International Journal of Enhanced Research in Science Technology & Engineering, ISSN: 2319-7463
Vol. 2 Issue 4, April-2013, pp: (91-99), Available online at: www.erpublications.com

Real-Time Service Composition and Deployment for Secure Computing in Cloud Environment

R. Ushadevi[1], V. Rajamani[2]
[1]Research Scholar, Department of Computer Applications, St. Peter's University, Chennai Tamilnadu, India
[2]Department of Electronics and Communication Engg., IndraGanesan College of Engg., Tiruchirappalli, India

Abstract: Mitigation attack in cloud environment questions secure computing in cloud environment. We propose a real time service composition and deployment method, in order to access the cloud resources safely in the cloud environment. The identity management, security management, data management services are generated, composed and deployed at runtime. The dynamic nature of the service composition and deployment provides a high security to the service providers and cloud servers. The services composed are selectively deployed in cloud servers; the selection of server is based on randomization technique. The cloud user can access only through the deployed services to access, update, and delete their data, which are out sourced by them. Whatever the private or public data, could be accessed only by the deployed services. This phenomenon makes no difference between the service providers or consumers. The proposed method reduces the internal attack and also reduces the degree of guessing attack.

Index Terms: Cloud Computing, Cloud Security, Mitigation Attack, Service Composition, Data Integrity.

1. Introduction

With the growing internet development and computing technology promise the cloud computing, by providing huge network bandwidth. The high price processors, with set of software services transform data centers to the processing pools forms the cloud environment. The enterprises which have huge volume of data to be processed may not be ready to offer high power processors and computing resources. The service providers outsource their resources to the external world. The services described can be accessed by public or private manner according to the service integrity.

Basically cloud environment is four tier architecture; each tier provides a specific set of services. The first tier provide the software s services, second tier provides the platform for the computing, third provides software infrastructure for the service, finally fourth tier provides hardware as a service for computation.

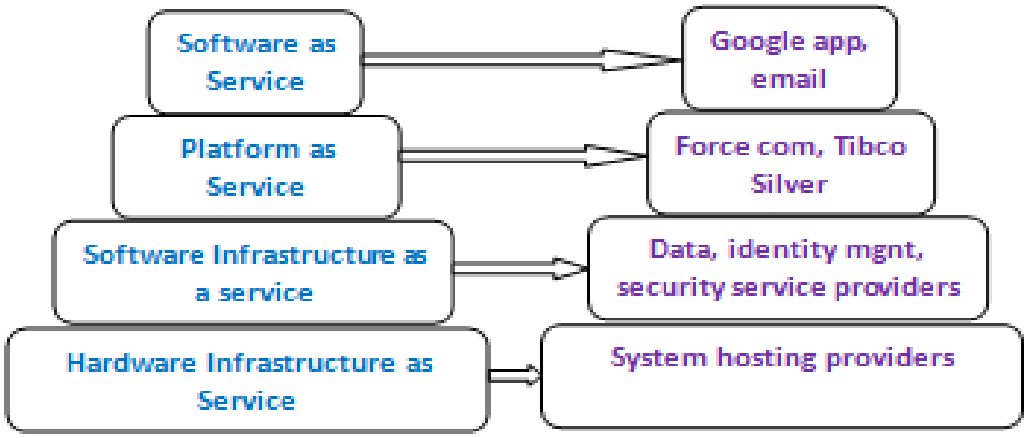


Figure 1: Four Tier cloud computing architecture

From figure 1, itcan be seen that each tier in the architecture has set of responsibility and each provides set of services. In our view there must be at least three tiers in the architecture. The security in the cloud environment is enforced in many ways, but according to the protocol the Third Party Auditor (TPA) maintains the security protocol and whatever the protocol is specified for the security is followed by TPA to provide service to the cloud users. The cloud servers and the service providers define set of security protocols, those protocols to be followed by both the cloud users and the TPA.

The public and private key based security protocol is commonly followed in the cloud environment. Still there are known attacks from the outsiders and insiders of the network. Even genuine registered users generate mitigation attacks to reduce the performance of the cloud environment. To overcome the difficulty in security enforcement, the researchers proposed many techniques, mostly using public and private key mechanism.

