World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:10, No:4, 2016

Towards a Secure Storage in Cloud Computing

Mohamed Elkholy, Ahmed Elfatatry

Abstract**—**Cloud computing has emerged as a flexible computing paradigm that reshaped the Information Technology map. However, cloud computing brought about a number of security challenges as a result of the physical distribution of computational resources and the limited control that users have over the physical storage. This situation raises many security challenges for data integrity and confidentiality as well as authentication and access control. This work proposes a security mechanism for data integrity that allows a data owner to be aware of any modification that takes place to his data. The data integrity mechanism is integrated with an extended Kerberos authentication that ensures authorized access control. The proposed mechanism protects data confidentiality even if data are stored on an untrusted storage. The proposed mechanism has been evaluated against different types of attacks and proved its efficiency to protect cloud data storage from different malicious attacks.

Keywords**—**Access control, data integrity, data confidentiality, Kerberos authentication, cloud security.

I. Introduction

CLOUD Computing is a promising computing model that provides better utilization of computing resources [1]. Cloud Computing enables on-demand network access to a shared pool of computing resources [2]. Computing recourses can be invoked and delivered as services by any client who has Internet access.

Cloud computing allows users to scale up and down their resources utilization in a pay-as-you-use fashion [3]. The flexibility offered attracted many users to migrate their businesses to the cloud. However, resource sharing leads to significant security concerns for users when their data is physically separated from them. Traditional access control architectures usually assume that the data owner and the servers storing the data are in the same trusted domain [4]. However, this assumption is not valid in cloud computing since the data owner and cloud servers are in two different trust domains [5].

In this work, we propose a mechanism to ensure data integrity on clouds. Applying the proposed mechanism allows a data owner to be aware of any modification done to his data. The proposed work also introduces an extension to the Kerberos authentication protocol to be suitable to the cloud environment. The Kerberos extension considers the situation in which the data storage provider is not trusted by the data owner. The data integrity mechanism is then integrated with the extended Kerberos to achieve a robust access control method. The proposed mechanism ensures data confidentiality and can be added to contracts between storage providers and data owners. The client data is divided into small segments that are passed to a hash function to create a hash table. This hash table is further used to check the data modification, and to be added to the access control mechanism.

Cloud providers use virtualization technologies to increase resources utilization. Virtualization leads to hosting several types of virtual machines on the same physical storage [6]. Several data owners do not depend on the cloud to store their data due to security concerns [7]. The process of outsourcing data to an untrusted storage that uses virtualization brings about a number of challenges. A provider may be a source of attack. Another client that shares the same physical storage on another virtual machine may be another source of attack [8]. Moreover, the data owner does not have the authority to protect the server against any malicious attack [9]. These challenges should be addressed to help many clients to store their data on the cloud. In this work, we propose a robust security mechanism at the virtualization level allowing different data owners to trust cloud storage services.

The remaining of the paper is organized as follows: Section II presents a literature review of cloud data integrity problems and solutions. Section III discusses the problem definition and solution requirements. Section IV proposes the security issues related to data storage on the cloud. Section V introduces the data integrity mechanism. Section VI presents the proposed access control method. An evaluation of the proposed work is presented in Section VII. Finally, Section VIII presents the conclusion.

II. Related Work

Reference [10] introduces a solution to facilitate secure interactions between two parties that do not trust each other. The solution depends on a trusted third party (TTP) which is an entity trusted by the two communicating parties. The third party is responsible for authentication and access control. However, using such approach results in an increase in the service cost and requires many messages to be delivered and processed by the trusted party. Moreover, data owners may need to frequently access their data. In each access process, a communication overhead is expected.

A browser-based Kerberos protocol was introduced in [11]. This protocol provides access control to different web browsers. The protocol guarantees that private keys and sessions keys are confidential for authorized clients. The work introduces three participants: a browser, a server and an authentication server. The browser is responsible for authentication process on behalf of the client. The browser authenticates itself to the Kerberos authentication server by TLS hand checking. The web browser choses a random number and sends it to Kerberos server encrypted with his public key. A symmetric key is then derived from this key and

Mohamed Elkholy is with the Alexandria University, Egypt (e-mail: eng_mikholy@yahoo.com).

sent back to the browser to start communication. This approach is suitable for message transfer. However, it is not suitable to access data from a public storage. It cannot provide data integrity as there is no key driven from the stored data. Also, the mechanism relies on TLS and cannot work as separate mechanism.

