A Survey and Analysis for Accountability and Privacy of Shared Data in the Cloud

Sheetal Deshpande
Sinhgad College of Engineering
Pune-411041, Maharashtra

Deepali D. Gatade (M.E. Comp. Engg)
Sinhgad College of Engineering
Pune-411041, Maharashtra

ABSTRACT

Cloud computing is an attractive utility-computing paradigm based on Service Level Agreements (SLAs) that is experiencing rapid uptake in the commercial sector. Cloud systems offer low cost public access to vast proprietary compute, storage, and network resources. These systems provide per-user and per-application isolation and customization via a service interface that is typically implemented using high-level language technologies, well-defined APIs, and web services. Web interactions usually require the exchange of personal and confidential information for a variety of purposes, including enabling business transactions and the provisioning of services. A key issue affecting these interactions is the lack of trust and control on how data is going to be used and processed by the entities that receive this data. The data processed on clouds are often outsourced, leading to a number of issues related to accountability, including the handling of personally identifiable information. Information accountability has become a major concern for the data on the cloud. To provide information accountability for data on the cloud some major goals need to achieved:

- Fair : Data available to user hosted by CSP has to fair as given to them by cloud customer.
- Consistent : Data integrity and consistency must be preserved i.e. CSP should not discard rarely accessed data without being detected in a timely fashion
- Reliable : CSP should not attempt to hide data loss incidents and also leak the data to untrusted sources.
- Complete : CSP should not behave unfaithfully towards the cloud customer by deleting data which is rarely accessed or not fetching them good business.

Different methods are introduced to provide integrity, accountability and security for data on clouds Some are applicable at platform-level, some are implementable on CSP-side, while some are outsourced to TPA(Third Party Auditor) who audits on behalf of the user. These methods use either encryption policies or Java policies for authentication using nested JARs or sometimes even both together are used.

General Terms

Cloud Computing, Secuirty, Data Sharing

Keywords

Integrity, Accountability, Data Privacy, Access Control

1. INTRODUCTION

Cloud computing enables highly scalable services to be easily consumed over the Internet on a pay-as-you-use basis. A major feature of the cloud services is that users' data are usually processed remotely in unknown machines that users do not own or operate. While enjoying the convenience brought by this new emerging technology, users' fear of losing control of their own data can become a significant barrier to the wide adoption of cloud services. As cloud computing slowly moves into the mainstream, more and more personal data is being moved out of companies' own data centres and into the cloud, which means the data could potentially reside on servers anywhere on the planet. To enter this virtual environment requires them to transfer data throughout the cloud. Consequently, several data storage concerns can arise. Typically, users will know neither the exact location of their data nor the other sources of the data collectively stored with theirs. To ensure data confidentiality, integrity, and availability, the storage provider must offer capabilities that, at a minimum. Web interactions usually require the exchange of personal and confidential information for a variety of purposes, including enabling business transactions and the provisioning of services. A key issue affecting these interactions is the lack of trust and control on how data is going to be used and processed by the entities that receive this data. The data processed on clouds are often outsourced, leading to a number of issues related to accountability, including the handling of personally identifiable information. Information accountability has become a major concern for the data on the cloud.

2. Policy-Driven Framework For Protecting Data Privacy During Service Provisioning

The concept of the cloud computing model is that customers' data, which can be of individuals, organizations or enterprises, is processed remotely in unknown machines that users do not own or operate. The convenience and efficiency of this approach, however, comes with privacy and security risks. A significant barrier to the adoption of cloud services is the users' fear of confidential data leakage and loss of privacy in the cloud. The process of protection of users' data begins from the stage the user starts his cloud experience. The user has to manually identify the cloud provider that meets his privacy requirements, and this is often significant burden for end-users. Users cannot rely on conventional privacy policy comparison approaches since those approaches are usually designed for off-line analysis and may not be efficient to be applied in the cloud for a quick selection of a suitable service provider. After the user chooses a service provider, a common privacy policy between the user and the service provider is established. In most cases, the provider's policy may not exactly match the user's privacy requirements. A time consuming solution to this problem is to write a new policy for all participating parties. The actual problem arises during the service provisioning phase where the challenge is to ensure that the user's data is actually handled as agreed by the participating parties. Although there are several mechanisms to ensure lawful access to privacy protected data, current technologies have little to offer for reassuring individuals that their personal data is being used for the use and purposes they consented. This is especially critical among the dynamic environment of the cloud. There is an urgent need of computational mechanisms that achieve strong data protection, beyond access control.