The genuine users become attackers at some stage, which could be difficult to identify. The genuine users can predict the service location and the service parameters. They can easily guess the other service parameters which are denied for them. The services in the cloud may be public or private, the public service could be access by any one registered in the cloud environment. But the private services could be accessed only by few users like owner of the data, or service provider. Reading a data in the cloud may be a public service, but deleting is allowed only for the owner of the data.

Data integrity is the way of safeguarding the data from unauthorized access. The integrity constraints should be specified clearly in cloud environment to provide secure access. Data integrity services to be enforced in efficient manner to provide integrity to the data exposed in the cloud environment.

Service composition makes a service complicated in design, and also reduces the probability of guessing the logic of the service implementation. There is always a chance to guess the service implementation or logic, so that the attackers could guess the logic of service and try to generate guessing attack. We implement service composition, which combine two or three services in a bundle and they all selected and bundled at runtime. The bundled services are deployed to provide services to the users.

2. Background

Gossip [1], a protocol specified for the resource allocation in cloud environment. It handles the resource allocation according to dynamically changing resource demand. It works dynamically using local input. The protocol does not require the global synchronization.

A flexible distributed storage integrity auditing mechanism [2] is proposed. It uses homomorphic token and distributed erasure-coded data. The users can audit the cloud storage with very lightweight communication and with reduced computation cost. The auditing ensures strong cloud storage correctness guarantee and fast data error localization. It supports secure and efficient dynamic operations on outsourced data, including block modification, deletion, and append.

Hierarchical attribute based access control in cloud computing [3], specifies the access control protocol, which is based on the attribute and in hierarchical manner. In this the attributes are encrypted in a hierarchical manner according to the structure of the users. It provides multiple value assignments to access expiration time for user revocation. It is based on cipher text policy with attribute based encryption technique.

Attribute-based encryption for fine-grained access control of encrypted data [4] defines, each cipher text has set of attributes and it has a user's decryption key. The decryption key is in form of monotonic tree access structure. The user can decrypt the cipher text only if the user's decryption key and its attributes satisfy the tree access structure.

In ciphertext policy attribute based encryption scheme [5], the encrypt or chooses a tree access policy to encrypt the cipher text. Using set of attributes the decryption key is created. If the attributes associated with decryption key satisfies the access policy the user can decrypt the cipher text using the decryption key. The data dynamics is also important consideration in cloud computing, Q Wang and C Wang [6] has discussed dynamic data storage in cloud computing with public verifiability. They combined BLS based homomorphic authenticator which uses Merkel Hash Tree to profile complete support for data dynamics. Whereas Erway [7] defined a skip list based technique for dynamic data support and Bellare [8] introduced set of cryptographic mechanism like hash, signature functions to maintain storage integrity in dynamic data support. Maintaining

multiple copies or replicas of data in a distributed environment is proposed by Curtmola [9]. They used PDP scheme in extended manner, without encoding each replica separately and also each replicas are maintained separately.

Reed-Solomon codes [10] for erasure correction in redundant data storage systems, which are typically described mathematically by coding theorists, in a way accessible to the programmers who need to implement them. For example, an information dispersal matrix *A*, which does not have the properties claimed -- that the deletion of any *m* rows results in an invertible *n*n* matrix. The purpose of this note is to present a correct information dispersal matrix that has the desired properties, and to put the work in current context.

Privacy preserving public auditing for secure cloud storage is discussed in [11], which propose a secure cloud storage system supporting privacy-preserving public auditing. Further, it enables the TPA to perform audits for multiple users simultaneously and efficiently. Towards publicly auditable cloud data storage [12], proposes public audit ability, a trusted entity with expertise and capabilities data owners do not possess can be delegated as an external audit party to assess the risk of outsourced data when needed. Such an auditing service not only helps save data owners computation resources but also provides a transparent yet cost-effective method for data owners to gain trust in the cloud.

Protocols for Public Key Cryptosystems [13], introduced a protocol for public key crypto system. In this a centralized key distribution system with de centralized key verification system persists the protocol efficiency. In a novel dependable and secure data storage scheme with dynamic integrity assurance [14] a hybrid share generation and distribution scheme to achieve reliable and fault-tolerant initial data storage by providing redundancy for original data components is proposed.

