Byzantine Resistant Reputation-based Trust Management

Amira Bradai* Walid Ben-Ameur* and Hossam Afifi*
* Telecom SudParis (TSP)
Institut Mines-Telecom
Evry, France
amira.bradai@it-sudparis.eu, walid.benameur@it-sudparis.eu, hossam.afifi@it-sudparis.eu

***Abstract*—Cloud computing is very useful for improving distributed applications performance. However, it is difficult to manage risks related to trust when collaborating with unknown and potentially malicious peers. Besides, trust evaluation is the target of dishonest behaviors trying to disturb the control process. In this paper, reputation-based trust management models for cloud computing are proposed. These Peer-to-Peer (P2P) reputation models are based on the interaction between peers. Using evaluations and feedbacks, a central entity can estimate the trust of a given peer. Three approaches are proposed to estimate the trust: PerronTrust, CredTrust and CredTrust-trust. They are studied, simulated and compared between them and to two existing methods for trust under several attack scenarios. Our analysis clearly shows that the third approach CredTrust-trust combining the concepts of trust and credibility in an appropriate way is the most efficient to avoid malicious behaviors and to guide and advise future executions in the open cloud in term of selecting the dependable and reliable peers in cloud environment.**

***Keywords*—trust management; reputation; credibility; computing.**

I. Introduction

Cloud computing proposes efficient methods for service delivery. It has however several security problems among which the trust in the execution platform. In many cases, customer doesn't know how trustful the remote cloud peer can be. According to [1], there are three major cloud service models:

• Infrastructure-as-a-Service (IaaS): is a cloud computing service offering on demand processing, network and storage. Here the service provider role is to manage his machines. He can provide controls in place regarding how machines are created, memory used, time and performance measured for clearing house procedures. Trust has to be given to the provider as a whole.

• Software-as-a-Service (SaaS): is a software provided in the form of service and not in the form of a computer program. Cloud providers operate application software in the cloud and users access the software from client frontends. Cloud users do not manage the cloud infrastructure and platform where the application runs. The trust problem is how to manage the access to applications (establishing controls and policy models and trusting the operator)

• Platform-as-a-Service (PaaS): is a platform service hosted by an operator and accessed from internet. Cloud providers deliver a computing platform that typically includes operating system, programming language execution environment, databases, and web servers. The primary trust problem is on protecting data (storage as a service) and the ability to encrypt the data.

Cloud computing is usually deployed in one of three scenarios:

• Private clouds built for the exclusive use of one client, providing the utmost control over data, security, and quality of service. Here trust can be totally granted to the service.

• Public clouds include Amazon Elastic Cloud Compute, Google App Engine, run by third parties and applications from different customers, are likely to be mixed together on the cloud's servers, storage systems, and networks. Trust in this case is to be built and is the main scope of our work.

• Hybrid clouds that combine both public and private cloud models can help to provide on-demand, externally provisioned scale services.

In a Cloud Security Survey [2], 32% of enterprises are studying the opportunity of moving applications in hybrid clouds (10% in production, 21% in implementation and 24% piloting). However, the hybrid cloud is the most critical in terms of identity management, open client, location awareness, metering, management and governance. Cloud computing systems offer infrastructures for applications shared by multiple tenants. These tenants are with different security domains. Moreover, enterprises rely on the security processes and algorithms put in place by providers, while not being able to verify the security of their applications.

Systems like Cloud@Home [3] and Nebulas [4] discuss deploying cloud services in a distributed volunteer way. In fact, deploying cloud services in such systems comes with advantages and drawbacks: Hybrid execution improves usage of available resources. It provides scalability and low cost deployment. But, malicious parties in public cloud may take advantage of the system by deliberately performing poorly and being dishonest or claiming more running time for monetary benefits. Moreover, the job submission strategy must be wise to choose the proper peers for submitting jobs. A malicious group can also subvert the system and cause disastrous results. These disadvantages can be addressed with a reputation based trust management system, which can effectively filter out poor performing and malicious nodes.

Trust management is increasingly attracting the attention of

COLLABORATECOM 2013, October 20-23, Austin, United States

DOI 10.4108/icst.collaboratecom.2013.254048

security experts. This concept has been studied in different disciplines from economics to psychology, from sociology to information and computer science. It is a major issue in the open environment because participants are usually unknown to each other since they belong to separate administrative domains [5]. Trust can play an important role in the hybrid cloud since it is an uncertain and a risky environment. Thus, reputation based trust management is a specific approach to evaluate and control the trust. For controlling trust, a good idea is to use root trust entities. Amazon and eBay businesses [6], rely on the broker network's trustworthiness and reliability. However, trust evaluation and control can be the target to attacks. Adversaries with dishonest behavior can affect the global trust evaluation process. That is why, we propose a defense mechanism for trust evaluation.

