SECURE DATA SHARING IN MOBILE CLOUD COMPUTING: A LIGHTWEIGHT APPROACH

[1] K.Mahesh, Assistant Professor, Department of CSE, Chalapathi Institute of Technology, Guntur.
[2] Ramineni Supraja, B.Tech, Department of CSE, Chalapathi Institute of Technology, Guntur.
[3] Muppa Ravi Kishore, B.Tech, Department of CSE, Chalapathi Institute of Technology, Guntur.
[4] Boda Ramu, B.Tech, Department of CSE, Chalapathi Institute of Technology, Guntur.
[5] Syed Sadaf S, B.Tech, Department of CSE, Chalapathi Institute of Technology, Guntur.

Abstract: With the popularity of cloud computing, mobile devices can store/retrieve personal data from anywhere at any time. Consequently, the data security problem in mobile cloud becomes more and more severe and prevents further development of mobile cloud. There are substantial studies that have been conducted to improve the cloud security. However, most of them are not applicable for mobile cloud since mobile devices only have limited computing resources and power. Solutions with low computational overhead are in great need for mobile cloud applications. In this paper, we propose a lightweight data sharing scheme (LDSS) for mobile cloud computing. It adopts CP-ABE, an access control technology used in normal cloud environment, but changes the structure of access control tree to make it suitable for mobile cloud environments. LDSS moves a large portion of the computational intensive access control tree transformation in CP-ABE from mobile devices to external proxy servers. Furthermore, to reduce the user revocation cost, it introduces attribute description fields to implement lazy-revocation, which is a thorny issue in program based CP-ABE systems. The experimental results show that LDSS can effectively reduce the overhead on the mobile device side when users are sharing data in mobile cloud environments.

1. INTRODUCTION

With the development of cloud computing and the popularity of smart mobile devices, people are gradually getting accustomed to a new era of data sharing model in which the data is stored on the cloud and the mobile devices are used to store/retrieve the data from the cloud. Typically, mobile devices only have limited storage space and computing power. On the contrary, the cloud has enormous amount of resources. In such a scenario, to achieve the satisfactory performance, it is essential to use the resources provided by the cloud service provider (CSP) to store and share the data. Nowadays, various cloud mobile applications have been widely used. In these applications, people (data owners) can upload their photos, videos, documents and other files to the cloud and share these data with other people (data users) they like to share. CSPs also provide data management functionality for data owners. Since personal data files are sensitive, data owners are allowed to choose whether to make their data files public or can only be shared with specific data users. Clearly, data privacy of the personal sensitive data is a big concern for many data owners. The state-of-the-art privilege management/access control mechanisms provided by the CSP are either not sufficient or not very convenient. They cannot meet all the requirements of data owners. First, when people upload their data files onto the cloud, they are leaving the data in a place where is out of their control, and the CSP may spy on user data for its commercial interests and/or other reasons. Second, people have to send password to each data user if they only want to share the encrypted data with certain users, which is very cumbersome. To simplify the privilege management, the data owner can divide data users into different groups and send password to the groups which they want to share the data. However, this approach requires fine-grained access control. In both cases, password management is a big issue. Apparently, to solve the above problems, personal sensitive data should be encrypted before uploaded onto the cloud so that the data is secure against the CSP. However, the data encryption brings new problems. How to provide efficient access control mechanism on ciphertext decryption so that only the authorized users can access the plaintext data is challenging. In addition, system must offer data owners effective user privilege management capability, so they can grant/revoke data access privileges easily on the data users. There have been substantial researches on the issue of data access control over cipher text. In these researches, they have the following common assumptions. First, the CSP is considered honest and curious. Second, all the sensitive data are encrypted before uploaded to the Cloud. Third, user authorization on certain data is achieved through encryption/decryption key distribution. In general, we can divide these approaches into four categories: simple cipher text access control, hierarchical access control, access control based on fully homo morphic encryption [1][2] and access control based on attribute-based encryption (ABE). All these proposals are designed for non-mobile cloud environment. They consume large amount of storage and computation resources, which are not available for mobile