To further dynamically ensure the integrity of the distributed data shares, an efficient data integrity verification scheme exploiting the technique of algebraic signatures is adopted. The proposed scheme enables individual sensors to verify in one protocol execution all the pertaining data shares simultaneously in the absence of the original data. Extensive security and performance analysis shows that the proposed schemes have strong resistance against various attacks and are practical for WSNs.

Keying Hash Functions for Message Authentication [15], use the hash function (or its compression function) as a black box, so that widely available library code or hardware can be used to implement them in a simple way, and replace ability of the underlying hash function. Incremental Cryptography: The Case of Hashing and Signing [16] , initiate the investigation of a new kind of efficiency for cryptographic transformations. The idea is that having once applied the transformation to some document M, the time to update the result upon modification of M should be proportional" to the amount of modification" done to M. Thereby one obtains much faster cryptographic primitives for environments where closely related documents are undergoing the same cryptographic transformations.

Demonstrating data possession and uncheatable data transfer [17], describe a protocol based on this hash function which prevents ‘cheating’ in a data transfer transaction, while placing little burden on the trusted third party that oversees the protocol. We also describe a cryptographic protocol based on similar principles, through which a prover can demonstrate possession of an arbitrary set of data known to the verifier. The verifier isn’t required to have this data at hand during the protocol execution, but rather only a small hash of it. The protocol is also provably as secure as integer factoring.

All the methodologies we discussed in this chapter are mainly discussed about data integrity, encryption standards and maintaining multiple replicas of data. We consider about the replication of service in multiple locations and how they can be composed, deployed and maintained in runtime. We propose a new technique to compose, deploy and maintain multiple copies of services at runtime.

3. Proposed System

The proposed system consists of four different components or users, named cloud users, Third Party Auditor (TPA), Cloud Servers and Service Providers (SP) and is depicted in Figure 2. The service providers are the data owners; the cloud servers are the resource owners where the resource may be processors, storage medium or anything which is highly valuable. The

TPA maintains the identity management of the cloud users, the user may be cloud user or service providers. The TPA has the responsibility to maintain the identity of each user in the cloud environment.

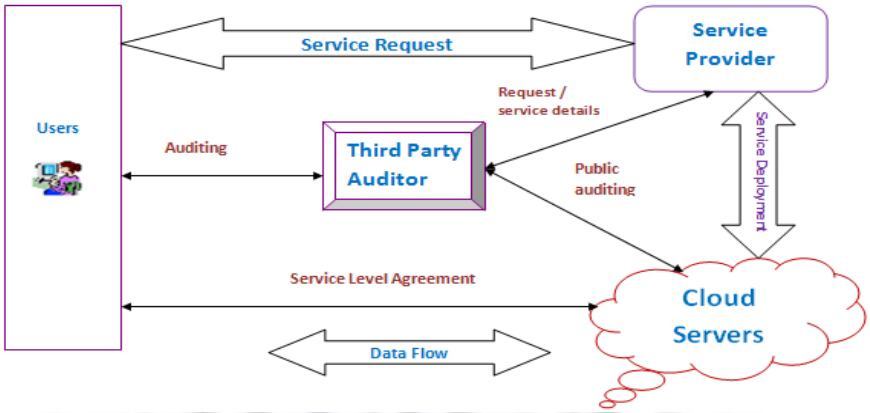


Figure 2: Real Time Service Composition and Deployment Architecture

3.1 Third Party Auditing TPA

Identity of users are maintained using public and private key mechanism, the keys are computed using RICS Hash function. TPA holds the responsibility of identity management. Every single user in the cloud environment have unique public and private key assigned to them at the time of their registration. He will be identified using the public key assigned to him and the private key is to access the data or service allowed to him. At the time of registration the cloud user request the service provider for access and the service provider generates both the public and private key for the user and advertise to the TPA and the user. TPA stores all the keys related to every user, who have registered to the cloud environment.