The policy-driven framework consists of three major components: policy ranking, policy integration, and policy enforcement. Figure 1 shows the entire process flow.Let us see an

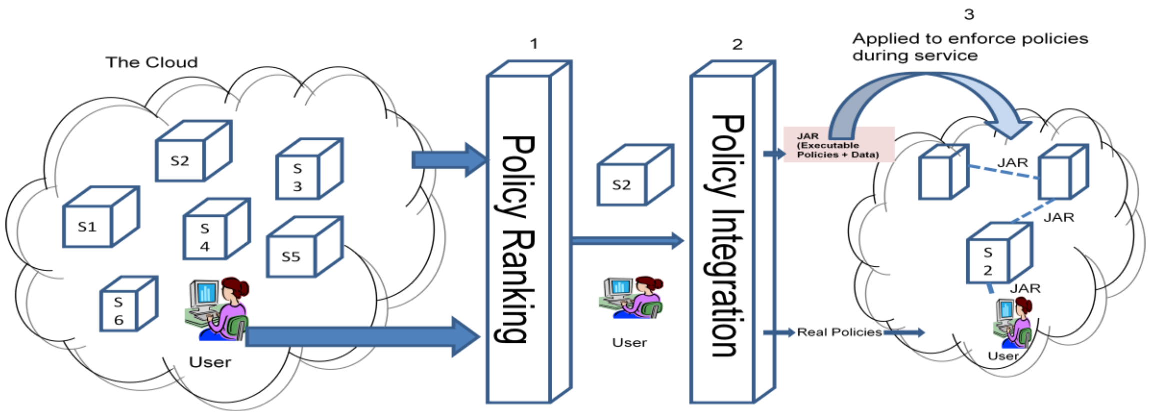

Figure 1. An overview of the policy-driven framework

example where a user named Dean joined the cloud and faces six cloud service providers, each of them able to provide the service that Dean needs. In order to find the service provider whose privacy policies best fit Dean's privacy requirements, Dean's privacy requirements and policies from service providers are fed into the policy ranking module together. The ranking module helps select service provider S2 for Dean. Since S2's privacy policies may not exactly match Dean's requirements. The second step is to send their policies to the policy integration module which will automatically generate an integrated policy as agreed by both parties. The integrated policy will be in two formats. One is in an actual policy format, i.e., a policy written in certain policy language. The other is in an executable format (like a Java JAR file) which will be used for the subsequent policy enforcement. Throughout the service, Dean's data privacy will be protected by the executable policy and the executable policy may also travel among contractors associated with service provider S2. It is worth noting that here we focus on user-related privacy policies rather than security policies at server side.

2.1. *Policy Ranking*

This step will help the user to find the service provider with the most similar privacy policies compared to the users' privacy requirements (or policies) and it is carried out at initial stage when user is searching for service providers. Policy ranking can be done using three different ways : (i) User-oriented ranking model - the users who want to avail cloud service facility need to carry out the ranking of policies; (ii) Service-provider-oriented ranking model - the service provider itself will carry out the policy ranking program; and (iii) Broker-based ranking model - Broker is a certified third party who act as a mediator between the users and the cloud service providers. The broker will be responsible for policy ranking and report back to the user a short list of service providers along with the ranking scores. Ranking of policies is done using a specific method. The final decision given by a policy is combined through certain rule combining algorithm which resolves possible conflicts among decisions yielded by different rules.[1]To analyze the similarity between two complex policies, a lightweight policy similarity comparison approach is followed in which the policy comparison approach is developed based on information retrieval techniques. It assigns a similarity score ranging in [0,1] between two policies. Higher (or lower) scores between two policies indicate that there are more (or less) numbers of requests for which the policies yield the same decision.

2.2. *Policy Integration*

The next step after the user selects a service provider, is to integrate the privacy policies of the user as well as the service provider, resolve all possible conflicts and achieve in harmony, agreement of all the requirements from both the parties. Policy integration module takes all privacy requirements as input and helps generate policies to be adopted by participating parties. Each party may have its own privacy requirements which must be satisfied. The policy integration approach handles a variety of privacy requirements raised by multiple participating parties and automatically generate actual policies as output. To integrate policies there are two approaches : (i) Centralized model :- direct service provider S is responsible for entire policy integration.; (ii) Collaborative model :- policy integration is carried out among neighbouring parties who have direct contact.