devices. According to the experimental results in [26], the basic ABE operations take much longer time on mobile devices than laptop or desktop computers. It is at least 27 times longer to execute on a smart phone than a personal computer (PC). This means that an encryption operation which takes one minute on a PC will take about half an hour to finish on a mobile device. Furthermore, current solutions don't solve the user privilege change problem very well. Such an operation could result in very high revocation cost. This is not applicable for mobile devices as well. Clearly, there is no proper solution which can effectively solve the secure data sharing problem in mobile cloud. As the mobile cloud becomes more and more popular, providing an efficient secure data sharing mechanism in mobile cloud is in urgent need. To address this issue, in this paper, we propose a Lightweight Data Sharing Scheme (LDSS) for mobile cloud computing environment. The main contributions of LDSS are as follows: (1) We design an algorithm called LDSS-CP-ABE based on Attribute-Based Encryption (ABE) method to offer efficient access control over ciphertext. (2) We use proxy servers for encryption and decryption operations. In our approach, computational intensive operations in ABE are conducted on proxy servers, which greatly reduce the computational overhead on client side mobile devices. Meanwhile, in LDSS-CP-ABE, in order to maintain data privacy, a version attribute is also added to the access structure. The decryption key format is modified so that it can be sent to the proxy servers in a secure way. (3) We introduce lazy re-encryption and description field of attributes to reduce the revocation overhead when dealing with the user revocation problem. (4) Finally, we implement a data sharing prototype framework based on LDSS. The experiments show that LDSS can greatly reduce the overhead on the client side, which only introduces a minimal additional cost on the server side. Such an approach is beneficial to implement a realistic data sharing security scheme on mobile devices. The results also show that LDSS has better performance compared to the existing ABE based access control schemes over ciphertext. The rest of this paper is organized as follows. Section 2 presents some fundamental concepts in secure mobile cloud data sharing and the security premise. Section 3 gives the detailed design of LDSS. Section 4 and 5 give the safety assessment and performance evaluation, respectively. Section 6 presents related works. Finally, Section 7 concludes our work with the future work.

2. LITERATURE SURVEY

2.1 Efficient fully homomorphic encryption from (standard) LWE

We present a fully homomorphic encryption scheme that is based solely on the (standard) learning with errors (LWE) assumption. Applying known results on LWE, the security of our scheme is based on the worst-case hardness of "short vector problems" on arbitrary lattices. Our construction improves on previous works in two aspects.

2.2 Data leakage mitigation for discertionary access control in collaboration clouds.

With the growing popularity of cloud computing, more and more enterprises are migrating their collaboration platforms from in-enterprise systems to Software as a Service (SaaS) applications. While SaaS collaboration has numerous advantages, it also raises new security challenges. In particular, since SaaS collaboration is increasingly used across enterprise boundaries, organizations are concerned that sensitive information may be leaked to outsiders due to their employees' inadvertent mistakes on information sharing. In this article, we propose to mitigate the data leakage problem in SaaS collaboration systems by reducing human errors. Built on top of the discretionary access control model in existing collaboration systems, we have designed a series of mechanisms to provide defense in depth against information leakage. First, we allow enterprises to encode their organizational security rules as mandatory access control policies, so as to impose coarse-grained restrictions on their employees' discretionary sharing decisions. Second, we design an attribute-based recommender that suggests and prioritizes potential recipients for users' files, reducing errors in the choices of recipients. Third, our system actively examines abnormal recipients entered by a file owner, providing the last line of defense before a file is shared. We have implemented a prototype of our solution and performed experiments on data collected from real-world collaboration systems.

2.3 On Implementing Deniable Storage Encryption for Mobile Devices.

Data confidentiality can be effectively preserved through encryption. In certain situations, this is inadequate, as users may be coerced into disclosing their decryption keys. In this case, the data must be hidden so that its very existence can be denied. Steganographic techniques and deniable

encryption algorithms have been devised to address this specific problem. Given the recent proliferation of smart phones and tablets, we examine the feasibility and efficacy of deniable storage encryption for mobile devices. We evaluate existing, and discover new, challenges that can compromise plausibly deniable encryption (PDE) in a mobile environment. To address these obstacles, we design a system called Mobiflage that enables PDE on mobile devices by hiding encrypted volumes within random data on a device's external storage. We leverage lessons learned from known issues in deniable encryption in the desktop environment, and design new countermeasures for threats specific to mobile systems. Key features of Mobiflage include: deniable file systems with limited impact on throughput; efficient storage use with no data expansion; and restriction/prevention of known sources of leakage and disclosure. We provide a proof-of-concept implementation for the Android OS to assess the feasibility and performance of Mobiflage. We also compile a list of best practices users should follow to restrict other known forms of leakage and collusion that may compromise deniability.