3.2 RICSH

The public and private keys related to every user in the cloud environment are computed using Random Integer Character Selection Hash function. It generates public and private key, which are unique for every user and updates to TPA. It generates a Group Key at the time of initialization; it maintains. Unlike RSA and Diffie-Helman techniques RICSH methodology have better hacking proof. Due to the nature of randomization, the hackers or attackers cannot compute exact key at any situation. We use two different set of characters using which the key is computed, and we use set of integers using which the randomization performed. At first a group key G_k is selected by the algorithm then a random number R is generated and generated random number is identified prime or not, based on the prime value of the random number a character from one of character set is selected. We use vowels V_s , character C_s as two set using which we compute the Private key Pr_k and Public Key P_k. We store all G_k, R, C in a linear array M. To compute the public key we randomly select a number within the size of array M for three times and we will extract the character at that index and append to the key P_k . In order to compute the private key Pr_k , we randomly select an integer within the size of array M, we select the inverse location of the index from the array M and the character at that location will be append to the private key Pr_k. We do not restrict that the size of the public and private key to the size three. We can extend the size of key up to twenty. The group key and the characters in the both the sets are periodically interchanged. The flexible nature of our key computation process makes the system more secure than other algorithms.

The Random Integer and Character Selection (RICS) Hash function generates the keys as follows:

RICS Function:

Step1: select group key G_k.
Step2: select randomize integer R.
Step3: compute N=Prime factor®.

Step4: if N€P_s then
C=RΩV_s.
Else
C=RΩC_s.
End
Step5:store G_k, R, and C in array M.
Step6: compute Public key P_k as
P_k=P_k+ (l× (RnΩM))
l- Size of array M.
Rn- randomly selected index in array M.
Step7: compute Private Key Pr_kas
Pr_k=Pr_k+ (l× (Û (RnΩM))).
l- Size of array M.
Rn- randomly selected index in array M.
Û- Inverse location of selected index Rn in array M.
Step8: End.

The computed public and private keys are used for both identification and data encryption and decryption process. Whenever a user request a service to access the data stored in the cloud, the identification process will be done and only if the verification process succeed, he will be allowed to access the service.

3.3 Service Composition and Deployment

Any cloud environments have many services, but the way how they are arranged makes the difference. Normally all the services in the cloud environment are composed at the time of development or installation.

The cloud environments have many different functions which are combined in formal way to provide service to the cloud user. For example if the user want to access a service from the cloud , he has to register and access using whatever the token the service provider gives. We propose a different technique to compose the services. The services in our environment are composed at runtime. Whenever a user requests a service, he will be provided an interface to access that service, which is composed on time. Initially the user request the service through the service provider, the service provider performs checking the work load and number of service available based on the metric analyzed, the service provider combines many services and randomly selects a server in the cloud environment and deploys the service for the use of the cloud user. The information about the newly deployed service will be updated to TPA. This process generates many replicas to the same service and user request will be services in earliest time.

The dynamic nature of the proposed system, reduces the degree of attack comes to the cloud environment. The attacker could not predict where the service is running and the flow of request and response. At the end of each session the services deployed will be undeployed, so that the prediction about the service and attacks which are comes to the service is avoided.

Algorithm:

Step1: Identify user requested service SR_i.
Step2: Identify the service locations L.
L=ϕ× (ℂ (s1, s2… sn))
Φ- Set of all locations where the service available from set of serversS in the cloud C.
ℂ - Set of servers in the cloud.
Step3: Identify set of services ISR_i included in SR_i.
$= ∫(SR).
$= Ø×(ϕ× (ℂ (s1, s2… sn))).
Step4: extract service locations SL from L.
SL = $×L.
Step4: for each service in $
SL_i = Rand($×L)
SList = SList+ $($SR_i$) +$SL_i$.
End.

Step5: Attach service in the order and location from SList.
Step6: Combine Services SR from Slist.
Cs = $(SR$_i$)+SList.
Step7: Deploy composite service Cs.
Step8: start Cs.
Step9: Update Cs reference to TPA.
Csf = R ($(SR$_i$) +SList.).
Step10: End.

3.4 Identity Management

The cloud user generates a request to the service provider with his public and private keys and waits for the response. The service provider SP makes auditing with the TPA about the user's public P_k and private key Pr_k. when the users keys are genuine the service provider composes various services whatever necessary to resolve the request and randomly select a server in cloud server, and deploys the service in particular server. An interface to the newly deployed service is shared with the TPA.

4. Results and Discussion

The proposed algorithm produces good results compare to others. Due to the dynamic nature of our algorithm the attackers could not predict where the service is running. The service composition and deployment is performed at each session so that it increases the security level in the system.