Shucheng et al. [12] used a combined cryptographic technique to achieve secure and fine-grained access control on data stored in the cloud. They used three cryptographic techniques: KP-ABE, PRE and lazy re-encryption. Their work associates each data file with a set of attributes, then each user is given access structure defined by these attributes. Their work provides a secure access control. However, employing three cryptographic techniques increases the complexity of the security mechanism. For large data storage and frequent demands of data access, such solution becomes unpractical.

Boyang et al. [13] introduced a public auditing mechanism to ensure data integrity. Their work deals with data stored in cloud storage where a group of users have the authority to modify it. To achieve data integrity every user is responsible for sinning the modified block of data before logging out. They introduced a mechanism that relies on a trusted third party to sign data blocks on behalf of users. Such mechanism cannot provide access control to protect the cloud data from unauthorized attacks. Moreover, asymmetric encryption of stored data blocks consumes a lot of time and computational resources. Another limitation of the mechanism is that it cannot provide fine grained access control. All users can access all data blocks without any classification.

Reference [14] introduces the PasS approach (Privacy as a Service) that relies on cryptographic coprocessors to provide security at the physical layer. A cryptographic coprocessor is a hardware card that interfaces with a main computer or server. It is used with different physical storages that apply virtualization. Each coprocessor has a tamper-responding mechanism which resets the internal state of the coprocessor upon detecting any unauthorized activity on the physical storage. A third-party loads the cryptographic data structures on the crypto coprocessor. PasS provides the data owner with trusted access control as well as data integrity control mechanism. Moreover, the approach provides the client with a privacy feedback process which informs the client with any potential risk to his data. However, the approach uses extra hardware that raises its cost, and needs more interface maintenance. It also relies on a third party who is responsible for reconfiguring the coprocessor.

Most of the work reported in the literature relies on the role of a third party that is trusted by both the cloud provider and the data owner [13]. Every time a client needs to access data, his request has to pass through a third party. Such situation increases the network traffic and delay the data access process. Moreover, each of the reported works focuses only on one type of security aspects. Hence, to achieve access control and data integrity, two different mechanisms should be used. Such situation increases the complexity and the delay time to access the cloud data. Also, most of the reported techniques do not deal with dynamic data storage in which users frequently access and update their data.

III. Problem Definition and Research Questions

This section specifies the research problems and the requirements that a successful solution has to satisfy.

A. Problem Definition

The first problem is the inability of current techniques to ensure data integrity and data confidentiality in shared large data storages. Cloud services are almost provided by commercial providers that are likely out of the trust domain of the cloud clients [4]. Hence, storage providers do not have the ability to provide the data owner with the mechanism that proves that his data has not been changed [9]. Moreover, the data owner is usually not aware of any malicious attempts to modify his data [10].

The second problem is that the cloud data storage provider cannot provide the data owner with a fine-grained access control [15]. The data owner is not capable of providing different types of access roles to different his clients. The traditional data encryption mechanisms allow any client that knows the encryption key to access all data. While in many cases the data owner needs to specify different roles for dealing with his data stored on the cloud.

B. Research Questions

In order to solve the problems defined in Section III. *A,* we need to address the following questions.

1. How to ensure data integrity in cloud data storages without downloading the data to the client side?
2. How to extend Kerberos authentication protocol to fit cloud environment in which the data servers are not trusted by the data owner?
3. How to build an access control mechanism that shares security aspects between the data owner and the cloud provider?
4. What is the general definition of trust? What does it mean that entity A trusts entity B?

C. Solution Requirements

A successful solution has to satisfy the following requirements.

- RQ1: There is a need for a trusted method that allows the cloud data owners to check their data integrity. This method should not be under the control of the data storage provider, since the cloud provider may be not trusted by data owner.
- RQ2: There is a need to automatically inform the data owner when his data has changed. The owner should have the capability to distinguish between authorized and unauthorized modification of his data.
- RQ3: There is a need for an access control mechanism that satisfies the security requirements of both the storage provider and the data owner.
- RQ4: There is a need to ensure data confidentiality without encrypting all the client data.

