A Secured Resource Access Management in Educational Cloud Computing Environment

Saley Mato Idrissa, Karimou Djibo and Saley Bisso
Département de Mathématiques et informatique
Université Abdou Moumouni
Niamey, Niger
saleymato@gmail.com; djibo.karimou@gmail.com

Hamadou Saliah-Hassane
TELUQ – Université du Québec
Montréal, Canada
hamadou.saliah-hassane@teluq.ca

***Abstract*— The paper discusses the problems of data security in the Cloud Computing and proposes an approach based on network technologies and algorithms. The main idea is to establish a criterion of trust between the service provider and the client to control data access and updates which are operated by the owner or a third party. The method allows limiting and filtering the access, to detect corrupted data and proposes corrective action in the case of an illegal access to the cloud computing services. Similarly, this approach examines the strategies to secure the resources through a distributed cloud computing.**

Keywords*— *cloud computing, network complexity, collaborative educational environment, resources as services, cloud security, hash function algorithms.

I. INTRODUCTION

The Cloud computing consists of a set of technologies that allows to offer on demand services. These technologies are offered in the form of three types of services: (1) Software as a service (SAAS), (2) Platform as a service (PAAS) and (3) Infrastructure as a service. They are generally grouped into three types of cloud. The first type is the private cloud which can be either internal or external to a given institution but remains the owner of that institution. The second type is the public cloud which in reality belongs to a service provider that offers an infrastructure to institution in order to host its resources. The management and supervision of this infrastructure is the responsibility of the supplier who remains the owner. The third type is the hybrid cloud formed generally of two or several types and ensures the portability of data and applications hosted by the cloud services. Nevertheless, apart from the above mentionned types of cloud, other models of cloud emerge. This is particularly the case of community cloud which is a model put in place by several institutions (communities) to share resources by using cloud services. The use of cloud technology by the institutions raises several questions with consequences heavy regarding data security, confidentiality and integrity. Several institutions find themselves before the obligation to question on how to integrate these technologies into their areas of activities. This distrust observed by the institutions is essentially due to the risks incurred in terms of loss, data alteration and financial impacts. For these institutions the issues related to security are among others:

- Maintain and keep virtualized areas secured in a dynamic IT environment;
- Maintain the existing security levels operational during migration of traditional resources toward cloud services;
- Migrate the machines' local security to a virtual and sometimes remote security;
- Allow actors and collaborators to find the priority objectives, the tools and their traditional environment of collaboration and exchange. Currently, the proposed solutions for securing the data in the cloud are particularly interesting because they provide authentication mechanisms, confidentiality and data integrity. This makes cloud environment less risky to the data. Nevertheless, the solutions can be increased. It is in this context that we propose this approach to strengthen data and cloud infrastructure security.

Our first motivation is to identify the network technologies which provide the possibility to migrate local security solutions toward a virtual and remote security. We rely on techniques used often in networks as well as hash functions. This approach is mainly applied at three levels:
- The first level is an authentication process filter that allows users to connect to the data center and switch between the virtual servers. This level filters access according to certain criteria applied by the servers.
- The second level propagates the authentication of the first level up to the virtual machines. This level uses filtering and hash functions.
- The third level is a process that applies patches to find corrupted data in the event of corruptions.

In this article, we propose an approach that strengthens the security in a cloud computing environment. This model is based on the methods applied in the networks and the methods of access controls that are implemented on the datacenter as well as on the virtual servers that host the data and applications. This approach, which is carried out in the vicinity and by the propagation of the datacenter to the virtual

servers, is different from the existing methods but comes to support them. The proposed platform allows clients connections by authentication to the datacenter. Filters are then applied to detect the unwanted customers. And to propagate the safety criteria on the servers, we use the hash functions. Strategies are put in place to ensure the dialog between the Data Center and virtual servers. Due to the information which is exchanged, it would be possible to apply patches on the virtual servers to secure and restore the data. The interest in applying a model by authentication and by phase is to differentiate the servers access policies from those of applications while supporting the management of corrupted data. This approach provides a great flexibility to enhance the platform safety and security.