Figure 3 shows the time taken by our algorithm to deploy number of services. Whenever a service request arises, the TPA computes the identity of the user who generated the request. Once the identity of the user is validated then the proposed algorithm selects various locations for each services whichever necessary to complete the service request. Services are deployed and started in selected locations. The references to access the service are composed to form a combined service. The combined service is given to the cloud server's and the response to the user is redirected to the cloud server. From this point the user will interact with the cloud server only. The time taken to complete this whole process is the service composition and deployment time. The figure shows that the time taken will increase slightly with the number of services. Still you can deploy hundreds of services in short time.

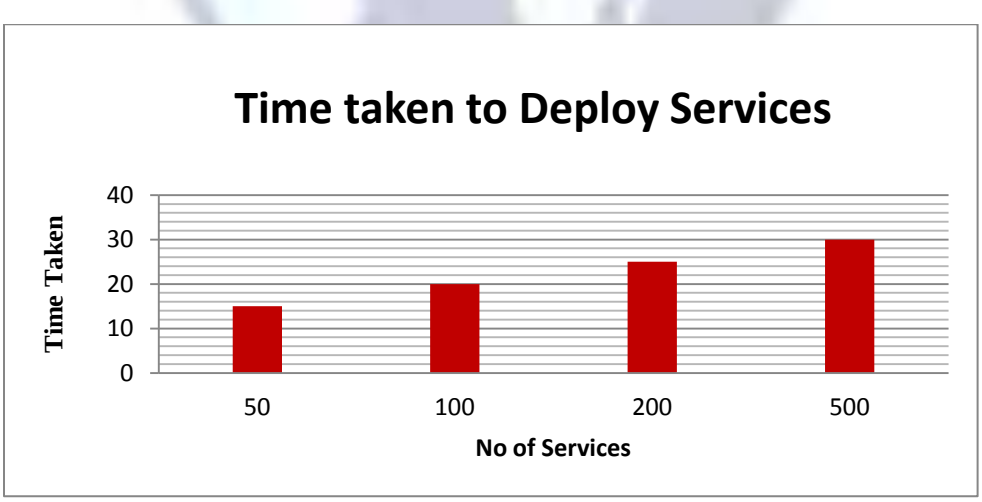


Figure 3: Variation of time taken with number of services of the proposed algorithm

Table 1 Set of deployed service in TPA.

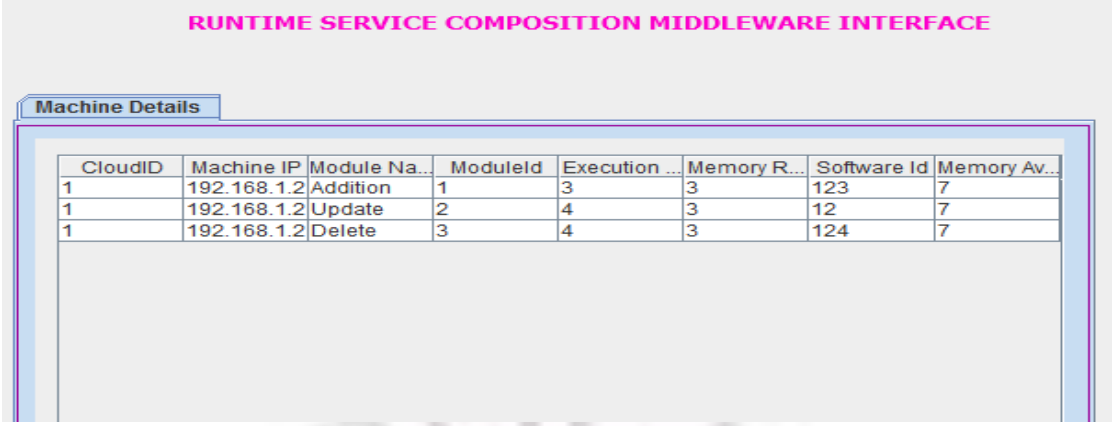


Table 1 shows the graphical interface of TPA. It shows the details of services deployed, module id , time required for execution, memory required, software id and cloud id and memory availability.