IV. Security Issues and System Model

The proposed work focuses on the security of public cloud. Public cloud offers services in an open environment on the Internet and any user can utilize it according to the service provider rules [4]. This work is concerned with Infrastructure as a Service (IaaS). IaaS refers to on-demand infrastructural resources, usually in terms of virtual machines [16]. The infrastructural resources vary from processor to memory to data storage [17]. Amazon EC2 is an example of IaaS providers. This section discusses security issues in cloud data storages

A. Security Issues in Cloud Data Storage

Cloud data storages are very large physical storages that offer shared data storage service for cloud clients [13]. Cloud storage providers use virtualization to achieve better utilization of their physical storage [18]. Hence, data owners deal with their data according to the logical address of their data blocks rather than their real physical addresses [19]. Fig. 1 shows the virtualization layers of cloud data storages. The shared pool of data storage raises many security issues. There is a possibility that the attack could be initiated by: the cloud provider, malicious nodes, and clients trusted by data owner. A trusted client could have the right to access the data but behaves in an unexpected way [20]. The data owner cannot use traditional data verification methods to ensure data integrity. It is not practical for data owners to download their data to verify its correctness and then upload it again [21]. Moreover, virtualization allows storage providers to use the same physical addresses to store different clients' data. Thus data can be attacked at the physical level as well as the virtualization level [22].

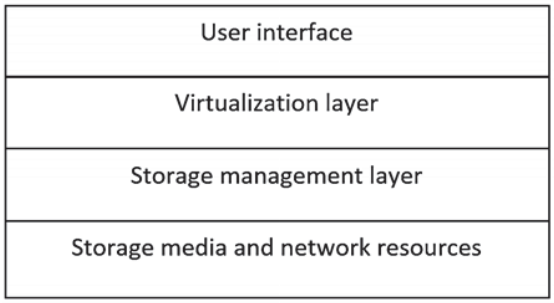

Fig. 1 Four layers of virtualization in cloud storage

B. Design Goals

The main design goal of the proposed system is to provide the cloud data owners with a mechanism that ensures their data integrity and confidentiality. The data owner should be aware of any modification to his data, and should be aware that this modification was done by an authorized user [23]. The second goal is to provide the data owners with fine grained access control. Hence, data owners are supported to enforce an access mechanism to different users according to their role. The data owner should also be supplied with the ability to grant or revoke access rights to his trusted clients [24]. In addition, the proposed mechanism should support dynamic environments in which clients frequently modify the stored data.

C. System Model

The proposed work assumes that the security system is composed of the following parties:

1. A cloud storage provider who owns a large physical data storage and offers his services to different users.
2. A data owner who stores his own data on the cloud
3. A number of clients trusted by the data owner.

Neither the data owner nor the trusted clients are always on line. They frequently come on line to access their data. The cloud provider is always on line. When a client needs to access the cloud data, he should be registered with the data owner and sends his credentials to the owner before getting the access grant. The data owner is responsible of granting access control to different users and is also responsible for offering specific role to each group of users. For instance, a group of users have the right to access definite blocks of data with certain role. We also assume that the data owner and the cloud provider do not trust each other. In addition, the storage provider changes the physical address of the stored data to achieve better resource utilization.

V. Proposed Data Integrity Mechanism

The proposed work aims to eliminate the role of trusted third parties in auditing the data stored on the cloud. We assume that the data owner trusts different clients but he does not trust the cloud provider. For the purpose of this research, we define trust as follows. Entity "A" trusts Entity "B" if "A" believes that "B" will behave exactly as expected, and if not, the gain of "A" will be more than his losses. In order to reach this trust between the data owner and the storage provider the following mechanism is implemented. The proposed mechanism starts after the client sends his data for the first time to be stored at the cloud storage. Then the mechanism is repeated after each time data owner logs off. The data is passed to a hash function and a hash table is created and is sent to the owner. This mechanism is performed through a number of steps, described below.

Step1: Data Storage Segmentation

The virtual space offered to the client is divided into blocks. In this work, we put no restriction on dividing the owner's data to blocks to enable flexibility when dealing with different database structures. Then each block is divided into smaller segments. There are two reasons behind data segmentation before hashing. The first reason is decreasing the overheads of using the hash function by hashing only small segments of data. In very large data storage, the owner's data could be stored on hundreds of Terabytes bytes. However, the size of the modified data may be only few Kilo bytes. Hence, hashing small modified segment decreases the overall time and resources consumed in the hashing process. The second reason is to allow the data owner to provide a structure access control. For instance, a group of users can have the right to

access specific segments of data and do not have the right to access other segments.