The rest of this work is organized as follows: some definitions used and the working environment are presented in sections 2 and 3. In section 4, we present our proposal to secure the platform in several phases. A phase which covers the points of the algorithms that are developed, a second phase which applies filters for access to the datacenter and then a last phase which makes use of hash functions, the networking technologies and algorithms to propagate the criteria of safety measures and apply patches in the event of corrupted data. Sections 5 and 6 deal with the adaptation of the solution to a model of Cloud in secure access. A conclusion ends this paper.

II. Previous Work

Given the popularity rise in cloud services, research in the area of cloud platforms' security are becoming crucial. Certainly, nowadays data migration and institutions infrastructures toward the cloud is done timidly but force is to see that cloud services are growing rapidly. Therefore, it is more than necessary to develop protocols to secure flows and platforms. To get there, in this paper, we have exploited the opportunities offered by network protocols and algorithmic tracks. Indeed, the approaches to security in the cloud are often proprietary. This is particularly the case of Cisco approaches [2][3][4]. It proposes several approaches both material and software (Cisco secureX, Cisco Axa). These approaches are often incorporated by the providers or by the client institution on the cloud services. These solutions integrate an intelligent security control through the network and cloud services. In their work [5], the authors present a model based mainly on the algorithms. They calculate the similarities of tokens and signatures which exist between the clients and the provider of cloud services to secure the data. In the same framework, they apply two complementary algorithms. The first algorithm (Solomon algorithm) is developing a process to recover corrupted data. The second algorithm called the Byzantine fault tolerant algorithm allows you to detect the reasons and causes in case where data on the cloud platform are corrupted.

III. Working Environment

Our working environment is a dynamic system of machines and data comprised of two main entities:

• Users, applicants of cloud services. Their request concerns the storage, the consultations or updates of the data or applications hosted on the cloud;

• And the service provider (in cloud) which ensures client authentication, storage as well as the management of data and applications integrity.

The user U accesses its data x hosted on server X through a control system. The latter is responsible for the user authentication as well as the verification of its data integrity based on a technique that we will define in the sections below.

- *X*: customers data servers;
- *Y*: backup servers for clients data which are on server *X*;
- S: all of the servers to ensure the integrity of the data. $S=\{s_1, s_2, ...s_n\}$;
- *h*: Hash function for data repartition on system *S* ;
- *P:* let $G=(V, E)$ be a graph with $|V|=n$ and $|E|=m$. A path $P(s_i,s_j)_{i,j \in \{1...n\}}=(A,B)$ is defined by $A=\{s_i\}_{i \in \{1...n\} \setminus \{i,j\}}$, $B=\{e_i\}_{i \in \{1...m\}}$;
- *Vertices-disjoint paths:* two paths $P_1(s_i,s_j)$ and $P_2(s_i,s_j)$ are called v*ertices-disjoint paths* if they have no common vertex except s_i and s_j.

IV. Theoretical Approach

We will use a set of *n* servers to ensure data integrity [1]. This security aspect being very important for the customers, it would not be as least important to consider a number *n* very large of these servers. In this case, we easily agree that the system S can be represented by a planar graph $G=(V, E)$, with $V=S$ and *E* The set of connections between the servers.

A. Model Formulation

The data of the user *U* are on the data server with a backup system. The user connects and accesses its data via the control server by pre-calculating a hash *h(u)*. The control server receives at the same time a hash *h(x)* pre-calculated by the data server, a hash *h(y)* pre-calculated by the backup, a pre-calculated hashs $h(P_1)$ and $h(P_2)$. We made a random choice of two servers s_i and s_j then we determine two vertices-disjoint paths:

$P_1(s_i,s_j)=(A_1,B_1)$ and $P_2(s_i,s_j)=(A_2,B_2)$ where $|A_1|=k_1$ and $|A_2|=k_2$.