In this paper, we develop a dynamic peer to peer reputation model. We aim to detect possible cheating behaviors both in the private and the IaaS public cloud scenarios. For this purpose, we don't use the reputation directly to identify malicious peers but to feed three centralized trust models. The first model uses directly reputation values to compute trust vector. The other models consider the credibility to refine the estimated trust and detect cheating on trust evaluation. Our proposal is evaluated via simulations with potential attack scenarios. The evaluation considers several attack scenarios such as grouped attack, non-cooperation in trust evaluation and falsification of trust results. We show that our refined algorithm is resistant to all different behaviors of various kinds of peers. Generally speaking, we believe that although our mathematical models are tailored for a cloud context, they can be generalized to other distributed peer to peer applications.

The rest of the paper is organized as follows: Section 2 exposes some contributions on trust and collaborative intrusion detection. In Section 3, the system architecture is described. Section 4 describes the proposed reputation model and algorithms for trust management. Section 5 shows some simulations results and comparison with existing works. Finally, the paper concludes and lists some new ideas that can be fetched.

II. Related Works

In this section, we enumerate some contributions on trust management and existing reputation-based trust models.

A. Trust Management

Trust can be seen as the general confidence in a person or a thing. Generally, it is evaluated by values on a scale from zero to one. In [9], authors explained that there are four major parts concerning trust management:

- initial trust can find its root in social aspects. Marsh, defining trust as a social phenomena, was one of the first authors to introduce a computational model for trust [10].
- trust metrics: can be binary state to express trust and distrust (0 and 1 or positive and negative), opinions or probability metric. They can be global (evaluation from all the users in the trust network) and local (evaluation between two specific users in the trust network)
- trust propagation (how trust is managed when transitivity is concerned), we can have two operations : concatenation and aggregation,
- trust management architecture: trust can be implemented as centralized, ad hoc or a mix of them.

In a trust system, nodes can rely on their own experience with a given node to assess reliability. In addition to personal experience, a node can use the experiences from other peers (indirect experience) obtained via recommendations. Recommendation systems are hence important (social networks, e-commerce, ...etc.). This system seeks to predict the rating or preference that users would give to an item they had not yet considered [7]. This system helps in the evaluation and the propagation of trust in various networks using trust management.
In general, collaborative filtering is the most popular approach for building recommendation systems in social environment [8]. Several trust management protocols [12], [13] and [14] have been proposed for network security, data integrity, and secure routing in different fields. In [14] a group-based trust management scheme for clustered Wireless Sensor Networks was proposed. This protocol reduces the use of memory for storing trust scores and minimizes the cost associated with trust evaluation of distant nodes compared to other works. Fuzzy logic was introduced to trust models in [15] focusing on the trustworthiness of sensor nodes. It was used to send trusted data between sources and destinations but didn't consider the overhead due to trust in sensor networks.
Reputation is a concept closely related to trust. It is what is generally said or believed about a thing. Reputation is seen as one measurable means by which trust can be built, since an entity can trust (distrust) another based on good (bad) past experience and observation as well as collected referral information about its past behavior.

B. Existing Reputation-based Trust Models

The aim of reputation-based trust models is to identify the trusted nodes. These models use a simple process for the selection of nodes. The first step is to rely on their own experience with a given node to assess its reliability (local trust). This trust can be directly assessed by taking into account several factors such as: the quality of previous interactions, their number and the satisfaction obtained after each interaction. After that, nodes can use others reputation.

Recently, studies have focused on reputation mechanisms and trust systems specific for P2P like applications [17] (PeerTrust), [18] (EigenTrust) and others for social networks like [16](Semantic Web). In eBay's reputation system, a well known system, buyers and sellers can rate each other after each transaction, and the overall reputation of a participant is the sum of these ratings over the last 6 months.