Table 2 Service details in cloud Server

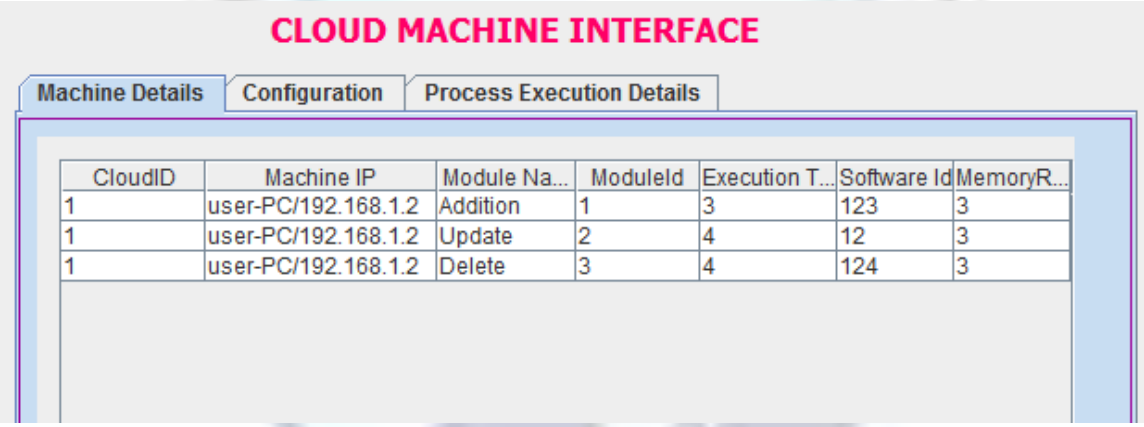


Table 2 shows the graphical interface of Cloud service provider interface. It shows the details of services deployed, module id , time required for execution, memory required, software id and cloud id.

Table 3Output of processed request in cloud server

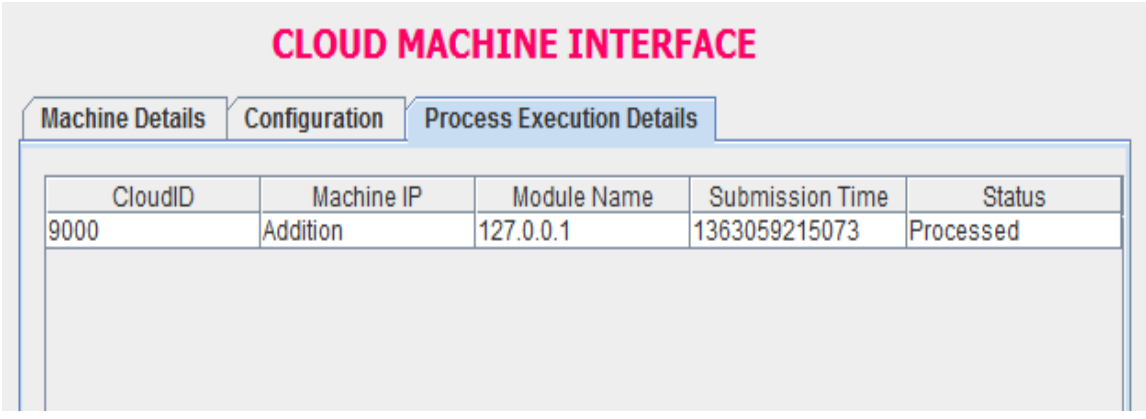


Table 3 shows the job submission details in cloud service provider interface. It shows the service port, module name, submission time, status of the process submitted.