We distribute data across k_1 vertices (servers) constituting P_1 and k_2 on the vertices (servers) constituting P_2.

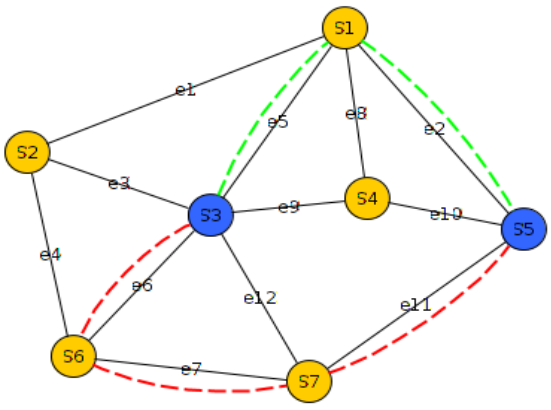

Fig. 1. Example of disjoint paths

$s_i = s_3$; $s_j = s_5$;
$P_1 = (A_1,B_1)$ where $A_1 = \{s_1\}$ and $B_1 = \{e_5,e_2\}$;
$P_2 = (A_2,B_2)$ where $A_2 = \{s_6, s_7\}$ and
$B_2 = \{e_6,e_7,e_{11}\}$.

B. Data access and recovery algorithm

1. Hash (U)
{A: H is a function which allows manage the authentication of the user}
2. If h(u) ≠ H(p) then the user is not the one that it claims to be end
{A: implies a connection error of the user}
3. Otherwise H(U) = h(p) then the user is who he claims to be.
{A: No login error}
If h(x) = h(y) = h(p) then
the data are healthy
END.
4. If h(x) ≠ H(p), then the data on the sever X is not valid.
{A: There is data corruption}
Correct the data x from P1 or P2.
{A: the user data is retrieved}
5. If h(y) ≠ H(p) , then the data on server backup is not valid.
{A: There is data corruption}
Correct the data y from P1 or P2.
{A: the user data is retrieved}

C. Description of the hash function

To the user U join data $d_u = \{x_1,x_2,...x_n\}$, then we can define a hash function h such as:

For general case:
H : $d_u \longrightarrow \{0,1,...m-1\}$
$x_i \longmapsto h(x_i)$
where $H = \sum h(x_i)$, $|d_u| = n$ and $m << n$.

For the servers X and Y :
H_x : $x \longrightarrow \{0,1,...m_x-1\}$
$x_i \longmapsto h(x_i)$
where $H_x = \sum h(x_i)$, $|x| = n_x$ and $m_x << n_x$.

H_y : $y \longrightarrow \{0,1,...m_y-1\}$
$y_i \longmapsto h(y_i)$
where $H_y = \sum h(y_i)$, $|y| = n_y$ and $m_y << n_y$.

For paths P_1 et P_2:
H_1 : $A_1 \longrightarrow \{0,1,...m_1-1\}$
$s_i \longmapsto h(s_i)$
where $H_1 = \sum h(s_i)$, $|A_1| = k_1$ and $m_1 << k_1$.

H_2 : $A_2 \longrightarrow \{0,1,...m_2-1\}$
$s_i \longmapsto h(s_i)$
where $H_2 = \sum h(s_i)$, $|A_2| = k_2$ and $m_2 << k_2$.

For example, we can consider that the value of the hash function is the size of user's data on the server. We can also consider that to access his data, this user presents the value $h(u)$ that the control server assigned to him/her during his/her last disconnection. To significantly reduce the complexity of algorithms, we will consider the following hash table:

TABLE I. HASH TABLE

Index	***h***	P_1	P_2
u_1	$h(u_1)$	P^1_1	P^1_2
u_2	$h(u_2)$	P^2_1	P^2_2
……	……	……	……
……	……	……	……
……	…….	…….	…….