EigenTrust model computes a global trust metric using system-wide information. The approach in [18] is based on the notion of transitive trust and addresses the collusion problem by assuming there are peers in the network that can be pre-trusted. However, no credibility concept is proposed in EigenTrust. In [17], authors calculate the credibility of peers based

on the feedback after each interaction and use trustworthiness factors: the feedback received by other peers, the number of transactions and the credibility of sources. CuboidTrust [19] is a model of trust for global peer-to-peer based on reputation, which is built around four relations based on three trust factors: the peer contribution in the system (Resources), peer trustworthiness (calculated from feedbacks) and quality of resources. Each of these factors is represented by a vector form of a cuboid coordinates x, y, z. Distributed models in P2P face the problem of subjectivity as the parameters are evaluated in peers.

Several studies in research and industry combine trust management and cloud computing environment [20]. Some trust models in grid computing apply trust for enhancement of resource allocation [21] [22]. In [23], authors include an independent trust management module on the top of other security modules. They provide trust strategies respecting cross-clouds environments. In [24], the trust model for cloud resource is based on a security level evaluator (by authentication type), a feedback evaluator and a reputation evaluator.

The proposed trust management in cloud computing are based on a simple arithmetic sum like in [24]. The models proposed for P2P and distributed network have not been tested in cloud computing environments. [25] presents a trust model to solve security issues in cross-clouds environment. The model is composed of three parts: trust decision for the customer and the provider, recommendation and trust update. The results of the presented works for trust management in cloud are not based on solid theoretical foundation. It is necessary to build a suitable solid foundation for trust management. In [26], a formal trust model is given for grid computing. The architecture combines trust based authentication, job request and recommendation for trust update. This work is not implemented. Previous work for trust management in cloud computing are not specialized on how to take efficiently the recommendation. In [27], we use some algebraic methods to evaluate the trust in multi-domain cloud based applications. Our model in this paper compute global trust needed for submitting tasks in public cloud taking into consideration the credibility factor and the history of previous interactions.

III. System Overview

In this section, we first describe our network architecture. Then, we describe the possible behavior of peers in the system following the BAR (Byzantine, Altruistic and Rational) model.

A. System Architecture

We start from the assumption that IaaS cloud is used to remotely execute some tasks of an application. Parameter Surveys Applications (PSA) is an example of applications executed in such hybrid environment. This application is composed of many executed tasks, one part in the IaaS and another one in the local resources of the enterprise (the private cloud). So, let us assume that there are N peers (a finite set of local resources of the enterprise and peers allocated from the IaaS cloud computing) executing tasks for the same application, a portal and a scheduler shown in Fig. 1 :

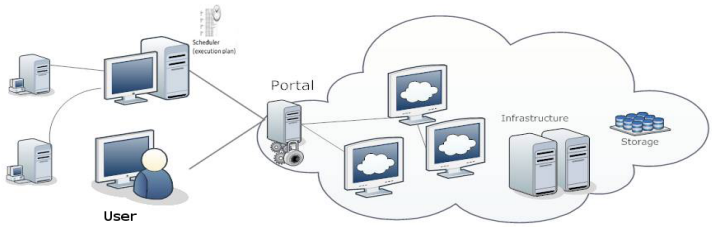

Figure 1. System overview

- The portal represents the interface to execute the application. It brings the other parties together. The portal distinguishes between two types of peers: local and remote.
  The local peers are supposed to be in a trusted and protected location (but they can launch internal attacks on trust evaluation (cheating)).
  Remote ones are supposed to be in different domains, that are not necessarily well protected.
- The scheduler organizes the application's execution. It makes the execution plan.
- The user submits the application and fixes the deadlines and execution time with the portal.

Fig. 2 gives more details about the architecture of our system. Reputation data is needed for the trust evaluation of the peers are stored in the portal. It stores the database of collected reputation. Its task is to compute the trust value for each peer and make the decision. For that, the portal has three modules:

- Reputation collector: responsible for retrieving local reputation vectors.
- Trust manager: responsible for calculating the global trust vector. We can notice in this module that the trust is performed in a dynamic and centralized way (in a trusted entity).
- Decision maker: responsible for deciding for current and future execution. The decision is based on the final trust vector. It consists on guiding the scheduler in the application organization. It means that if the trust score is good, the scheduler will keep the peer in the execution plan.