2.4 Secure and efficient access to outsourced data:

Providing secure and efficient access to large scale outsourced data is an important component of cloud computing. In this paper, we propose a mechanism to solve this problem in owner-write-users-read applications. We propose to encrypt every data block with a different key so that flexible cryptography-based access control can be achieved. Through the adoption of key derivation methods, the owner needs to maintain only a few secrets. Analysis shows that the key derivation procedure using hash functions will introduce very limited computation overhead. We propose to use over-encryption and/or lazy revocation to prevent revoked users from getting access to updated data blocks. We design mechanisms to handle both updates to outsourced data and changes in user access rights. We investigate the overhead and safety of the proposed approach, and study mechanisms to improve data access efficiency.

2.5 Attribute-based fine-grained access control with efficient revocation in cloud storage systems:

A cloud storage service allows data owner to outsource their data to the cloud and through which provide the data access to the users. Because the cloud server and the data owner are not in the same trust domain, the semi-trusted cloud server cannot be relied to enforce the access policy. To address this challenge, traditional methods usually require the data owner to encrypt the data and deliver decryption keys to authorized users. These methods, however, normally involve complicated key management and high overhead on data owner. In this paper, we design an access control framework for cloud storage systems that achieves fine-grained access control based on an adapted Cipher text-Policy Attribute-based Encryption (CP-ABE) approach. In the proposed scheme, an efficient attribute revocation method is proposed to cope with the dynamic changes of users' access privileges in large-scale systems. The analysis shows that the proposed access control scheme is provably secure in the random oracle model and efficient to be applied into practice.

3. EXISTING SYSTEM

We implement a data sharing prototype framework based on LDSS. The experiments show that LDSS can greatly reduce the overhead on the client side, which only introduces a minimal additional cost on the server side. Such an approach is beneficial to implement a realistic data sharing security scheme on mobile devices. The results also show that LDSS has better performance compared to the existing ABE based access control schemes over cipher text. We first evaluate the computational overhead of LDSS and compare it with existing access control schemes. DO's overhead in different ABE schemes is shown. As shown in Table 4, in existing programs, the overhead on mobile user DU's side is proportional to the number of attributes in access control policy. In LDSS, the overhead is a small constant value.

DISADVANTAGES OF EXISTING SYSTEM

- Current solutions don't solve the user privilege change problem very well. Such an operation could result in very high revocation cost.
- This is not applicable for mobile devices as well.
- Clearly, there is no proper solution which can effectively solve the secure data sharing problem in mobile cloud.

4. PROPOSED SYSTEM

Attribute-based encryption (ABE) is proposed. It is derived from the Identity-Based Encryption (IBE) and is particularly suitable for one-to-many data sharing scenarios in a distributed and open cloud environment. Attribute-based encryption is divided into two categories: one is the Cipher text-Policy Attribute Based Encryption (CP-ABE), in which the access control policy is embedded into cipher

text; the other one is Key- Policy Attribute Based Encryption (KP-ABE), in which the access control policy is embedded in the user's key attributes. In real applications, CP-ABE is more suitable since it resembles role-based access control. In CP-ABE, the data owner designs the access control policy and assigns attributes to data users. A user can decrypt the data properly if the user's attributes satisfy the access control policy.

ADVANTAGES

- The overhead of user revocation is proportional to the number of data users, and LDSS works better than other CP-ABE. When the number of revoked attributes grows bigger, this advantage becomes more obvious.
- We design an algorithm called LDSS-CP-ABE based on Attribute-Based Encryption (ABE) method to offer efficient access control over cipher text.
- We use proxy servers for encryption and decryption operations. In our approach, computational intensive operations in ABE are conducted on proxy servers, which greatly reduce the computational overhead on client side mobile devices.