2.3. *Policy Enforcement*

After the policies have been created the last step is to correctly enforce them to guarantee the protection promised by the policies. Policy enforcement should occur while satisfying integrity, availability and confidentiality of data and policies. Policy enforcement can be achieved using two ways : (1) Tight Coupling : the policies are physically attached with the data being exchanged, so that the data cannot be left unprotected at any time; (2) Loose Coupling : The policies are stored at a remote trusted location in their native form, and then mapped into programmatic rules as the policy is invoked.

3. Flexible Distributed Storage Integrity Auditing Mechanism

Moving data into the cloud offers great convenience to users since they don't have to care about the complexities of direct hardware management. But this is resulting on dependency on clouds. There are a lot of possibilities for the CSPs to behave unfaithfully towards the cloud users regarding the status of their own outsourced data. The CSP can discard rarely accessed data without informing the user to increase the profit margin; they can also attempt to hide data loss incidents so as to maintain a reputation. Thus, even if outsourcing of data into the clouds is beneficial for long-term large-scale data storage, it is lacking assurance of data integrity and availability and hence its wide and tremendous use will affect both enterprise and individual cloud users. In order to achieve the assurances of cloud data integrity and availability and enforce the quality of cloud storage service, efficient methods that enable on-demand data correctness verification on behalf of cloud users have to be designed. However, since users no longer have physical possession of data in the cloud, it prohibits the direct adoption of traditional cryptographic primitives for the purpose of data integrity protection. Another effective and flexible distributed storage verification scheme with explicit dynamic data support can be used to ensure the correctness and availability of users' data in the cloud. The purpose of this method is to effectively detect any unauthorized data modification and corruption.

3.1 File Distribution Preparation

The first step is to prepare the file distribution across cloud servers by making use of coding theory. The erasure-correcting code is used to tolerate multiple failures in distributed storage systems. Using this technique in cloud data storage, data file is dispersed redundantly across a set of distributed servers. A Reed-Solomon erasure-correcting code is used to create redundancy parity vectors from a part of data vectors in such a way that the original set of data vectors can be reconstructed from any of the data vectors out of the complete set of data and parity vectors.[6] By placing each of the vectors on a different server, the original data file can survive the failure of any of the servers without any data loss.

3.2 Challenge Token Precomputation

Before file distribution the user precomputes a certain number of short verification tokens on individual vector, each token covering a random subset of data blocks. Later, when the user wants to make sure the storage correctness for the data in the cloud, he challenges the cloud servers with a set of randomly generated block indices. Upon receiving challenge, each cloud server computes a short "signature" over the specified blocks and returns them to the user. The values of these signatures should match the corresponding tokens precomputed by the user. To ensure the correctness of data storage, the user must precompute verification tokens for each individual vector, using a pseudorandom function , and pseudorandom permutation [6]. It is the response the user expects to receive from any server when he challenges it on the specified data blocks. After token generation, the user has the choice of either keeping the precomputed tokens locally or storing them in encrypted form on the cloud servers. Once all tokens are computed, the final step before file distribution is to blind each parity block . After blinding the parity information, the user disperses all the n encoded vectors across the cloud servers.

3.3 Correctness Verification and Error Localization

In this step the response values from servers for each challenge not only determine the correctness of the distributed storage, but also contain information to locate potential data errors. This is achieved using the challenge-response protocol. To cross check over the servers the challenge-response protocol algorithm generates a linear combination of specified rows by indices. The user verifies whether the received values remain a valid codeword determined by the secret matrix. If the received values match with secret matrix then the challenge is passed. Otherwise, it indicates that among those specified rows, there exist file block corruptions. After the inconsistency among the storage has been successfully detected, the precomputed verification tokens are used to further determine where the potential data error lies in.

3.4 File Retrieval and Error Recovery

This is the last step in which the error recovery is done and the original file is recovered. The user can reconstruct the original file by downloading the data vectors from the first m servers. After the data corruption is detected, the comparison of precomputed tokens and received response values can guarantee the identification of misbehaving servers. the user can ask servers to send back blocks of the rows specified in the challenge and regenerate the correct blocks by erasure correction. The newly recovered blocks can then be redistributed to the misbehaving servers to maintain the correctness of storage.

