Public Integrity Auditing for Shared Dynamic Cloud Data with Group User Revocation

K. Prasanna Kumari¹, Sk. Shabaaz², Dr. K. Rama Krishniah³, D Sarath Babu⁴

¹M.Tech Student, Dept of CSE, R K College of Engineering, Krishna-521456, A. P., India
²Assistant Professor, Dept of CSE, R K College of Engineering, Krishna-521456, A. P., India
³Professor & Principal Dept of CSE, R K College of Engineering, Krishna-521456, A. P., India
⁴Assistant Professor, Dept of CSE, N V R College of Engineering, Krishna-522201, A. P., India
E-Mail: prassu5k@gmail.com

ABSTRACT:
The advent of the cloud computing makes storage outsourcing becomes a rising trend, which promotes the secure remote data auditing a hot topic that appeared in the research literature. Recently some research considers the problem of secure and efficient public data integrity auditing for shared dynamic data. However, these schemes are still not secure against the collusion of cloud storage server and revoked group users during user revocation in practical cloud storage system. In this paper, we figure out the collusion attack in the exiting scheme and provide an efficient public integrity auditing scheme with secure group user revocation based on vector commitment and verifier-local revocation group signature. We design a concrete scheme based on the our scheme definition. Our scheme chains the public checking and efficient user revocation and also some nice properties, such as confidently, efficiency, count ability and traceability of secure group user revocation. Finally, the security and experimental analysis show that compared with its relevant schemes our scheme is also secure and efficient.

KEYWORDS: Integrity auditing, dynamic data, vector commitment, group signature, cloud computing

INTRODUCTION:
Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the common use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation. Cloud computing consists of hardware and software resources made available on the Internet as managed third-party services. These services typically provide access to advanced software applications and high-end networks of server computers.

Structure of cloud computing
How Cloud Computing Works?

The goal of cloud computing is to apply traditional supercomputing, or high-performance computing power, normally used by military and research facilities, to perform tens of trillions of computations per second, in consumer-oriented applications such as financial portfolios, to deliver personalized information, to provide data storage or to power large, immersive computer games. The cloud computing uses networks of large groups of servers typically running low-cost consumer PC technology with specialized connections to spread data-processing chores across them. This shared IT infrastructure contains large pools of systems that are linked together. Often, virtualization techniques are used to maximize the power of cloud computing.

Characteristics and Services Models:

The salient characteristics of cloud computing based on the definitions provided by the National Institute of Standards and Terminology (NIST) are outlined below:

On-demand self-service: A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service’s provider.

Broad network access: Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).

Resource pooling: The provider’s computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location-independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or data center). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.

Rapid elasticity: Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.

Measured service: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be managed, controlled, and reported providing transparency for both the provider and consumer of the utilized service.

Characteristics of cloud computing

Services Models:
Cloud Computing comprises three different service models, namely Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). The three service models or layer are completed by an end user layer that encapsulates the end user perspective on cloud services. The model is shown in figure below. If a cloud user accesses services on the infrastructure layer, for instance, she can run her own applications on the resources of a cloud infrastructure and remain responsible for the support, maintenance, and security of these applications herself. If she accesses a service on the application layer, these tasks are normally taken care of by the cloud service provider.

Benefits of cloud computing:

1. **Achieve economies of scale** – increase volume output or productivity with fewer people. Your cost per unit, project or product plummets.

2. **Reduce spending on technology infrastructure**. Maintain easy access to your information with minimal upfront spending. Pay as you go (weekly, quarterly or yearly), based on demand.

3. **Globalize your workforce on the cheap**. People worldwide can access the cloud, provided they have an Internet connection.

4. **Streamline processes**. Get more work done in less time with less people.

5. **Reduce capital costs**. There’s no need to spend big money on hardware, software or licensing fees.

6. **Improve accessibility**. You have access anytime, anywhere, making your life so much easier!

7. **Monitor projects more effectively**. Stay within budget and ahead of completion cycle times.

8. **Less personnel training is needed**. It takes fewer people to do more work on a cloud, with a minimal learning curve on hardware and software issues.

9. **Minimize licensing new software**. Stretch and grow without the need to buy expensive software licenses or programs.

10. **Improve flexibility**. You can change direction without serious “people” or “financial” issues at stake.

Advantages:

1. **Price**: Pay for only the resources used.

2. **Security**: Cloud instances are isolated in the network from other instances for improved security.

3. **Performance**: Instances can be added instantly for improved performance. Clients have access to the total resources of the Cloud’s core hardware.

4. **Scalability**: Auto-deploy cloud instances when needed.

5. **Uptime**: Uses multiple servers for maximum redundancies. In case of
server failure, instances can be automatically created on another server.

6. **Control:** Able to login from any location. Server snapshot and a software library lets you deploy custom instances.