Step2 Creating Hash Table

Segments are passed to a hash function SHA-1 or MD5 to create a hash for the data in this segment. The output of this hash function is stored in a hash table with the logical number of the segment. After that, the hash table is encrypted by the public key of the owner, and is sent to the owner. Encrypting the hash table by the owner's private key prevents any malicious node from storing a copy of the hash table. Hence, the only one who can store a copy of a hash function is the data owner. The hash table is further used in the access control mechanism as will be shown in Section VI.

Step 3 Data Owner Check

The data owner now has a hash image of his data encrypted by his public key. The owner will then decrypt the table by his private key and store the table in his own storage. Next time the owner needs to access his data; he will check the hash table. The data owner compares the hash table stored on his physical storage with the hash table stored on his cloud storage. If the two tables are similar then, the owner is certain that his data had not been changed. Fig. 2 demonstrates these steps.

The hash table is then used to create the key that allows the data owner to access his data the next time. Assume that a malicious client has broken the access control to the user data and modified some fields in it. In such case, the provider will automatically send a new hash to the owner. The owner is now aware that his data has been altered. The next time the owner accesses his cloud storage, he will compare his hash table with the cloud hash table. If they are not identical the owner will know that his data has been modified. The action that the data owner can take after that is controlled by the contract between the data owner and the provider. The contract should enforce the provider to restore the owner's data. For the purpose of judgment, the hash tables should be stored in static physical addresses. Hence, hash tables should be safe from security attacks related to virtualization technology.

VI. Access Control Mechanism

The proposed access control mechanism combines the provider private key with the data owner's private key and the hash value of the stored data. Hence, the access control responsibility depends on three values which are distributed between two parties. The first value is the private key of the data owner. The second value is the private key of the cloud provider. The third value is the hash value which is dynamically calculated after each access. The hash value is known by both the provider and the data owner as mentioned in Section V. The idea of using the hash value in access control mechanism has two reasons. First it is a dynamic value; hence it protects the system against playback attack. The second, it proves that the owner is aware with the last operation done to his data.