3.5 Delegating to Third Party Auditing

If the user does not have the time, feasibility, or resources to perform the storage correctness verification, he can optionally delegate this task to an independent third-party auditor. But the only concern is the TPA should not learn of users' data content because of delegating the auditing process. The only change that needs to be done is only change the sequence of file encoding, token precomputation and blinding.

Using the above flexible distributed scheme, cloud data integrity and availability and enforce the quality of dependable cloud storage service for users can be achieved. This method also provides explicit dynamic data support, including block update, delete, and append.

4 The Cloud Information Accountability (CIA) Framework

Researchers have investigated accountability mostly as a provable property through cryptographic mechanisms, particularly in the context of electronic commerce. The authors propose the usage of policies attached to the data and present a logic for accountability data in distributed settings .Similarly, Jagadeesan et al. recently proposed logic for designing accountability-based distributed systems . Also Crispo and Ruffo proposed an interesting approach related to accountability in case of delegation. Delegation is complementary to our work, in that we do not aim at controlling the information workflow in the clouds. In a summary, all these works stay at a theoretical level and do not include any algorithm for tasks like mandatory logging. In terms of authentication techniques, Appel and Felten proposed the Proof-Carrying authentication (PCA) framework. The PCA includes a high order logic language that allows quantification over predicates, and focuses on access control for web services.[2] While related to ours to the extent that it helps maintaining safe, high-performance, and mobile code, the PCA's goal is highly different from our research, as it focuses on validating code, rather than monitoring content. Another work is by Mont et al. who proposed an approach for strongly coupling content with access control, using Identity-Based Encryption (IBE) . We also leverage IBE techniques, but in a very different way. We do not rely on IBE to bind the content with the rules. Instead, we use it to provide strong guarantees for the encrypted content and the log files, such as protection against chosen plaintext and ciphertext attacks. In [4], the authors propose a novel approach, namely Cloud Information Accountability (CIA) framework, based on the notion of information accountability. Unlike privacy protection technologies which are built on the hide-it-or-lose-it perspective, information accountability focuses on keeping the data usage transparent and traceable. The proposed CIA framework provides end-to-end accountability in a highly distributed fashion.

The Cloud Information Accountability (CIA) framework conducts automated logging and distributed auditing of relevant access performed by any entity, carried out at any point of time at any cloud service provider. The design of the CIA framework presents substantial challenges, including uniquely identifying CSPs, ensuring the reliability of the log, adapting to a highly decentralized infrastructure, etc. The basic approach toward addressing these issues is to leverage and extend the programmable capability of JAR (Java ARchives) files to automatically log the usage of the users' data by any entity in the cloud. By means of the CIA, data owners can track not only whether or not the service level agreements are being honoured, but also enforce access and usage control rules as needed.

4.1 Problem Statement

A user who subscribed to a certain cloud service, usually needs to send his/her data as well as associated access

control policies (if any) to the service provider. After the data is received by the cloud service provider, the service provider will have granted access rights, such as read, write and copy, on the data. Using conventional access control mechanisms, once the access rights are granted, the data will be fully available at the service provider.
To understand the above problem statement let us look into an example which illustrates the problem more better.

4.1.1 Example
Alice, a professional photographer, plans to sell her photographs by using the SkyHigh Cloud Services. For her business in the cloud, she has the following requirements: Her photographs are downloaded only by paid users , Users can view the photographs before they make payment, Due to some business policies, only users from certain countries can view or download some sets of photographs, For some of her works, users are allowed to only view them for a limited time, In case any dispute arises with a client, she wants to have all the access information of that client, She wants to ensure that the cloud service providers of SkyHigh do not share her data with other service providers, so that the accountability provided for individual users can also be expected from the cloud service providers.

