that both client and the organization understand the requirement at any given point of time. SRS document itself is precise and it provides the functions and capabilities of system that it should provide. It act as a blueprint for completing a project such that it is cost effective. It provides feedback to customer. An SRS is customer assurance that the development organization understands the issues or problems to be solved and the software behavior necessary to address those problems.

It serves as an input to design specification it also serves as the parent document to subsequent documents. Therefore, the SRS should be easy to understand and also should contain sufficient details in the system requirements so that a design solution can be devised easily.

SRS should have certain properties: A good SRS is:

- Correct: An SRS is correct if every requirement included in this document represents something that is required in the final system.
- Complete: An SRS is complete if the response of the software to all classes of input data are specified in the SRS.
- Unambiguous: An SRS is unambiguous if and only if every requirement stated has one and only one interpretation.
- Verifiable: An SRS is verifiable if and only if stated requirement is verifiable. A requirement is verifiable if there is some cost effective process that can check whether software meets the requirements.
- Modifiable: An SRS is modifiable if its structure and style are such that any necessary changes can be made easily preserving completeness and consistency.
5.2 Purpose of the Requirements Document

Requirements are the most important and crucial part of project. These are necessary during the development cycle of the project. The requirements defines the requirements for more features an existing project or it defines a new project. These are necessary during the development. Different projects will have different need and therefore different requirements are needed for the project so as to develop the project in the right way. The requirements document is needed to keep track of all the things which are needed in the development of the project.

The requirements are the basic need of project. The requirements make the project complete in every way. Requirements documents enables us to make use of every product at perfect time. It keeps record of needed ones for the development of the project till the successful completion of the project and after the completion of the project for viewing the details in the recorded files. Programmers often copy and paste code so that they can reuse the existing code to complete a similar task. Many times, modification to the newly pasted code include renaming all instances of an identifier such as a variable name, consistency throughout the fragment. To help programmers to detect the place where mistake occur due to copy pasting by detecting clones.

5.3 System Requirements

5.3.1 Hardware Requirements:

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Monitor : 15 VGA Color.
- Mouse : Logitech
- Ram : 512 Mb.

5.3.2 Software Requirements:

- Coding Language: JSP
- IDE : ECLIPSE
- Database : MYSQL
Chapter 6

SYSTEM DESIGN

6.1 Introduction

The system design aims to identify the modules that should be in system, the specification of these modules and their interaction with each other to produce the desired result. At the end of the system design, all the major data structured, file format, output format and the major modules in the system and their specification are decided. The design activity begins when the requirements documents for software to be developed is available, this may be the software requirements specification for the complete system. Design is the first step in the development phase of any engineered product or system. The goal of the design process is to produce a model or representation of a system. The design phase is a transition from a user oriented document to a document oriented to the programmers. This chapter covers the design process of the project. It mainly categorize this design into four parts i.e. High-level, user interface, database design and detailed design. In high level design, simple data flow diagrams and use case diagrams of the application.

6.2 System Architecture

In the proposed scheme, there are three users, a creator, a reader, and a writer. Creator Alice receives a token γ from the trustee, now it is assumed to be who is honest. SKs are secret keys given for decryption, KX are keys for signing. The message MSG is encrypted under the access policy X. The access policy decides who can access the data stored in the cloud. The creator define a claim policy Y to prove the authenticity and signs of the message under this claim. The cipher text C with a signature c is sent to the cloud. The cloud verifies the signature and stores the cipher text C. When a reader wants to read the message in the cloud sends C. That the user has attributes matching with the access policy, it can be decrypted and get back the original message.

Write also proceeds in the similar way as file creation. By designating the verification of the data 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 decrypting and using the secret keys it receives from the KDCs. If it has enough attributes matching with the
access policy, then it decrypts the information stored in the cloud. The secret keys it receives from the KDCs.

Fig.6.1 Secure cloud storage model

6.3 Use Case Diagram

This use case diagram depicts a static view of a system functions and their static relationship both with external entity and with one and another.

It consist of:

**System**: the system is depicted as rectangle.

**Actor**: each actor is shown as a arrow figure.

**Use case**: each use case is shown as a oval label with the name of the use case.
6.4 Sequence Diagram

Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a message sequence chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios. Sequence diagrams show, as parallel vertical lines (life lines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.
Decentralized access control with anonymous authentication of data stored in clouds

Fig. 6.3 Sequence diagram
6.5 Flow Chart

A flow chart is a type of diagram that represents an algorithm or process, showing the steps as boxes of various kinds, and their order by connecting these with arrows, this diagrammatic representation can give a step by step solution to a given problem. Process operations are represented in these boxes, and arrows connecting them represent flow of control. Flow charts are used in analyzing, designing, documenting or managing a process or program in various fields.

Fig.6.4 Flow chart
6.6 ER Diagram

An entity relationship diagram is a way of graphically representing the logical relationships of entities in order to create a database.

**Entities:**

It is a basic object that the ER model represents, like the entities we have considered are registration entity and decentralized entity.