SYSTEM ARCHITECTURE

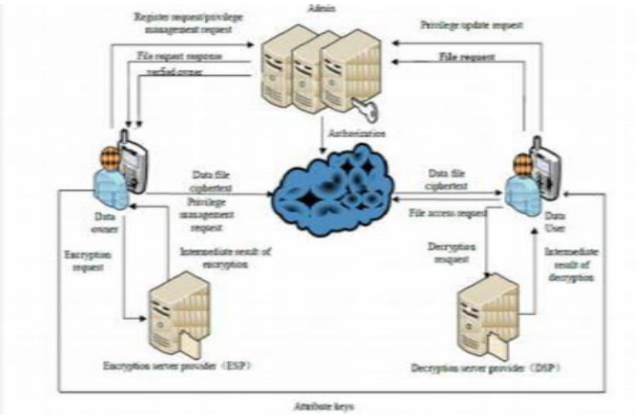

Fig 1: System Architecture

5. UML DIAGRAMS

1. CLASS DIAGRAM

The cornerstone of event-driven data exploration is the class outline. Both broad practical verification of the application's precision and fine-grained demonstration of the model translation into software code rely on its availability. Class graphs are another data visualisation option.

The core components, application involvement, and class changes are all represented by comparable classes in the class diagram. Classes with three-participant boxes are referred to be "incorporated into the framework," and each class has three different locations:

• The techniques or actions that the class may use or reject are depicted at the bottom.