4.2 The CIA Architecture
The Cloud Information Accountability (CIA) framework has two major components: logger and log harmonizer. The logger is strongly coupled with user's data (either single or multiple data items). Its main tasks include automatically logging access to data items that it contains, encrypting the log record and periodically sending them to the log harmonizer. It may also be configured to ensure that access and usage control policies associated with the data are honoured.The logger is also responsible for generating the error correction information for each log record and sends the same to the log harmonizer. The error correction information combined with the encryption and authentication mechanism, provides a robust and reliable recovery mechanism. The log harmonizer is responsible for auditing. It supports two auditing strategies: push and pull. In the push strategy, the log file is pushed back to the data owner periodically in an automated fashion. The pull mode is an on-demand approach, whereby the log file is obtained by the data owner as often as requested. If there are multiple loggers for the same set of data items, the log harmonizer will merge log records from them before sending back to the data owner. The log harmonizer is also responsible for handling log file corruption. The logger and the log harmonizer are both implemented as lightweight and portable JAR files.

The data flow in the CIA framework is shown in Figure 2. Initially , each user creates a pair of public and private keys based on Identity-Based Encryption (IBE) . Using the generated key, the user will create a logger (i.e., a JAR file) to store its data items, and sign and seal it. The JAR file includes a set of access control rules specifying whether and how the cloud servers, and possibly other data stakeholders are authorized to access the data. Then, he sends the JAR file to the cloud service provider (CSP) that he subscribes to. To authenticate the CSP to the JAR, OpenSSL based certificates are used, wherein a trusted certificate authority certifies the CSP. In the event that the access is requested by a user, SAML-based authentication is employed, wherein a trusted identity provider issues certificates verifying the user's identity based on his username.[3] Using SAML-based authentication for the users allows to increase or decrease the number of users to whom the access is granted by simply adding more user identities and corresponding SAML certificates.
After the authentication is successful, the CSP (or the user) will be allowed to access the data enclosed in the JAR. Depending on the configuration settings defined at the time of creation, the JAR will provide usage control associated with logging, or will provide only logging functionality. As for the logging, each time there is an access to the data, the JAR will automatically generate a log record, encrypt it using the public key distributed by the data owner, and store it along with the data. The encryption of the log file prevents unauthorized changes to the file by attackers. The data owner can choose to reuse the same key pair for all JARs or create different key pairs for separate JARs. Using separate keys can enhance the security without introducing any overhead except in the initialization phase. In addition, some error correction information will be sent to the log harmonizer to handle possible log file corruption. The encrypted log files can be decrypted later and accessed by the data owner or other authorized stakeholders at any time for auditing purposes with the help of the log harmonizer. The programmable capability of JARs is used to conduct automated logging. A logger component is a nested Java JAR file which stores a user's data items and corresponding log files. The primary duty of the outer JAR is to handle authentication of entities which want to access the data stored in the JAR file. The outer JAR also has to select correct inner-JAR according to the identity of the entity who requests the data. Each inner JAR contains the encrypted data, class files to facilitate retrieval of log files and display enclosed data in a suitable format, and a log file for each encrypted item. There are two types of logs :

- PureLog: Its main task is to record every access to the data. The log files are used for pure auditing purpose.
- AccessLog: It has two functions: logging actions and enforcing access control. In case an access request is denied, the JAR will record the time when the request is made. If the access request is granted, the JAR will additionally record the access information along with the duration for which the access is allowed.

Log records are generated by the logger which is trigged by any access to the data in the JAR. Each record is encrypted individually and added to the log file one by one. In particular, a log record is in the form of (ID,Act,Time, Loc),[3] indicating that an entity identified by ID has performed an action Act on the user's data at time Time at location Loc. If more than one file is handled by the same logger, an additional ObjID field is added to each record. Four types of action are supported by the system : view, download, timed access, and Location-based access.[3] The time of access is determined using the Network Time Protocol to avoid suppression of the correct time by a malicious entity. The location of the cloud service provider (CSP) can be determined using IP address. The JAR can perform an IP lookup and use the range of the IP address to find the most probable location of the CSP.

For the view action type, the entity can only read the data but is not allowed to save a raw copy of it anywhere permanently. For this type of action, the PureLog will simply write a log record about the access, while the AccessLogs will enforce the action through the enclosed access control module. When there is a view-only access request, the inner JAR will decrypt the data and create a temporary decrypted file. The decrypted file will then be displayed to the entity using the Java application viewer in case the file is displayed to a human user. Presenting the data in the Java application viewer disables the copying functions using right click or other hot keys such as PrintScreen. Further, to prevent the use of some screen capture software, the data will be hidden whenever the application viewer screen is out of focus. The content is displayed using the headless mode in Java on the command line when it is presented to a CSP.