7. **Traffic:** Deals with spike in traffic with quick deployment of additional instances to handle the load.

The development of cloud computing motivates enterprises and organizations to outsource their data to third-party cloud service providers (CSPs), which will improve the storage limitation of resource constrain local devices. Recently, some commercial cloud storage services, such as the simple storage service (S3) on-line data backup services of Amazon and some practical cloud based software Google Drive, Drop box, Mozy, Bitcasa, and Memo pal, have been built for cloud application. Since the cloud servers may return an invalid result in some cases, such as server hardware/software failure, human maintenance and malicious attack, new forms of assurance of data integrity and accessibility are required to protect the security and privacy of cloud user’s data. To overcome the above critical security challenge of today’s cloud storage services, simple replication and protocols like Rabin’s data dispersion scheme are far from practical application. The formers are not practical because a recent IDC report suggests that data-generation is outpacing storage availability. The later protocols ensure the availability of data when a quorum of repositories, such as k-out-of-n of shared data, is given. However, they do not provide assurances about the availability of each repositories, this will limit the assurance that the protocols can provide to relying parties. For providing the integrity and availability of remote cloud store, some solutions and their Variants, have been proposed. In these solutions, when a scheme supports data modification, we call it dynamic scheme, otherwise static one (or limited dynamic scheme, if a scheme could only efficiently support some specified operation, such as append). A scheme is publicly verifiable means that the data integrity check can be performed not only by data owners, but also by any third-party auditor. However, the dynamic schemes above focus on the cases where there is a data owner and only the data owner could modify the data.

**Cloud server:** In the first module, we design our system with Cloud Server, where the datas are stored globally. Our mechanism, Oruta, should be designed to achieve following properties:

**Public Auditing:** A public verifier is able to publicly verify the integrity of shared data without retrieving the entire data from the cloud.

**Correctness:** A public verifier is able to correctly verify shared data integrity.

**Unforgeability:** Only a user in the group can generate valid verification metadata (i.e., signatures) on shared data.
Identity Privacy: A public verifier cannot distinguish the identity of the signer on each block in shared data during the process of auditing. of users in a group: the original user and a number of group users. The original user initially creates shared data in the cloud, and shares it with group users. Both the original user and group users are members of the group. Every member of the group is allowed to access and modify shared data. Shared data and its verification metadata (i.e., signatures) are both stored in the cloud server. A public verifier, such as a third party auditor providing expert data auditing services or a data user outside the group intending to utilize shared data, is able to publicly verify the integrity of shared data stored in the cloud server. Owner Registration: In this module an owner has to upload its files in a cloud server, he/she should register first. Then only he/she can be able to do it. For that he needs to fill the details in the registration form. These details are maintained in a database.

Owner Login: In this module, owners have to login, they should login by giving their email id and password.

User Registration: In this module if a user wants to access the data which is stored in a cloud, he/she should register their details first. These details are maintained in a Database.

User Login: If the user is an authorized user, he/she can download the file by using file id which has been stored by data owner when it was uploading.

Public verifier: When a public verifier wishes to check the integrity of shared data, it first sends an auditing challenge to the cloud server. After receiving the auditing challenge, the Cloud server responds to the public verifier with an auditing proof of the possession of shared data. Then, this public verifier checks the correctness of the entire data by verifying the correctness of the auditing proof. Essentially, the process of public auditing is a challenge and-response protocol between a public verifier and the cloud server.

Auditing Module: In this module, if a third party auditor TPA (maintainer of clouds) should register first. This system allows only cloud service providers. After third party auditor gets logged in, He/She can see how many data owners have uploaded their files into the cloud. Here we are providing TPA for maintaining clouds. We only consider how to audit the integrity of shared data in the cloud with static groups. It means the group is pre-defined before shared data is created in the cloud and the membership of users in the group is not changed during data sharing. The original user is responsible for deciding who is able to share her data before outsourcing data to the cloud. Another interesting problem is how to audit the integrity of shared data in the cloud with dynamic groups — a new user can be added into the group and an existing group member can be revoked during data sharing — while still preserving identity privacy.

SYSTEM MODEL:
Fig 1. The cloud storage model

In the cloud storage model as shown in Figure 1, there are three entities, namely the cloud storage server, group users and a Third Part Auditor (TPA). Group users consist of a data owner and a number of users who are authorized to access and modify the data by the data owner. The cloud storage server is semi-trusted, who provides data storage services for the group users. TPA could be any entity in the cloud, which will be able to conduct the data integrity of the shared data stored in the cloud server. In our system, the data owner could encrypt and upload its data to the remote cloud storage server. Also, he/she shares the privilege such as access and modify (compile and execute if necessary) to a number of group users. The TPA could efficiently verify the integrity of the shared data stored in the cloud server. In our system, the data owner could encrypt and upload its data to the remote cloud storage server. Also, he/she shares the privilege such as access and modify (compile and execute if necessary) to a number of group users. The TPA could efficiently verify the integrity of the shared data stored in the cloud server; even the data is frequently updated by the group users. The data owner is different from the other group users, he/she could securely revoke a group user when a group user is found malicious or the contract of the user is expired.