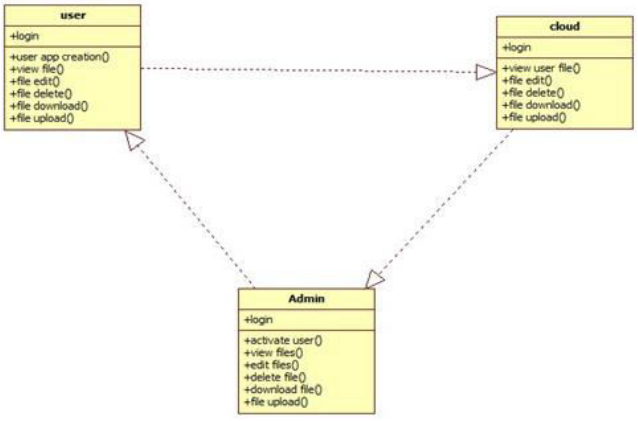

Fig 5.1 shows the class diagram of the project

2. USECASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

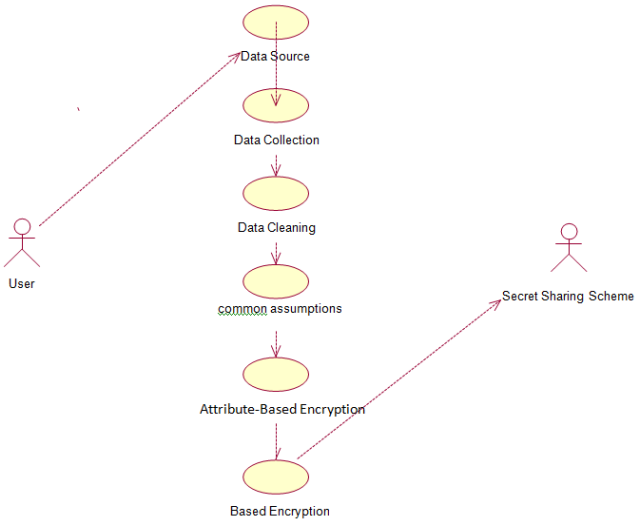

Fig 5.2 Shows the Use case Diagram

3. SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

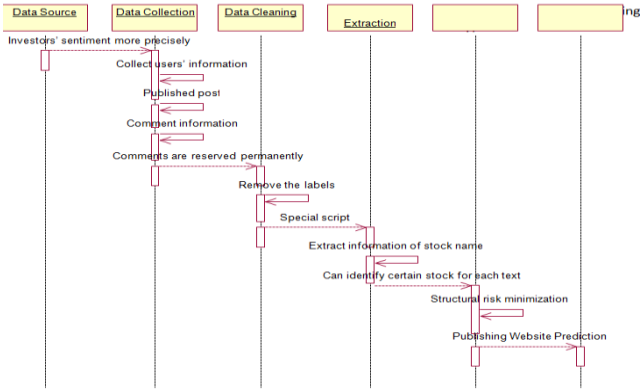

Fig 5.3 Shows the Sequence Diagram

6. RESULTS

6.1 Output Screens

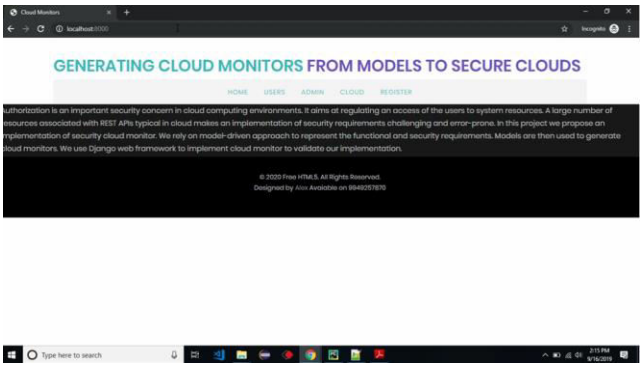

Fig 6.1 Home Page

To run the manage.py file it will generate the home page

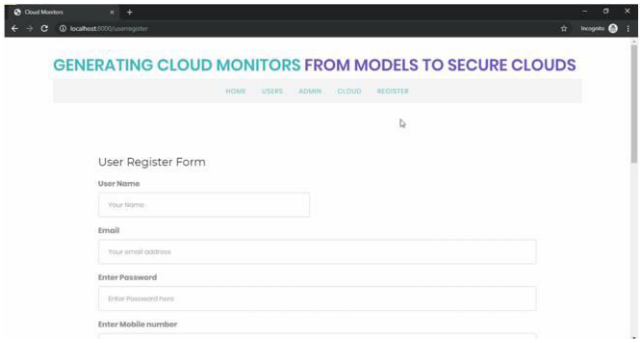

Fig 6.2 User Registration

In above to register the remote user.

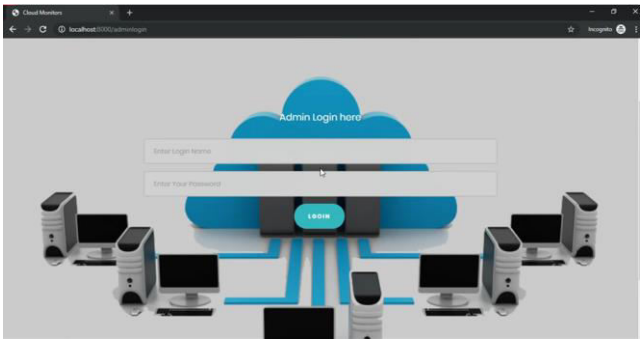

Fig 6.3 Admin Login

In above screen shows the admin login.

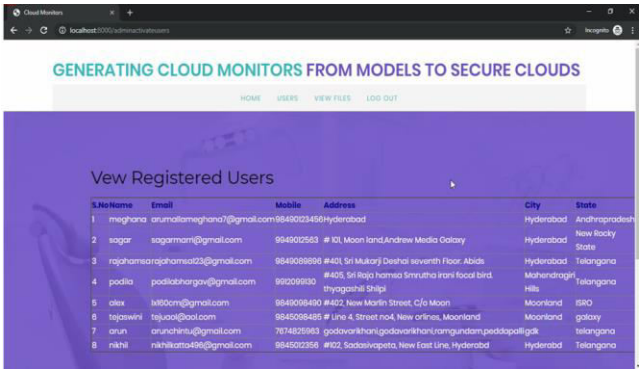

Fig 6.4 Admin Activate the Remote users

In above screen the admin can activate the remote users.