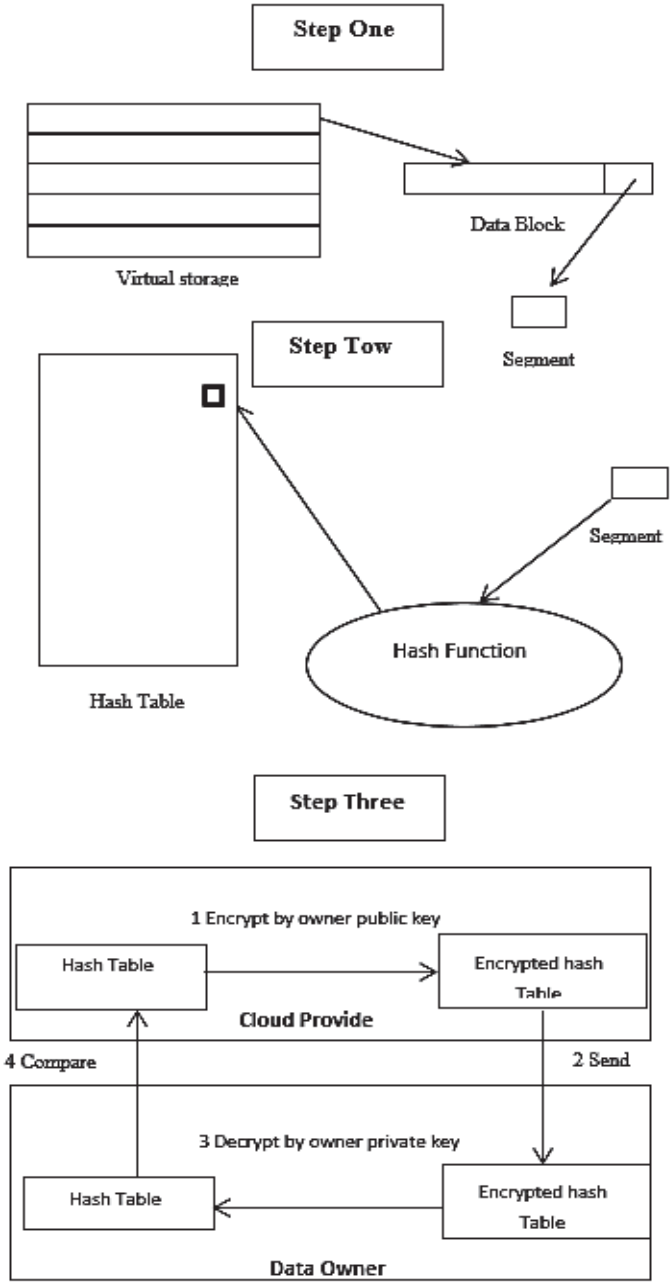

Fig. 2 Three Steps of Ensuring Data Integrity

The access control scenario performs the following steps.

1. A client who is trusted by the data owner requests a ticket from the data owner to allow him to access the data storage. The client sends his ID and the requested key role to the data owner. The authentication server at the data owner's side checks the ID of the client from the stored client database. If the client is trusted by the data owner a ticket is sent to him.
2. The ticket which is sent by the data owner to the client includes (client ID, role key, last hash value). The data in the ticket is encrypted twice. First, it is encrypted by the private key of the data owner and then by the public key

of the storage provider. Then the ticket is sent to the client encrypted by a key driven from the client's password.

3. The client receives the ticket ensuring two security requirements. The first no one can use the ticket but the client as it is encrypted by a key driven from his password. Hence, only the client can decrypt this key. The second is that the client cannot alter the ticket as it is encrypted by the public key of the cloud provider. The client decrypts the ticket and sends the ticket to the cloud provider.
4. The cloud provider receives the ticket ensuring another two security requirements. First, no one but the cloud provider can read the data in the ticket because it is encrypted by his public key. This prevents any malicious node from getting use of the ticket or storing the data owner credentials. Second, the cloud provider will not be able to alter the message as it is encrypted by the data owner's private key. The provider uses his private key to decrypt the ticket.
5. The cloud provider stores a copy of the ticket and then decrypts it using the data owner's public key. Such process ensures two security requirements. First, it ensures that the sender is the data owner because it is encrypted with the owner's private key. Second, it ensures that the data owner is aware of the last modification as the ticket includes the hash table of the last modification.

Messaging between the client and the data owner and the provider can be modeled as follows.

1. C → DO: (CID, R)
2. DO → C: Kc'(Kp'(Kdo'(CID, HT, KR))
3. C → CP: Kp'(Kdo'(CID, HT, KR))

Where C is the client; DO is the data owner; CP is the cloud provider; CID is the client identity; R is the role requested by the client; Kcp' is the public key of the cloud provider; Kdo' is the private key of the data owner; Kc' key driven from client password; HT: hash table; KR: role key.

Fig. 3 presents the messaging between the client and the data owner and the cloud provider.

VII. Evaluation

The efficiency of the proposed work can be evaluated from different perspectives. However, we evaluated the security mechanism against three different possible attacks. The scenarios of the three attacks are listed in the literature as common attacks in cloud data storages [23]. We considered the evaluation metrics as a value that represents whether the malicious attacker can succeed to access the storage or not. First a cloud data storage was created using Microsoft Azure. Microsoft Azure was chosen to evaluate our work because it offers flexible cloud services that can be monitored from both the client side and the provider side.

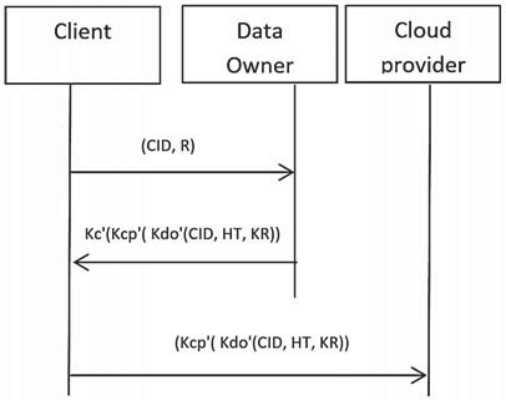

Fig. 3 Access control Scenario

A data base was created using .Net cloud to store data. A table was created with name “students” containing four fields. The data was stored at Azure Blob storage which is able to store any structure of data. The data was then passed to hash function (SHA1). Azure certificates service was used to get a private key for the data owner. An authentication key was produced form the hash value of the stored data. The authentication key was encrypted twice by the public key of the provider and then by the private key of the data owner. Then, the following scenarios were traced by using the proposed security system. The same scenarios are also traced using PasS approach (Privacy as a service) [14] which is reported in the related work. Hence, it is possible to compare between the robustness of the two security approaches.