In our system, each peer has three modules and local storage on its executer manager:

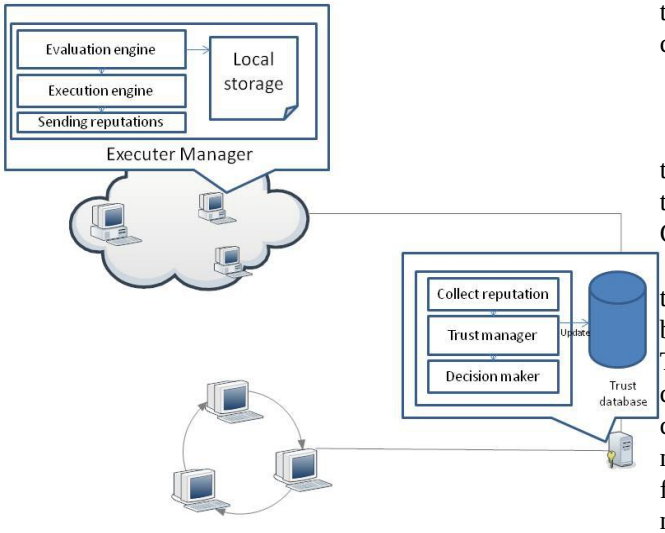

Figure 2. Architecture Details

- Evaluation engine: responsible for a cyclic update of evaluation using the model explained in the section IV-A.
- Execution engine: responsible for the task of execution and result delivery to other peers.
- Reputation sender: responsible for sending evaluations to the portal.

B. Some Possible Peers Behavior

The executer in our system can have two possible behaviors: a normal behavior or a cheating behavior. We can clarify the behaviors by using the BAR model recalled below.

Definition 1: BAR model [28], Byzantine Altruistic Rational model (known as BAR) is a model of computer security, distributed systems used to serve as an error detection model. Currently it is mainly used in P2P systems. Thanks to this model, a peer can be classified into one of three categories that represent the BAR model, namely:

- Altruistic: the peer is considered as a peer working accurately according to the protocol.
- Rational: the peer is only interested in optimizing the use of its resources. It does deviate from the protocol used if it considers that because of using this protocol, its performances decrease.
- Byzantine: (or malicious) peer does not follow the used protocol, either because it is compromised (or not well configured) or because it follows a power optimization of resources that is not the same as for rational peer.

In our system:

• Normal behavior: this corresponds to the behavior of Altruistic peers.

• Cheating behavior: Peers that are rational or byzantine. This type of peer adopts resource optimization strategy or will not respect the reputation management model. they can choose not to cooperate in the evaluation process. The probability to have cheating local peers is less than to have cheating remote ones.

IV. Proposed trust management system

In this section, we propose a reputation model and three trust management algorithms based on algebraic calculations: PerronTrust (can be called Power trust), CredTrust and CredTrust-trust.

The first approach is basically a power method to compute the trust vector. Since the convergence of the algorithm is based on Perron-Frobenius theorem, we call it PerronTrust. To make PerronTrust more resistant to attacks, we add a credibility parameter. So, the trust is calculated based on the credibility of peers. This idea will be illustrated in CredTrust model. To take into account more sophisticated attacks, a further enhancement is proposed leading to CredTrust-trust model. Details will follow in Section IV-B.

All these trust models use the same reputation model described in Section IV-A. The reputation values are maintained in the trusted entity "Portal" that can compute the credibility and the trust in an efficient and objective way. The advantage is that this entity is a trusted one and can decide the future of the application (avoid the peer subjectivity problem).

A. Proposed Reputation Model

The reputation of peers can be estimated based on local observations of their behaviors. So, after the interaction between peers, each peer will give reputation values to the verified peers.

This reputation value in the context of execution on clouds is based on each interaction in the past. We can notice the importance of some factors in the evaluation process: the performance of the peer in terms of time taken, the correctness of the returned results and the crashes experience. Cloud peers can be prone to errors or may be byzantine or rational. When each peer ensures that the returned results are correct and the time respected, the risks are reduced. If the results are not correct and the peer is cheating, the task is resubmitted to other peers. This ameliorates the quality and security of execution.

Definition 2: Let T be a reputation matrix. In order to have the global view of the reputation management, we construct this matrix of peer to peer reputation scores. It is initialized to be 0.5 (ignorance).

The result of the reputation evaluation process is this matrix, T containing peer to peer reputation values between peers.

Definition 3: The P2P reputation score $T_{ij}(t)$ is the evaluation of peer i to peer j at time t. $T_{ij}(t)$ depends on time.

Definition 4: $d_{ij}(t)$ is the direct evaluation of behaviors. $d_{ij}(t)$ is updated through verifications like challenge sent periodically: it represents the fraction of positive results when peer i verifies peer j. As we said before , the verification is in term of time taken to do a job, crashes of a node, and correctness of returned results. For example after verifications between two peer 1 and 2 with 4 positive results and 5 negative results, the evaluation score is 4/4+5=0.44. So, $d_{ij}(t) = 0.44$.