Table 4 Service details in client side

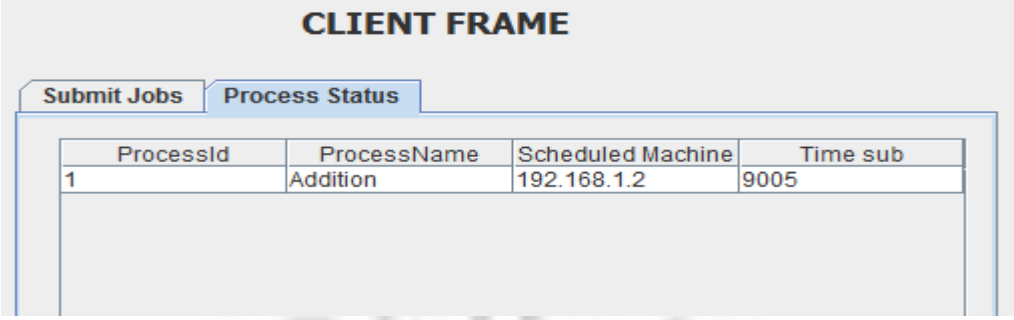
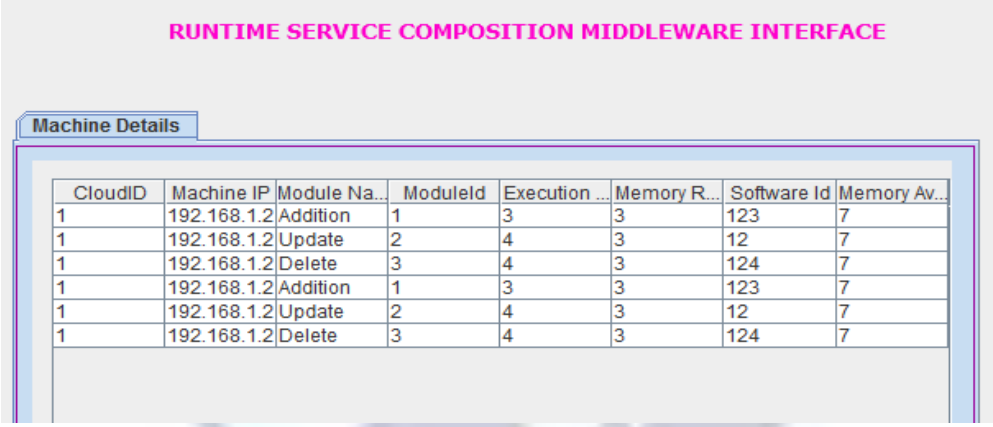


Table 4 shows the result of service submission from client side of cloud environment. It shows that the process id is scheduled in a machine with ip 192.168.1.2 at port number 9005 and its process name is Addition.

Table 5 Service details in TPA



The details of machines in the cloud and services deployed and its module name and module id, execution time and memory required and machine where it is scheduled etc is provided in Table 5.

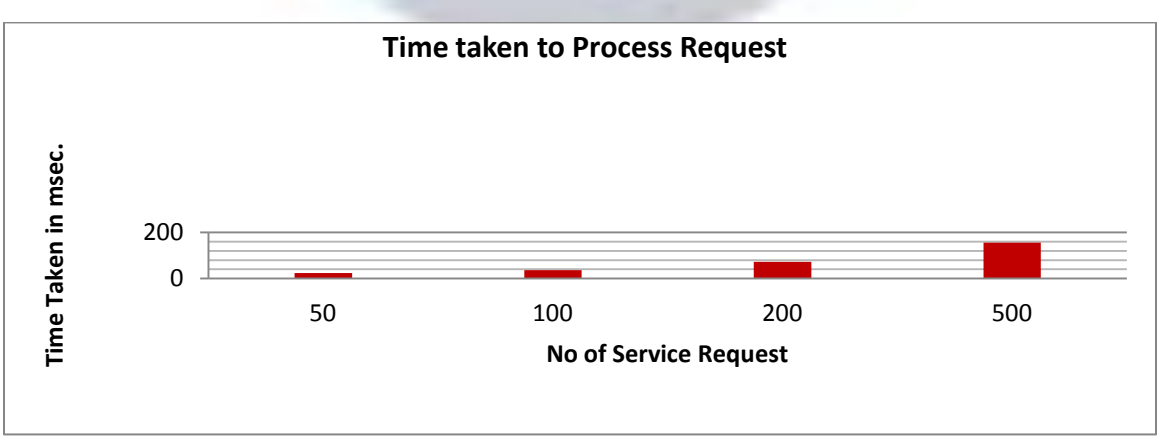


Figure 4: Average time taken to process service request

Figure 4 shows the average time value required for set of service request to be processed. The process time of a request is combination of service composition and deployment , execution time of the service. The processing time of the service is related to the service composition and deployment time, because for every service request the process of identifying the location and deployment and composition of the services has to be done. Only after service composition, the services can be executed to fulfill the user request. The service composition time varies depend on how many services are going to be composed. If we need to compose many services then the time required for service composition also increases. The processing time is also proportional to the number of services. If number of services increases then overall processing time also will increase. The figure shows that average processing time increases with number of service request , because for each service request the service location, composition , deployment and processing has to be done.