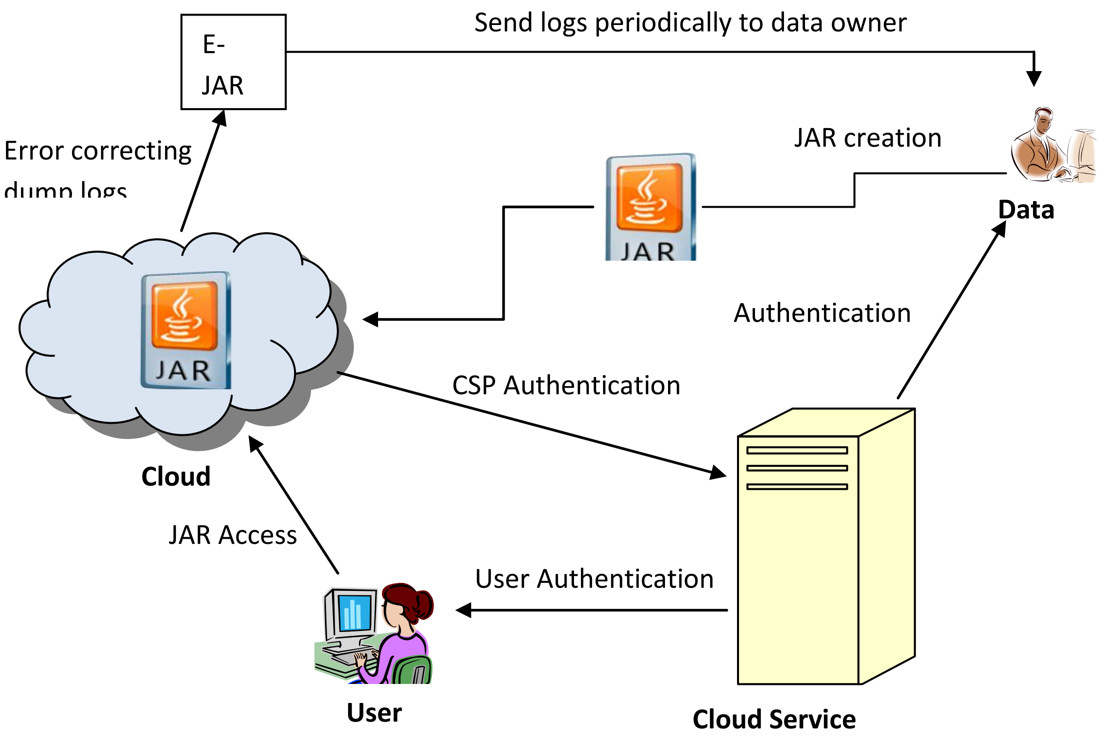

Figure 2. Overview of the CIA framework

In the download type of action, the entity is allowed to save a raw copy of the data and the entity will have no control over this copy neither log records regarding access to the copy.

The timed access action is combined with the view only access, and it indicates that the data is made available only for a certain period of time. The Purelog will just record the access starting time and its duration, while the AccessLog will enforce that the access is allowed only within the specified period of time. The duration for which the access is allowed is calculated using the Network Time Protocol (NTP).[3] To enforce the limit on the duration, the AccessLog records the start time using the NTP, and then uses a timer to stop the access.

In the location-based access PureLog will record the location of the entities. The AccessLog will verify the location for each of such access. The access is granted and the data is made available only to entities located at locations specified by the data owner.

To handle the distributed nature of this approach each log harmonizer is in charge of copies of logger components containing the same set of data items. The harmonizer is implemented as a JAR file. It does not contain the user's data items being audited, but consists of class files for both a server and a client processes to allow it to communicate with its logger components. The harmonizer stores error correction information sent from its logger components, as well as the user's IBE decryption key, to decrypt the log records and handle any duplicate records. Duplicate records result from copies of the user's data JARs. Since user's data is strongly coupled with the logger component in a data JAR file, the logger will be copied together with the user's data.[2] For recovering purpose, logger components are required to send error correction information to the harmonizer after writing each log record. Therefore, logger components always ping the harmonizer before they grant any access right. If the harmonizer is not reachable, the logger components will deny all access. In this way, the harmonizer helps prevent attacks which attempt to keep the data JARs offline for unnoticed usage.