Algorithm of hash (U)
calculate H_x in $h(x)$
calculate H_y in $h(y)$
determine P_1 and P_2
calculate H_1 in $h(p_1)$
calculate H_2 in $h(p_2)$
IF $H_1 = H_2$ THEN
$h(p) = H_1$
return $h(x)$, $h(y)$, $h(p)$
return $h(x)$, $h(y)$, $h(p_1)$, $h(p_2)$

D. Determination of disjoint paths

It is a linear time algorithm which allows to find the maximum number of paths to disjoint summits between any two vertices s and t in a planar graph. In the planar graphs non oriented, this problem is most often resolved by applying flow techniques with a complexity around $O(n^{3/2})$, $O(kn)$ and $O(n)$. The best known algorithm is based on the technique of divide and rule and its complexity is around $O(n \log n)$ [1].

Let be $G= (V,E)$ a Non-oriented planar graph, we transform the initial graph G in a directed graph. For this, we replace each edge $\{u,v\} \in E$ by the arcs (u,v), and (v,u), and each edge $\{s,v\} \in E$ by (s,v) only. We note A, the set which contains all the arcs. The problem is to find two disjoint summits (s,t)-paths (oriented from s to t).

The algorithm consists of a loop on all the arches e_1 ,..., e_k (in an arbitrary order) leaving s. For any e_i, $i \in [1,k]$, it tries to establish a (s,t)-path using the technical research in depth. At each step, all the arches leaving the current summit are searched from the right toward the left (with the exception of the reverse of the main arc of the current search path). The i^{th} iteration ends when one reaches t or when it returns to s (by backward steps). We must consider the conflicts of the path of the current research with itself and with the other paths. We resolve a conflict by doing backward steps. The idea is to manipulate all conflicts so that it removes any arc of G once it performs a return back with him. It performs a return in the

background whenever an occupied summit (s it is internal to a (s,t)-path already achieved) is reached on the left side. If the path of the current search reached a peak occupied by another path from the right, it resolves the conflict by the following manner:

Let be *v* the summit where the conflict occurs, *p* and *q* designating the segments of the other path from *s* to *v* and *v* to *t*, respectively, and let be *r* the path of the current search (see Figure 2).

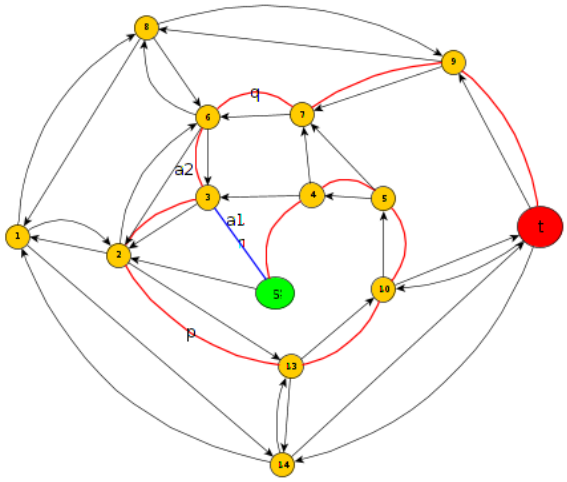

Fig. 2. Example of disjoint paths Search

We concatenate now *r* with *q* and take *p* as the new path of the current search. As in the first case, it performs a backward step, it eliminates from *G* the corresponding arc and continues the search with the new path. Of the fact that we are not able to deal with the path conflicts in the current research (where we reach a summit occupied), we want to avoid these conflicts in advance. To do this we maintain a time counter overall and for each summit a local timestamp (*time_stamp*). The global time counter is incremented by *1* each time that the path of the current Search Change. The timestamp of a summit takes the value of the global time counter when this summit becomes the main summit of the current research path.

Every time that we consider an arc in order to go forward, first, it compares the two timestamps of its two summits. If the two are equal, it eliminates this arc, otherwise it uses it to go forward.