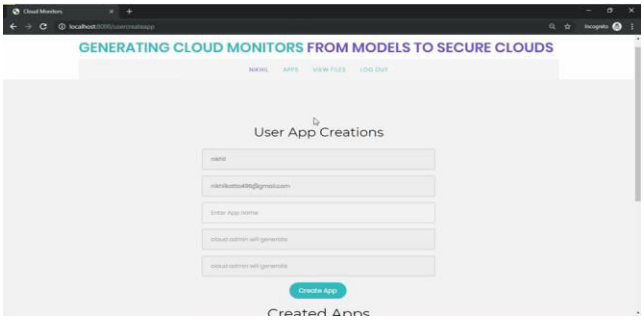

Fig 6.5 User create the app

In above screen the user can create the own application

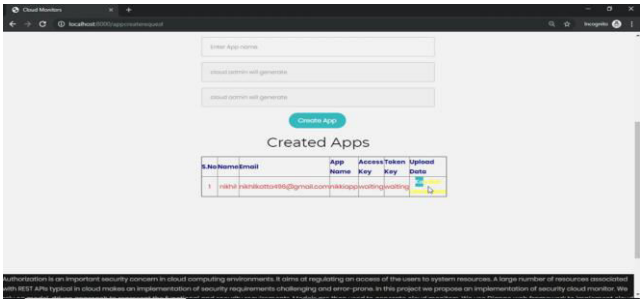

Fig 6.6 User Created Apps

In the above screen shows user created apps.

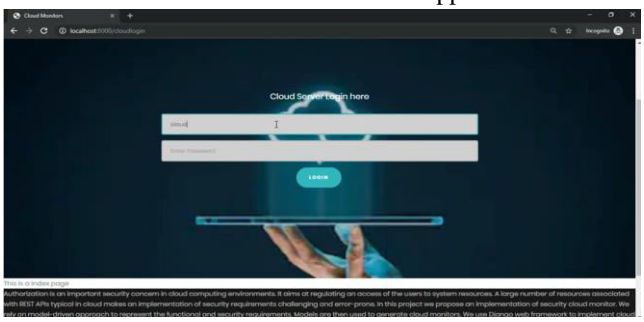

Fig 6.7 Cloud Server Login

In above screen shows the cloud server login.

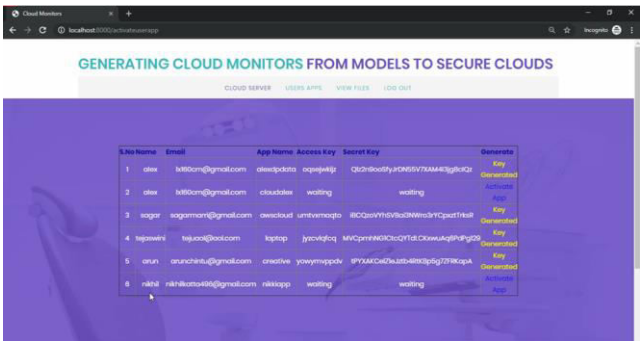

Fig 6.8 Cloud Server Generates the Key

In the above screen shows the cloud server generating the key.

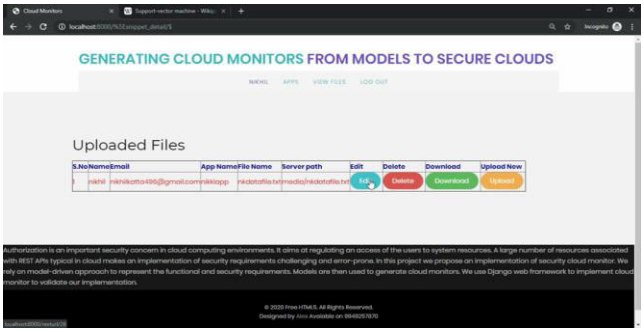

Fig 6.9 User Uploading the File to the Cloud Server

In the above screen shows the user can uploading the file to the cloud server.