Definition 5: δt is the update period

The reputation value is evaluated following the Eq.1.

$$\mathrm{T}_{ij}(t) = \begin{cases} p.T_{ij}(t-\delta t) + (1-p).d_{ij}(t), & \text{if i verifies j;} \\ T_{ij}(t-\delta t), & \text{else.} \end{cases} \quad (1)$$

In equation (1), we want to assign more weight to recent interactions and less weight to the previous ones. So, p is used for that purpose.

B. Proposed Trust Models

Let first introduce some definitions and notations needed for our models. Then, we describe the proposed algorithms to compute the trust.

$||.||_1$ and $||.||_2$ respectively stand for the l_1-norm and l_2-norm of vectors.

T^t is the transpose of the matrix T.

Definition 6: Let π be the trust vector of size N. Initially, all peers are equally trusted, i.e $\pi_i = 0.5$ where $i = 1, 2, ..., N$. It is represented as a real number in the range of [0, 1] where 1 indicates complete trust, 0.5 ignorance, and 0 distrust.

Definition 7: ε is the convergence threshold.

Definition 8: Threshold is the trust limit used by the portal in the "decision maker" module.

1) PerronTrust Model : The PerronTrust algorithm is a power method. Notice that the power method is used in different areas including, for example, the computation of the PageRank of web documents [29]. PageRank represents a way of ranking the best search results based on a page's reputation. It ranks a page according to how many other pages are pointing to it. To derive a reputation score, they combine the collection of hyperlinks to a page seen as public information. Google's search engine is based on this PageRank.

Using the reputation matrix T, and the current trust vector, we get a new approximation of the trust of each peer j through a combination of the reputations of j:

$$\pi_j \leftarrow \frac{\sum_{i=1}^{N}(T_{ij}.\pi_i)}{||\pi||_1}. \quad (2)$$

If a peer i has a high trust score, then it is natural to give more importance to the reputation values that it is assigning to other peers. Writing Eq.2 for each peer leads to Eq.3:

$$\pi \leftarrow \frac{T^t.\pi}{||\pi||_1}. \quad (3)$$

The trust will then be iteratively computed by repeating Eq.3 until the trust vector becomes stable. The convergence of Algorithm 1 is based on the Perron-Frobenius theorem that we recall here for sake of completeness. The Perron Frobenius theorem asserts that a real square matrix with strictly positive entries has a unique largest real eigenvalue and that the corresponding eigenvector has strictly positive components.

THEOREM 1: Algorithm 1 will converge to the trust vector result π after a certain number of iterations.

Proof: Since we can assume that T is the real square matrix with positive entries, then we know that by a repetitive application of the previous iterative formula, the vector π will converge to the unique eigenvector associated with the largest eigenvalue $\lambda_{max}(T^t) = \lambda_{max}(T)$. Notice that we will get $||\pi||_1 = \lambda_{max}(T)$. The convergence is guaranteed if the starting point (the first approximated score vector) is not orthogonal to the eigenvector. This is clearly satisfied by the positive vector $\pi = (0.5, 0.5, 0.5...)$ since the eigenvector is also positive by the Perron-Frobenius theorem. ■

The iterative computation in Algorithm 1 continues until the total difference between $\pi(t)$ and $\pi(t-1)$ becomes smaller than ε. We have proved this convergence before. The convergence threshold is often predefined by the portal of the application. The threshold can be adapted depending on the precision wanted by the portal.

Algorithm 1 PerronTrust algorithm

Require: ε, N
Ensure: π
1: Retrieve reputation values (in T)
2: Initialize the trust vector π
3: **while** $||\pi(t) - \pi(t-1)||_1 > \varepsilon$ **do**
4: Calculate trust vector : $\pi \leftarrow \frac{T^t.\pi}{||\pi||_1}$
5: **end while**

2) CredTrust Model : In the previous model, some peers might have a good trust score while they are not really able to give a good estimation of the reputation of the other peers. Then, it is important to assign less importance to their evaluation. For that, we want to introduce a parameter that can measure this ability. This parameter is called credibility.

If a peer gives wrong evaluation about other peers, its credibility value is decreased and its evaluation values have a reduced impact on the trust of other peer. Similarly, if a peer's evaluation is good and in agreement with other evaluation peers, its credibility should be high. The credibility of a peer is used to weight the feedback it reports.