Determine Max (s,t)Paths Algorithm
time_counter := 0 ;
FOR i:=1 TO k DO right_first_search(ei) ;
PROCEDURE right_first_search((s,u) ∈A) ;
time_counter := time_counter + 1 ;
let the current path be s → u ;
v := u
REPEAT
time_stamp(v) := time_counter ;
IF the current search path touches some path at v from the left
THEN perform a backtrack/ remove step
ELSEIF the current search path touches some path at v from the right
THEN time_counter := time_counter + 1 ;
let p be the segment of this path from s to v, and let q be the remaining segment ;
connect the current search path with q ;
let p be the current search path ;
perform a backtrack / remove step ;

ELSEIF at least one arc leaving v is not yet seached (except of the reverse of the in- going arc)
THEN let (v,w) be the first such arc to appear after the in-going arc in reverse clockwise ordering ;
IF time_stamp(v) = time_stamp(w)
THEN remove (v,w) ;
ELSE go forward via (v,w)
ELSE performe a backtrack/ remove step
UNTIL v ∈{s,t} ;

See [1].

Given that this algorithm returns the maximum number of disjoint paths between two vertices *s* and *t*, we use it in the following algorithm to determine the two paths P_1 et P_2.

Determine P_1 and P_2 Algorithm
i= random(0..n) //n : number of servers on S
j= random[(0..n)\{i}] // i ≠ j
P_1 = first path of ***determineMaxPaths*** *(s_i,s_j)*
P_2 = second path of ***determineMaxPaths*** *(s_i,s_j)*
return P_1 and P_2.

If we observe well an example of a cloud environment, we can easily see that a machine can use another machine's services. Particularly in the case where a laboratory equipment is used on line. It is to add a module for controlling access to these resources. The control will be function of the type of user access. As in all the clouds generally, we are going to separate the storage system of other services (on line labs, on line libraries…).

Regarding the first aspect (storage) it is to ensure user authentication and data integrity. We can say that a user has all the rights on its data, therefore there is no management role at this level. However the access rights and users' roles are different according on the groups to which they belong. In the field of education, a student has not, most often, the same rights as the teacher. For example, a teacher of network security course can access the cloud for a virtual network and give access to a student group to detect the security flaw.

As well the control system filled three functions:

- To ensure the authentication of users;
- Check and ensure users data integrity during each access;
- Allow users accessing resources in a cloud according on their rights.

To achieve this solution, we must separate the data system and the resources system. In this case the server *X* is unknown in advance, it is rather a parameter in the Algorithm *IV.B*.
So we have:

- $Q = S \cup R_i$ where R_i = all the machines of the resources system;
- $X = q_i \in Q$, q_i is the machine that the user requests to access;
- A new integrity system $S = Q \setminus \{q_i\}$;
- And *Y* does not change.

V. Practical Approach

We used openvz as cloud server. The figure 3 shows explicitly the servers provision in this last. However it should be noted that the Control Server (control on the figure 3) is the only one which access to Internet because we have used the iptables commands to block the other servers. Data Access and Recovery Algorithm supports two aspects when a user requests to be connected. The first aspect is the authentication. To facilitate the implementation of the model, we have used a simple authentication system (user and password). For the second aspect (Integrity management), we used SSH to access server *X* and the s_i. After we use scripts to allow user to connect and disconnect to its data.