Our threat model considers two types of attacks:
1) An attacker outside the group (include the revoked group user cloud storage server) may obtain some knowledge of the plaintext of the data. Actually, this kind of attacker has to at lease break the security of the adopted group data encryption scheme.
2) The cloud storage server colludes with the revoked group users, and they want to provide an illegal data without being detected.

We aim to achieve:
The following security goals in our paper:
1) Security. A scheme is secure if for any database and any probabilistic polynomial time adversary, the adversary cannot convince a verifier to accept an invalid output.
2) Correctness. A scheme is correct if for any database and for any updated data m by a valid group user, the output of the verification by an honest cloud storage server is always the value m. Here, m is a cipher text if the scheme could efficiently support encrypted database.
3) Efficiency. A scheme is efficient if for any data, the computation and storage overhead invested By any client user must be independent of the size of the shared data.
4) Count ability. A scheme is countable, if for any data the TPA can provide a proof for this misbehavior,
When the dishonest cloud storage server has tampered with the database.
5) Traceability. We require that the data owner is able to trace the last user who updates the data (data item), when the data is generated by the
generation algorithm and every signature generated by the user is valid.

**Bilinear Groups:**
We review a few concepts related to bilinear maps, which follow the notation of [28]. Let $G_1$ and $G_2$ be two multiplicative cyclic groups of prime order $p$, $g_1$ is a generator of $G_1$ and $g_2$ is a generator of $G_2$. $\psi$ is an efficiently computable isomorphism from $G_2$ to $G_1$ with $\psi(g_2) = g_1$, and $e : G_1 \times G_2 \rightarrow G_T$ is a bilinear map with the following properties:
1) Computability: there exists an efficiently computable algorithm for computing map $e$;
2) Bilinearity: for all $u \in G_1$, $v \in G_2$ and $a, b \in \mathbb{Z}_p$, $e(ua, vb) = e(u, v)ab$;
3) Non-degeneracy: $e(g_1, g_2) \neq 1$.

**INPUT DESIGN:** The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple.

**OBJECTIVES:** Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

**OUTPUT DESIGN:** A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to
the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements. Select methods for presenting information. Create document, report, or other formats that contain information produced by the system. The output form of an information system should accomplish one or more of the following objectives. Convey information about past activities, current status or projections of the Future. Signal important events, opportunities, problems, or warnings. Trigger an action. Confirm an action.

**Group Signature with User Revocation**

We present the formal definition of group signatures with verifier-local revocation [27] as follows. A verifier-local group signature scheme is a collection of three polynomial-time algorithms \( \text{VLR.KeyGen, VLR.Sign, VLR.Verify} \), which behaves as follows:

- **VLR.KeyGen**(n). This randomized algorithm takes as input a parameter \( n \), the number of members of the group. It outputs a group public key \( \text{gpk} \), an \( n \)-element vector of user keys \( \text{gsk} = (\text{gsk}[1], \text{gsk}[2], ..., \text{gsk}[n]) \), and an \( n \)-element vector of user revocation tokens \( \text{grt} \), similarly indexed. **VLR.Sign**(gpk, gsk[i], M). This randomized algorithm takes as input the group public key gpk, a private key gsk[i], and a message \( M \in \{0, 1\}^* \), and returns a signature \( \sigma \).

- **VLR.Verify**(gpk, RL, \( \sigma, M \)). The verification algorithm takes as input the group public key guppy, a set of revocation tokens RL (whose elements form a subset of the elements of grt), and a purported signature \( \sigma \) on a message \( M \). It returns either valid or invalid. The latter response can mean either that \( \sigma \) is not a valid signature, or that the user who generated it has been revoked.

**CONCLUSION:**

The primitive of verifiable database with efficient updates is an important way to solve the problem of verifiable outsourcing of storage. We propose a scheme to realize efficient and secure data integrity auditing for share dynamic data with multi-user modification. The scheme vector commitment, Asymmetric Group Key Agreement (AGKA) and group signatures with user revocation are adopt to achieve the data integrity auditing of remote data. B,eside the public data auditing, the combining of the three primitive enable our scheme to outsource cipher text database to remote cloud and support secure group users revocation to shared dynamic data. We provide security analysis of our scheme, and it shows that our scheme provide data confidentiality for group users, and it is also secure against the collusion attack from the cloud storage server and revoked group users. Also, the performance analysis shows that, compared With its relevant schemes, our scheme is also efficient in different phases.
REFERENCES