To allow users to be timely and accurately informed about their data usage two auditing mechanisms are used : push mode and pull mode. In push mode, the logs are periodically pushed to the data owner by the harmonizer. The push action will be triggered by either type of the following two events: one is that the time elapses for a certain period according to the temporal timer inserted as part of the JAR file; the other is that the JAR file exceeds the size stipulated by the content owner at the time of creation. After the logs are sent to the data owner, the log files will be dumped, so as to free the space for future access logs. Along with the log files, the error correcting information for those logs is also dumped. The pull mode allows data owners to retrieve the logs anytime when they want to check the recent access to their own data. The pull message consists simply of an FTP pull command, which can be issues from the command line.

5. Conclusion and Future Work

Cloud computing has raised a range of important privacy and security issues. Such issues are due to the fact that, in the cloud, users' data and applications reside at least for a certain amount of time on the cloud cluster which is owned and maintained by a third party. Concerns arise since in the cloud it is not always clear to individuals why their personal information is requested or how it will be used or passed on to other parties. The privacy problem in the cloud is also compounded by the fact that some of the issues are not technical in nature , and rather deal with law and regulations.

The Table 1. of comparison is based on the comparative study carried out by us which is subject to vary on ranking.

Table 1 Comparision of the methods

Sr. No	Name of Method	Security	Privacy	Accountability	Speed	Description
1	Policy-Driven Framework For Protecting Data Privacy During Service Provisioning	Average	Very good	Not applicable	Good	This method helps the user to choose an appropriate CSP.
2	Flexible Distributed Storage Integrity Auditing Mechanism	Good	Good	Not applicable	Good	This method ensures correctness and availability of users' data in the cloud.
3	The Cloud Information Accountability Framework	Excellent	Very Good	Excellent	Very Good	This method conducts automated logging and distributed auditing of relevant access performed by any entity, carried out at any point of time at any cloud service provider.

The approach used in the CIA method provides security of the users' data hosted on the cloud as well as provides accountability of the data.The policy driven framework only protects users' data by helping them choose an appropriate CSP. The flexible distributed storage mechanism assures integrity and availability of data but again doesnot provide any accountability mechanism. Thus out of all the methods only CIA provides access control on the users' data.

6. References

[1] Anna Squicciarini and Dan Lin, 2010. Data protection models for service provisioning in the cloud.

[2] Anna C. Squicciarini , Dan Lin and Smitha Sundareswaran, 2012. Ensuring Distributed Accountability for Data Sharing in the Cloud. In IEEE Transactions on Dependable and Secure Computing.

[3] Anna C. Squicciarini , Dan Lin, Shuo Huang and Smitha Sundareswaran, 2011. Promoting Distributed Accountability in the Cloud. In IEEE Transactions on Dependable and Secure Computing.

[4] B. Crispo and G. Ruffo, 2001. Reasoning about Accountability within Delegation.

[5] Cong Wang, Jin Li, Kui Ren and Wenjing Lou, 2010. Toward Publicly Auditable Secure Cloud Data Storage Services.

[6] Cong Wang, Kui Ren, Ning Cao, Qian Wang, and Wenjing Lou, 2012. Towards Secure and Dependable Storage Services in Cloud Computing.

[7] Cong Wang, Qian Wang, Kui Ren and Wenjing Lou, 2010. Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing. In IEEE INFOCOM 2010.

[8] Song, D., Shi, E., Fischer, I. and Shankar, U., 2012. Cloud Data Protection for the Masses. In Magazine on Computer.

[9] Cloud Security Alliance (2009) Security Guidance for Critical Areas of Focus in Cloud Computing.Available online at: http://www.cloudsecurityalliance.org/guidance/

[10] Hsiao-Ying Lin, Wen-Guey Tzeng, 2012. A Secure Erasure Code-Based Cloud Storage System with Secure Data Forwarding. In IEEE Transactions On Parallel And Distributed Systems.

[11] OASIS Security Services Technical Committee, "Security Assertion Markup Language (saml) 2.0," http://www.oasis-open.org/committees/tc home.php?wg abbrev=security, 2012.

[12] R. Corin, S. Etalle, J.I. den Hartog, G. Lenzini, and I. Staicu, 2005. A Logic for Auditing Accountability in Decentralized System.