**Attributes:**

They are the particular properties that describe an entity. Considering the registration table entity attributes are user name, password, role, id, etc.

**Relationships:**

There is ‘requests token’, ‘generate key’, ‘upload file’, ‘download file’ relationship between registration table and decentralized table.

![ER Diagram](image-url)

**Fig.6.5 ER diagram**
Chapter 7

SYSTEM IMPLEMENTATION

7.1 Implementation

Implementation is initiated after the system has been tested and has been accepted by the user. In this phase the system is installed to support the intended business function. System performance is compared to performance objectives established during the planning phase. Implementation include user notification, user training, installation of hardware, installation of software onto production computers and integration of the system into daily work processes. The phase continues until the system is operating in production in accordance with the defined user requirements. This phase focuses on the remaining levels of testing, such as acceptance, load, stress, performance and recovery testing. The application needs to be verified under the conditions with respect to the SRS. Various documents are uploaded and different matrices for testing are completed at this stage of the software testing life cycle.

Acceptance testing is the final stage of this testing that is performed on the system prior to the system prior to the system being delivered to a live environment. Acceptance tests are generally performed as a “black box” jet-black box testing means that the tester uses specified inputs into the system and verifies that the resulting outputs are correct, without knowledge of the system’s internal workings. Load testing is the process of subjecting the computer, peripheral, server, network or application to a work level approaching the limits of its specifications load testing can also be done in the field to obtain a qualitative idea of how well a system function in the “real world”. Stress testing is a fundamental quality assurance activity that should be part of every significant software testing effort. The key idea behind stress testing is simple instead of running manual or automated tests under normal condition, run your tests under condition of reduced machine or system resources. Performance testing is the process of determining the speed or effectiveness of the computer, network, software program or device.

7.2 Execution Steps

1. System Initialization.
2. User Registration.
3. KDC setup.
4. Attribute generation.
5. Sign.
6. Verify.

Fig. 7.1 Module diagram

7.3 Modules Description

1. System Initialization

Select a prime q, and groups G1 and G2, which are of order q. We define the mapping i.e.: \( G_1 \times G_1 \rightarrow G_2 \). Let \( g_1, g_2 \) be generators of \( G_1 \) and \( h_j \) be generators of \( G_2 \), for \( j \in [t_{\text{max}}] \),
for arbitrary $t_{\text{max}}$. Let $H$ be a hash function. Let $A_0 = ha_0 0$, where $a_0 \in \mathbb{Z}_q$ is chosen at random. $(TSig,TVer)$ mean $TSig$ is the private key with which a message is signed and $TVer$ is the public key used for verification. The secret key for the trustee is $TSK = (a_0, TSig)$ and public key is $TPK = (G_1,G_2,H, g_1,A_0, h_0, h_1, \ldots, h_{t_{\text{max}}}, g_2, TVer)$.  

2. User Registration

For a user with identity $U_u$ the KDC draws at random $K_{\text{base}} \in G$. Let $K_0 = K_1/a_0$ base. The following token $\gamma$ is output $\gamma = (u, K_{\text{base}}, K_0, \rho)$, where $\rho$ is signature on $u||K_{\text{base}}$ using the signing key $TSig$.

3. KDC Setup

We emphasize that clouds should take a decentralized approach while distributing secret keys and attributes to users. It is also quite natural for clouds to have many KDCs in different locations in the world. The architecture is decentralized, meaning that there can be several KDCs for key management.

4. Attribute Generation

The token verification algorithm verifies the signature contained in $\gamma$ using the signature verification key $TVer$ in $TPK$. This algorithm extracts $K_{\text{base}}$ from $\gamma$ using $(a, b)$ from $ASK[i]$ and computes $K_x = K_1/(a+bx)$ base, $x \in J[i, u]$. The key $K_x$ can be checked for consistency using algorithm $\text{ABS}$. Key Check($TPK, APK[i], \gamma, K_x$), which checks $\hat{e}(K_x, A_{ij}B_{xij}) = \hat{e}(K_{\text{base}}, h_j)$, for all $x \in J[i, u]$ and $j \in [t_{\text{max}}]$.

5. Sign

The access policy decides who can access the data stored in the cloud. The creator decides on a claim policy $Y$, to prove her authenticity and signs the message under this claim. The cipher text $C$ with signature is $c$, and is sent to the cloud. The cloud verifies the signature and stores the cipher text $C$. When a reader wants to read, the cloud sends $C$. If the user has attributes matching with access policy, it can decrypt and get back original message.

6. Verify

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.
Chapter 8

SYSTEM TESTING

8.1 Introduction

Testing is intended to be used throughout coding and testing phases of the project. This document outlines the procedures used for testing and verification of the code. Software testing is the process used to help identify the correctness, completeness, security and quality of developed computer software. This includes the process of executing the program or application with the intent of finding errors.

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

The starting point of the testing is unit testing where a module is tested separately and is often performed by the coder himself simultaneously along with the coding of the module e.g.: compose. After this, the modules are gradually integrated into subsystems, and integration testing is performed to detect design errors by focusing on testing the interconnection between modules e.g.: Individual forms. After the system is put together, the system testing is performed. Here the system is tested against the system requirement to see if all the requirements are met and if the system performs as specified by the requirements e.g.: A hard disk with minimum 1GB Available space for a full installation process.