Let us first add some definitions:

Definition 9: Let **Cred** be the vector containing the ability to evaluate correctly the trust of peers (credibility). $Cred_i$ corresponds to the credibility of peer i. Cred values are normalized so that they lie between 0 and 1.

Given the reputation value of peer j seen by peer i and the trust estimated for peer j in the portal, we can evaluate the global credibility of the peer i using the following formula (4):

$$Cred_i = 1 - \left(\frac{\sum_{j=1}^{N} |\pi_j - T_{ij}|^2}{\sum_{j=1}^{N} |\pi_j - [1-\pi_j]|^2} \right)^{\alpha} \quad (4)$$

where $[1-\pi_j]$ denotes the nearest integer to $1-\pi_j$ and α is fixed number. Observe that $[1-\pi_j]$ is equal to 0 if $\pi_j > 0.5$ and to 1 if $\pi_j < 0.5$.

The credibility of i is equal to 0 if the evaluation given by i is always the farthest possible evaluation that can be accepted by the system. Conversely, if T_{ij} is equal to π_{ij} for each j, then $Cred_i$ is equal to 1.

Given these credibilities scores of each peer i in the system, it is now natural to estimate the global trust vector using the formula (5). To have trust between 0 and 1, we divide by $||Cred||_1$.

$$\pi \leftarrow \frac{T^t.Cred}{||Cred||_1} \tag{5}$$

The algorithm CredTrust performed by the portal is summarized below.

While we have not a formal convergence proof of Algorithm 2, all our simulations show that convergence is obtained. We might add that (5) can be written in the form $\pi \leftarrow f(\pi)$ where f is a function obviously defined by combining (5) and (4). The function f is clearly continuous. Each vector π belonging to the convex hull of the rows of the matrix T (columns of T^t) is mapped to a vector $f(\pi)$ belonging to the same convex hull. Using Brouwer fixed-point theorem, we can deduce that f has at least one fixed point: i.e., a vector π such that $f(\pi) = \pi$. A deeper study of the function f is required to deduce that the iterative process $\pi \leftarrow f(\pi)$ converges to a fixed point.

Algorithm 2 CredTrust algorithm

Require: ε, N
Ensure: π
1: Collect reputation evaluation T
2: Initialize the trust vector π
3: Initialize the credibility vector Cred
4: **while** $||Cred(t) - Cred(t-1)||_1 > \varepsilon$ **do**
5: **for** $i \in \{1, ..., N\}$ **do**

$$Cred_i = 1 - \left(\frac{\sum_{j=1}^{N} |\pi_j - T_{ij}|^2}{\sum_{j=1}^{N} |\pi_j - [1 - \pi_j]|^2} \right)^{\alpha}$$

6: **end for**
7: $\pi \leftarrow \frac{T^t.Cred}{||Cred||_1}$
8: **end while**

3) CredTrust-Trust ($CredTrust^2$) Model : As it will be shown in the simulation section, while the CredTrust approach is generally more efficient than the PerronTrust approach, there are some situations where it gives less precise results. This happens if there are some malicious peers who decide to correctly evaluate most of the other peers except one chosen malicious peer who is intentionally given a good evaluation. Since the malicious peers are giving the right evaluation in almost all cases, they will have a high credibility. This means that the evaluation given by these malicious peers will have more impact on the final result. Since these malicious peers decided to over-estimate a chosen malicious peer, this malicious peer will have a final trust higher than the one obtained by PerronTrust approach.

To overcome this problem, we are going to re-introduce again the trust in the iterative process. Instead of using only the credibility (as in CredTrust) or the trust (as in PerronTrust), we combine both of them.

Algorithm 3 CredTrust-Trust algorithm

Require: ε, N
Ensure: π
1: Collect reputation evaluation (in T)
2: Initialize the trust vector π
3: Initialize the credibility vector Cred
4: **while** $||Cred(t) - Cred(t-1)||_1 > \varepsilon$ **do**
5: **for** $i \in 1..N$ **do**

$$Cred_i = 1 - \left(\frac{\sum_{j=1}^{N} |\pi_j - T_{ij}|^2}{\sum_{j=1}^{N} |\pi_j - [1 - \pi_j]|^2} \right)^{\alpha}$$

6: **end for**
7: $\pi \leftarrow \frac{T^t.(Cred \otimes \pi)}{||Cred \otimes \pi||_1}$
8: **end while**

More precisely, we modify the Eq. 5 to have the new following formula :

$$\pi \leftarrow \frac{T^t.(Cred \otimes \pi)}{||Cred \otimes \pi||_1} \tag{6}$$

where $Cred \otimes \pi$ denotes the componentwise product of $Cred$ and π. It is now clear that even if a peers has a high credibility, the impact of its opinion is attenuated if he has a low trust.