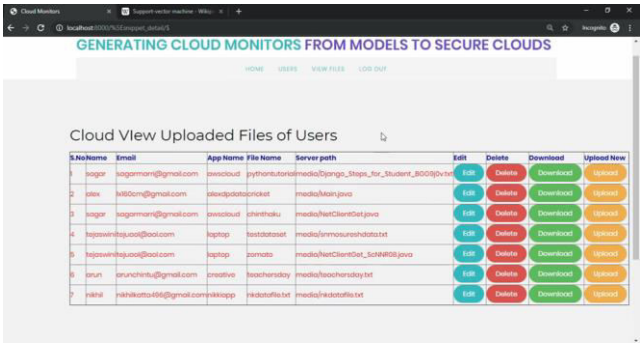

Fig:6 10 Users in the Cloud

In the above screen shows the users available in the cloud.

7. CONCLUSION

In this paper, we have presented an approach and associated tool for monitoring security in cloud. We have relied on the model-driven approach to design APIs that exhibit REST interface features. The cloud monitors, generated from the models, enable an automated contract-based verification of correctness of functional and security requirements, which are implemented by a private cloud infrastructure. The proposed semi-automated approach aimed at helping the cloud developers and security experts to identify the security loopholes in the implementation by relying on modeling rather than manual code inspection or testing. It helps to spot the errors that might be exploited in data breaches or privilege escalation attacks. Since open source cloud frameworks usually undergo frequent changes, the automated nature of our approach allows the developers to relatively easily check whether functional and security requirements have been preserved in new releases.

8. REFERENCES

[1] Amazon Web Services. https://aws.amazon.com/. Accessed: 30.11.2017.

[2] Block Storage API V3 . https://developer.openstack.org/api-ref/ block-storage/v3/. retrieved: 126.2017.

[3] Cloud Computing Trends: 2017 State of the Cloud Survey. https://www. rightscale.com/blog/cloud-industry-insights/. Accessed: 30.11.2017.

[4] cURL. http://curl.haxx.se/. Accessed: 20.08.2013.

[5] Extensible markup language (xml). https://www.w3.org/XML/. Accessed: 27.03.2018.

[6] Keystone Security and Architecture Review. Online at https://www.openstack.org/summit/ openstack-summit-atlanta-2014/session-videos/presentation/keystonesecurity-and-architecture-review. retrieved: 06.2017.

[7] Nomagic MagicDraw. http://www.nomagic.com/products/magicdraw/. Accessed: 27.03.2018.

[8] OpenStack Block Storage Cinder. https://wiki.openstack.org/wiki/ Cinder. Accessed: 26.03.2018.

[9] OpenStack Newton - Installation Guide. https://docs.openstack.org/ newton/install-guide-ubuntu/overview.html. Accessed: 20.11.2017.

[10] urllib2 - extensible library for opening URLs. Python Documentation. Accessed: 18.10.2012.

[11] Windows Azure. https://azure.microsoft.com. Accessed: 30.11.2017. [

12] MM Alam et al. Model driven security for web services (mds4ws). In Multitopic Conference, 2004. Proceedings of INMIC 2004. 8th International, pages 498–505. IEEE, 2004.

[13] Mohamed Almorsy et al. Adaptable, model-driven security engineering for saas cloud-based applications. Automated Software Engineering, 21(2):187–224, 2014.

[14] Christopher Bailey et al. Run-time generation, transformation, and verification of access control models for self-protection. In Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pages 135–144. ACM, 2014.

[15] Tim Berners-Lee et al. Hypertext transfer protocol–HTTP/1.0, 1996.

[16] Gaurav Bhatnagar and QMJ Wu. Chaos-based security solution for fingerprint data during communication and transmission. IEEE Transactions on Instrumentation and Measurement, 61(4):876–887, 2012.

[17] David Ferraiolo et al. Role-based access control (rbac): Features and motivations. In Proceedings of 11th annual computer security application conference, pages 241–48, 1995.

[18] Django Software Foundation. Django Documentation. Online Documentation of Django 2.0, 2017. https://docs.djangoproject.com/en/2.0/.

[19] Michal Gordon and David Harel. Generating executable scenarios from natural language. In International Conference on Intelligent Text Processing and Computational Linguistics. Springer, 2009.

[20] Robert L Grossman. The case for cloud computing. IT professional, 11(2):23–27, 2009.