Conclusion

The proposed methodology uses public and private key mechanism for identity management. The identity management is done by a third party auditor. The keys generated using RICHS method has very good security in nature; the attackers couldn't identify and compute duplicate keys easily, because randomization of characters used is changed periodically. The runtime composition of services makes the difference with other protocols proposed in this scenario. The proposed methodology groups and combines the necessary services at runtime and changes it at regular interval. The attackers could not identify or guess where the service is running in order to generate guessing attack or flooding attack.

References

[1]. Fetahi Wuhib, A Gossip Protocol for Dynamic Resource Management in Large Cloud Environments, IEEE Transaction on Network and service management, volume 9, No 2,Page(s): 213 - 225 ,2012.
[2]. Cong Wang, Toward Secure and Dependable Storage Services in Cloud Computing, IEEE Transaction on service computing, vol. 5 no. 2, pp. 220-232, 2012.
[3]. Zhiguo Wan, Hierarchical attribute based access control in cloud computing, IEEE Transactions, Information Forensics and Security, 7(2) , pp. 743 - 754.2012.
[4]. Goyal V., Fine-grained access control of encrypted data, ACM, Computer and Communication Security, ACM, pp. 89-98, 2006.
[5]. J. Bethencourt, "Ciphertext-policy attribute based encryption," in Proc. IEEE Symp. Security and Privacy, vol 7, no 2, pages 321-334, 2007.
[6]. Q. Wang, C. Wang "Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing," volume 22,issue 5,pages 847-859, 2009.
[7]. C. Erway, Dynamic Provable Data Possession, ACM Conf. Computer and Comm. Security, vol 8, issue 7,pages 213-222,2009.
[8]. M. Bellare, Incremental Cryptography: The Case of Hashing and Signing Advances in Cryptology, vol 8, pages 216-233 1994.
[9]. R. Curtmola, Multiple-Replica Provable Data Possession, IEEE, Conf. Distributed Computing Systems, vol 22, pages 410-420,2008.
[10]. L. Carter , Universal Hash Functions, Computer and System Sciences, Vol 18, pp. 143–154, 1979.
[11]. J. Hendricks, "Verifying Distributed Erasure-Coded Data," ACM Symp. Principles of Distributed Computing, vol 10, pp 163-168, 2007.
[12]. J.S. Plank and Y. Ding, "Note: Correction to the 1997 Tutorial onReed-Solomon Coding," Technical Report CS-03-504, Univ. of Tennessee, Apr. 2003.
[13]. C. Wang, Q. Wang, "Privacy-Preserving Public Auditing for Storage Security in Cloud Computing," IEEE INFOCOM,pp.355-370,Mar. 2010.
[14]. C. Wang, "Towards Publicly Auditable Secure Cloud Data Storage Services," IEEE Network Magazine, vol 24, pp. 220-232, 2010.
[15]. R.C. Merkle, "Protocols for Public Key Cryptosystems," IEEE Security and Privacy, vol 11,pp.122-134, 1980
[16]. Q. Wang, K. Ren, W. Lou, and Y. Zhang, "Dependable and Secure Sensor Data Storage with Dynamic Integrity Assurance," Proc. IEEE INFOCOM, Apr. pp. vol 11, 954-962, 2009.
[17]. M. Bellare, R. Canetti, and H. Krawczyk, "Keying Hash Functions for Message Authentication," Proc. 16th Ann. Int'l Cryptology Conf. Advances in Cryptology (Crypto '96), pp. 1-15, 1996.
[18]. M. Bellare, O. Goldreich, and S. Goldwasser, "Incremental Cryptography: The Case of Hashing and Signing," Proc. 14th Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '94), pp. 216-233, 1994.
[19]. D.L.G. Filho and P.S.L.M. Barreto, "Demonstrating Data Possession and Uncheatable Data Transfer," Cryptology ePrint Archive, Report 2006/150, http://eprint.iacr.org, 2006.