The same convergence remarks related to the previous algorithm apply also for Algorithm 3.

4) EigenTrust Model : Algorithm 4 presents a reminder of EigenTrust.

Algorithm 4 Basic EigenTrust algorithm

Require: ε, N
Ensure: π
1: Retrieve reputation values (in T)
2: Normalize the values in T to have stochastic matrix $T2$
3: Initialize the trust vector π
4: **while** $||\pi(t) - \pi(t-1)||_1 > \varepsilon$ **do**
5: Calculate trust vector : $\pi \leftarrow T2^t.\pi$
6: **end while**

5) AverageTrust Model : This method is used for eBay and some models in cloud computing [24]. Algorithm 5 presents this simple method.

Algorithm 5 AverageTrust algorithm

Require: ε, N
Ensure: π
1: Retrieve reputation values (in T)
2: Initialize the trust vector π
3: Calculate trust vector : For each i $\pi_i \leftarrow Average(T_i)$

V. Simulation Analysis

In this section, we analyze the performance results of the three approaches described in the previous section. After presenting the main simulation parameters, we show by simulation the benefit of trust in jobs affectation. We describe a set

of attacks and behaviors that are considered here. Then, we compare the three models PerronTrust, CredTrust, CredTrust-trust, AverageTrust and EigenTrust under illustrated attacks.

A. Simulation Parameters

We first consider an hybrid execution with 20 peers in an IaaS cloud and 80 peers running on local resources. So, the total number of peers in the system is $N = 100$. Peers can be either normal or cheating. The behavior of each peer is chosen randomly depending on: 1/ whether the peers is a local one or it belongs to the IaaS, 2/ the attack scenario (peers behavior) described below.

B. Attack Models: Cheating Peers' Behaviors

In this section, we present some the possible behaviors of the peers that are considered in our experiments.

1) Byzantine Peers' Behaviors:

- **Inverting peers:** This type of peers always inverses the scores of reputation obtained after interaction with others (gives a good evaluation to honest peers and a bad evaluation to malicious peers).
- **Coalition peers**: Peers form a malicious group by assigning a high reputation value to most of the other peers except one malicious peer who is intentionally given a good evaluation.

2) Rational Peers' Behaviors:

- **Less efficient peers:** Partially altruistic peers are rational, they want to optimize resources of interaction with the evaluated peer. They are less efficient and affect the computation of the reputation.

C. Results and Comparison Analysis

We focus in this section on the simulation of the different attack scenarios described before and try to analyze the results. We compare the results of our scheme with the EigenTrust [18] scheme and AverageTrust under these scenarios. We compare the final trust assign to cheating peers and the difference between normal peer and cheating peer of our three schemes with AverageTrust scheme and EigenTrust. The second metric, the difference in term of trust between normal peer and cheating peer, is the ratio of trust assign to cheating peers over trust assign to normal peers, is used to evaluate the accuracy of the decision based on trust.

1) Byzantine Peers' Behaviors: Inverting Peers: We suppose here that all peers cooperate in the evaluation of reputation values. the reputation values given by peers in the system are as follow:

- inverting peers estimate other inverting peers with 0.8 and altruistic peers with 0.2,
- altruistic peers estimate inverting peers with 0.2 and others with 0.8.

The trust values estimated for inverting peers are shown in Fig. 3.

When the proportion of malicious peers is about 10%, the three algorithms give almost the same estimation for the trust of malicious peers. When the fraction of malicious peers increases, the trust values increase for the three models.