A. First Scenario: Playback Attack

A malicious node sniffs the data owner's credentials and stores it, and masquerades as if it is the data owner. The malicious node sends the same credentials included in an old message (client ID, hash value) encrypted by the owner's private key and the provider's public key. The access to data was denied. By tracing the provider side, the message was decrypted by the public key of the data owner then decrypted by private key of the provider. The plain text contained an old copy of the hash table. Hence, the access is denied proving that the proposed mechanism succeeded to protect the storage from the playback attack.

The same scenario was applied to a cloud storage that is protected using PasS. The malicious node sniffed the data owner's message and then stored it. After a period of time the malicious node sends the credentials to the storage provider. Theoretically the provider will accept the authentication process, hence the parameters of authentication message are correct. Adding a time stamp would not be efficient because the malicious message could be sent in the same time interval. The proposed approach succeeded to protect the data against replay attack but the PasS did not.

B. Second Scenario (Malicious Storage Provider)

The untrusted cloud storage is the subject of this attack. He receives the ticket from the client and stores it. After the authorized access of the client finishes, the data owner uses

this ticket to access the stored data and modify it. Using Azure data access service, the owner tries to access the stored data. The owner authentication includes a code that compares the hash table stored at the client side and the hash table stored at the owner side. If the two hash tables are not identical the code returns a mismatch alert to owner informing him that his data were altered. Hence, the proposed mechanism succeeded to detect the second attack scenario.

PasS does not include a mechanism to ensure data integrity. Therefore, it is not able to detect the modification of the stored data.

C. Third Scenario

Trusted node attack takes place when a client trusted by the owner performs an illegal action after a legal access to the data storage. Then the data owner detects these illegal actions. The owner claims that the storage provider has violated the contract. Azure authentication service provides the facility to store all tickets sent by clients. At this point, the provider has a copy of the client's ticket that contains the client ID and is encrypted by the private key of the data owner. This ticket ensures that this is the legal client as it includes his client ID. De-encrypting the ticket ensures that it was issued by the data owner because it is encrypted by the owner's private key. Hence, the proposed mechanism is able to detect the attack. Moreover, this mechanism presents valuable evidence that helps in judgment between the owner and the storage provider. This attack was detected by PasS but its reason cannot be specified because PasS does not have the ability to store authentication messages.

VIII. Conclusion

We introduced a mechanism to check data integrity on cloud. The mechanism includes an access control method that combines security for the data owner and the cloud provider. The suggested solution provides cloud data owners with the protection needed to encourage them to store their data on the cloud. The proposed mechanism can be used to protect the cloud data against different attacks from malicious nodes or from untrusted storage provider(s). In addition, the work provides an evidence collection mechanism that can be used in the judgment between the data owner and the storage provider, in case of disputes over data integrity violation.

Reference

[1] Farhan Bashir Shaikh and Sajjad Haider, "Security Threats in Cloud Computing," In 6th International Conference on Internet Technology and Secured Transactions, IEEE. 2011.

[2] B. Meena, Krishnaveer Abhishek Challa, "Cloud Computing Security Issues with Possible Solutions," In IJCST Vol. 3, Issue 1, Jan. – March 2012.

[3] Weiliang Luo, Li Xu, Zhenxin Zhan, Qingji Zheng, and Shouhuai Xu, "Federated Cloud Security Architecture for Secure and Agile Clouds," In High Performance Cloud Auditing and Applications, DOI 10.1007/978-1-4614-3296-87, Springer Science and Business Media New York, USA, 2014.

[4] S Narula, A. Jain, "Cloud Computing Security: Amazon Web Service Advanced," In Proceeding of Computing & Communication Technologies (ACCT), fifth International Conference , 501 - 505 , IEEE. 2015

[5] Richard Chow, Philippe Golle, Markus Jakobsson,"Controlling Data in the Cloud: Outsourcing Computation without Outsourcing Control," In Fujitsu Laboratories of America, CCS 2009.