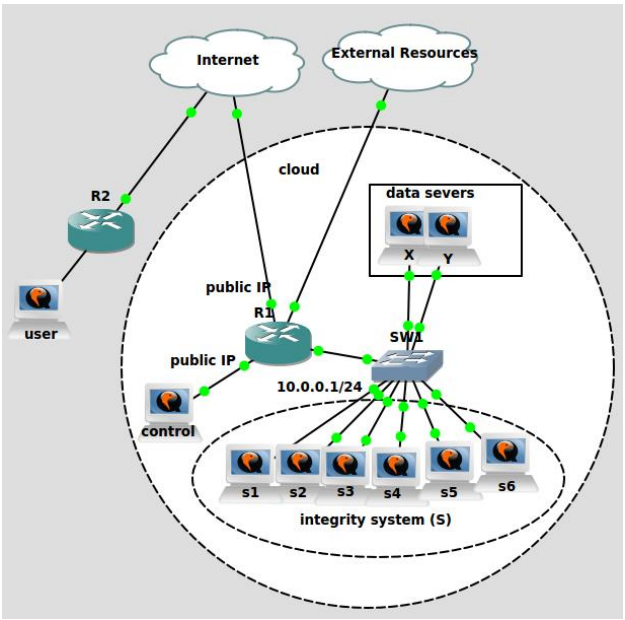

Fig. 3. Secure cloud model Architecture

After all the checks, the user *i* has access to the server *X* (only his partition: *X/home/useri*). The dialog between the user and the server is done in passing by the control server with an FTP client (proFTP) because, as a reminder, the server *X* is not accessible via Internet. This is an important safety factor. It should also be noted that theoretically we have used graph course method to have the servers that will be responsible to ensure user's data integrity at each time he/she connects and disconnects. In practice, however, we have used a random function which allows you to randomly select three servers of the system of integrity *S* to accommodate the three partitions of the user data. The numbers returned by the Random function correspond to the servers numbers. For example if the function reviews $n_1=12$, $n_2=16$ and $n_3=24$, then it will be the servers s_{12}, s_{16} and s_{24}. The numbers (12, 16, 24) of these three servers are backed up in a file which is consulted when the user requests to connect again. When the user requests to disconnect, it deletes the following file to avoid a useless occupation of the disk space on the different servers.

To materialize the fact that the integrity system lodges the user's data on two distinct groups of servers s_i (let us recall of the notion of disjoint paths), we have simply used a backup system Rsync. This allows the backup of the server *X* on the server *Y* and the backup of the first group of s_i, that is to say s_{ai} which constitute the nodes of the path P_1, on the second group of s_i, that is to say the s_{bi} constituting the nodes along the path P_2. We have installed openVz server which has allowed us to build our work environment. We have installed openssh server or client (on all servers including those of the system of integrity) to allow a secure execution of scripts on the servers constituting the cloud. We used proFTP to allow users to manage the transfer of their data on the server *X*. We have installed Rsync to automate the backup of all servers requiring a backup.

TABLE II. INSTALATIONS SUMMARY TABLE

servers	ssh-server	ssh-client	Rsync	proFTP
control		x		x
Server X	x		x	x
Server Y	x		x	x
S_i	x	x	x	x

VI. Conclusion

The migration to the cloud is more than necessary for businesses and even for education and research institutions. This allows a gain not negligible from financial and temporal view point. However, the future users of the cloud have a certain reluctance to this migration due to the guarantee uncertainty about their data integrity and availability. To go in the direction of the prime criterion (integrity) and bring a more to existing work, we relied on a cloud environment in the form of a planar graph. This has allowed us to apply the methods of graph traversal, quite complex but to linear time, already existing as well that a simple hash function to ensure user authentication and his data integrity.

REFERENCES

[1] H. Ripphausen-Lipa, D. Wagner and K. Weihe, "The vertex-disjoint Menger Problem in planar graphs", Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms, 1993, pp. 112-119.

[2] Cisco Cloud Security Accelerates Cloud Adoption http://www.cisco.com/c/en/us/solutions/collateral/security/cloud-web-security/white_paper_c11-674558.html.

[3] Cisco ASA 5500-X Series Next-Generation Firewalls. http://www.Cisco.com/go/asa.

[4] Cisco Cloud Security: http://www.cisco.com/en/US/netsol/ns1066/index.html.

[5] P .Dhanalakshmi, V .Ramesh, "Remote Data integrity in cloud security services", International journal of emerging technology and advanced engineering. Vol.3 January 2003.

[6] A. Juels, A. Oprea, "New Approaches to Security and Availability for Cloud Data", Communications of the ACM, Volume 56 Issue 2, pp. 64-73, February 2013.