8.2 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business
process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results. Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach
Field testing will be performed manually and functional tests will be written in detail.

Test objectives
- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested
- Verify that the entries are of the correct format
- No duplicate entries should be allowed

All links should take the user to the correct page

8.3 Integration Testing
Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components. Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.
8.4 Functional Test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.
Invalid Input : identified classes of invalid input must be rejected.
Functions : identified functions must be exercised.
Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

8.5 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

8.5.1 White Box Testing

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

8.5.2 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document. It is a testing in
which the software under test is treated, as a black box. You cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

### 8.6 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements. 

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

### 8.7 Test Report:

**Table 8.1 Testing for valid user name**

<table>
<thead>
<tr>
<th>Test case</th>
<th>Input</th>
<th>Test description</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User name starts with number</td>
<td>User name cannot start with number</td>
<td>Appropriate error message</td>
</tr>
<tr>
<td>2</td>
<td>User name is left blank</td>
<td>User name cannot be left blank</td>
<td>Enter username</td>
</tr>
</tbody>
</table>

**Table 8.2 Testing for valid password**

<table>
<thead>
<tr>
<th>Test case</th>
<th>Input</th>
<th>Test description</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Password is left blank</td>
<td>Password cannot be blank</td>
<td>Enter password</td>
</tr>
<tr>
<td>2</td>
<td>Password less than 8 characters</td>
<td>Password should be greater than 8</td>
<td>Incorrect password</td>
</tr>
<tr>
<td>3</td>
<td>Invalid password entered</td>
<td>Valid password must be entered</td>
<td>Password mismatch</td>
</tr>
</tbody>
</table>
Table 8.3 Testing for valid Email address

<table>
<thead>
<tr>
<th>Test case</th>
<th>Input</th>
<th>Test description</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Email address with an @ symbol</td>
<td>Email address cannot have @ symbol</td>
<td>Appropriate error message</td>
</tr>
<tr>
<td>2</td>
<td>Email address with space</td>
<td>Email address cannot have space</td>
<td>Appropriate error message</td>
</tr>
<tr>
<td>3</td>
<td>Email address without above faults</td>
<td>Valid email address</td>
<td>Email address accepted</td>
</tr>
</tbody>
</table>
Chapter 9

SNAPSHOTS

Fig. 9.1 Registering into the system

Fig. 9.2 Getting secret key
Fig. 9.3 Login of user

Fig. 9.4 Incorrect login
Decentralized access control with anonymous authentication of data stored in clouds

Fig. 9.5 Correct login

Fig. 9.6 Sharing of file between users
Decentralized access control with anonymous authentication of data stored in clouds

Fig. 9.7 Sharing file

Fig. 9.8 Checking shared file
Decentralized access control with anonymous authentication of data stored in clouds

![Decryption by Receiver](image1)

Fig. 9.9 Decryption by receiver

![Signature Verification](image2)

Fig. 9.10 Signature verification
Decentralized access control with anonymous authentication of data stored in clouds

Fig. 9.11 Recovering the key

Fig. 9.12 Searching for the file
Decentralized access control with anonymous authentication of data stored in clouds

Fig. 9.13 Incorrect authentication

Fig. 9.14 Signature verification
Decentralized access control with anonymous authentication of data stored in clouds

Fig. 9.15 Searching file in cloud

Fig. 9.16 Asking for cipher key
Decentralized access control with anonymous authentication of data stored in clouds

Decryption by Receiver:

Cipher Key
C1

Secret Key

Private Key

Encrypt

Not a Authenticated

Fig. 9.17 Not authenticated

Upload File

Choose File 4.jpg

upload

Fig. 9.18 Uploading file
Decentralized access control with anonymous authentication of data stored in clouds

Fig. 9.19 Uploaded file

Fig. 9.20 Download file
CONCLUSION

The decentralized access control with anonymous authentication provides a secure cloud storage in which the files are associated with file access policies that used to access the files placed on the cloud. Uploading and downloading of a file to a cloud with standard Encryption or Decryption. It is a Decentralized access of system in which every system have the access control of data. The Cloud which is a Secured storage area where the anonymous authentication is used, so that only the permitted users can be accessed. Decrypting of data can be viewed only by a valid users and can also store information only by valid users. This Scheme prevents Replay attack which mean Eaves Dropping can be avoided, Support Creation of data inside storage, Modifying the data by unknown users and Reading data stored in Cloud. The authentication and accessing the Cloud is Robust, Hence Overall Communication Storage area been developed by comparing to the Centralized approaches.
FUTURE WORK

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. In future, we would like to hide the attributes and access policy of a user. In future the file access policy can be implemented with Multi Authority based Attribute based Encryption. Using the technique it can avoid the number of wrong hits during authentication. Create a random delay for authentication, so the hacker can confuse to identify the algorithm.
REFERENCES