[6] Beloglazov, Rajkumar Buyya, "Energy Efficient Resource Management in Virtualized Cloud Data Centers," In 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing 2010.

[7] Kresimir Popovic, Zeljko Hocenski, "Cloud computing security issues and challenges," In The Third International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services, pp. 344-349 2010.

[8] Deyan Chen1, Hong Zhao, "Data Security and Privacy Protection Issues in Cloud Computing," In International Conference on Computer Science and Electronics Engineering DOI 10.1109/ICCSEE, IEEE 2012.

[9] Cong Wang, Sherman S.-M, Qian Wang, Kui Ren, Wenjing Lou, "Privacy-Preserving Public Auditing for Secure Cloud Storage," IEEE Transactions on Computers vol: 62 NO: 2. 2013.

[10] Dimitrios Zissis, Dimitrios Lekkas, "Addressing Cloud Computing Security Issues," Future Generation Computer Systems 28, Elsevier 583–592doi:10.1016/j.future, 2012.

[11] Sebastian Gajek, Tibor Jager, Mark Manulis, and J¨org Schwenk, "A Browser-Based Kerberos Authentication Scheme," ESORICS 2008, pp. 115–129, Springer-Verlag Berlin Heidelberg 2008

[12] Shucheng Yu, Cong Wang, Kui Ren, Wenjing Lou, "Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing Shucheng," In proceeding of the IEEE INFOCOM, 2010.

[13] Boyang Wang, Baochun Li and Hui Li Panda, "Public Auditing for Shared Data with Efficient User Revocation," In the Cloud IEEE Transactions On service computing, computing, (Volume:8, Issue:1) 2015.

[14] Wassim Itani, Ayman Kayssi, Ali Chehab, "Privacy as a Service: Privacy-Aware Data Storage and Processing in Cloud Computing Architectures," In Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing, 2009.

[15] Bashkar Parasad, Eunmi Choi, "A Taxonomy and Survey of Cloud Computing Systems," In Fifth International Joint Conference on INC, IMS and IDC, IEEE DOI 10.1109/NCM. 2009.

[16] Khurana Sumit and Gaurav Verma Anmol, "Comparison of Cloud Computing Service Models: SaaS, PaaS, IaaS," In IJECT, vol. 4,2013.

[17] Balachandra Reddy Kandukuri, Ramakrishna Patur, Atanu Rakshit, "Cloud Security Issues," In Proceedings of the 2009 IEEE International Conference on Services Computing, pp. 517-520, 2009.

[18] Lifei Wei, Haojin Zhu, Zhenfu Cao, Xiaolei Dong, Weiwei Jia, Yunlu Chen, Athanasios V. Vasilakos," Security and Privacy for Storage and Computation in Cloud Computing," Information Sciences, 258, 371–386 Elsevier2014.

[19] Xun Yi, Fang Yu Rao, Elisa Bertino, "Privacy-Preserving Association Rule Mining in Cloud Computing," In proceeding of the 10th ACM Symposium on Information, Computer and Communication Security Pages 439-450 New York, USA, 2015.

[20] Meiko Jensen, Jörg Schwenk, Nils Gruschka and Luigi Lo Iacono, "On Technical Security Issues in Cloud Computing," In IEEE ICCC, Bangalore, pp. 109-116, 2009.

[21] Minqi Zhou, Rong Zhang and others, "Security and Privacy in Cloud Computing: A Survey," In Sixth International Conference on Semantics, Knowledge and Grids, IEEE, 2010.

[22] Anton Beloglazov, Rajkumar Buyya, Allocation of Virtual Machines in Cloud Data Centers," In the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing Energy Efficient 2010.

[23] Lanxiang Chen, Shuming Zhou, Xinyi Huang, Li Xu, "Data dynamics for Remote Data Possession Checking in Cloud Storage," In Computers & Electrical Engineering 39, 7, 2413–2424, 2013.

[24] Chi-Chun Lo, Chun-Chieh Huang and Joy Ku, "A Cooperative Intrusion Detection System Framework for Cloud Computing Networks," In 39th International Conference on Parallel Processing Workshops, 2010.