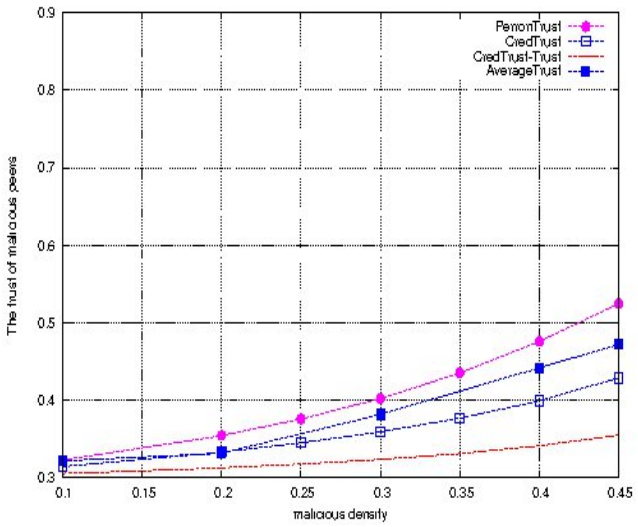

Figure 3. Trust under byzantine peers' behaviors: inverting peers

Not taking into consideration the credibility factor, PerronTrust and AverageTrust can not punish the peers that give wrong reputation values. While CredTrust is doing better than PerronTrust since it is considering the credibility of peers, CredTrust-trust outperforms PerronTrust, CredTrust and AverageTrust. This is again due to the fact that CredTrust-trust penalizes more the inverting byzantine peers.

Suppose that we want to compare our three algorithms to EigenTrust and AverageTrust in term of $RTI = \frac{Trust\ of\ invertingpeers}{Trust\ of\ altruistic\ peer}$

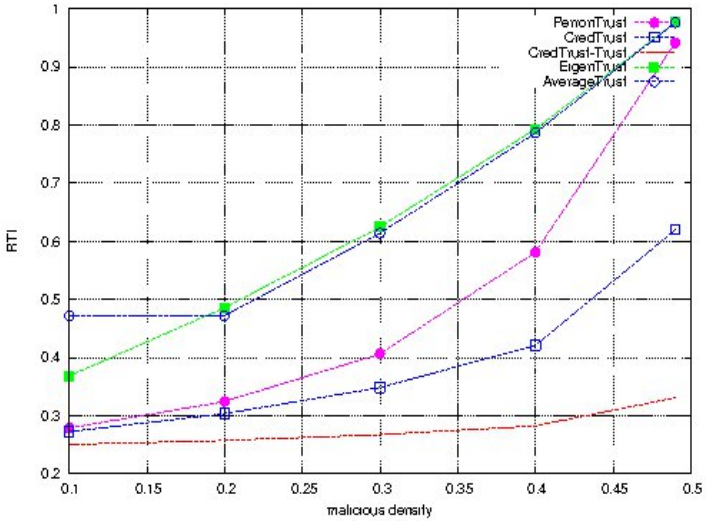

Figure 4. RTI under inverting byzantine peers' behaviors

Results are shown in Fig. 4 where we consider the three algorithms CredTrust-trust, PerronTrust and CredTrust compared to EigenTrust and AverageTrust, with different densities

of malicious peers.

Observe that CredTrust-trust outperforms PerronTrust, CredTrust, EigenTrust and averageTrust. This is again due to the fact that CredTrust-trust penalizes more the inverting byzantine peers. It is clearly that our three methods compared to EigenTrust and AverageTrust help in taking secure decision and then assure trusted execution of tasks in the public cloud.

2) Byzantine Peers' Behaviors: Different Level of Maliciousness: We consider that we have two types of byzantine peers:

- Group 1: Peers inverse the reputation of others. They estimate peers from group 1 and 2 with 0.8 and altruistic peers with 0.2,
- Group 2: Peers different from altruistic peers and peers from group 1 (They don't inverse the scores). They estimate peers from group 1 and 2 with 0.6, altruistic peers with 0.2.

We will assume that 20% of the peers belong to group 1, 20% belong to group 2 and 60% of the peers are altruistic.

We consider also ratios:

- $R1 = \frac{Trust\ for\ group\ 1\ peers}{Trust\ for\ altruistic\ peers}$
- $R2 = \frac{Trust\ for\ group\ 2\ peers}{Trust\ for\ altruistic\ peers}$

Obtained results are shown in the table I:

In this scenario, we affirm that:

- the credibility of peers from group 1 is clearly less important in CredTrust-trust than in CredTrust,
- the credibility of peers from group 2 is clearly less important in CredTrust-trust than in CredTrust,
- the credibility of peers in group 1 is less important than the credibility of peers in group 2,
- the trust assigned to peers from group 1 is clearly less important in CredTrust-trust than in PerronTrust and CredTrust and AverageTrust,
- the trust assigned to peers from group 2 is clearly less important in CredTrust-trust than in PerronTrust , CredTrust and AverageTrust,
- altruistic peers are more recognized in the CredTrust-trust model than in PerronTrust, CredTrust and AverageTrust,
- R1 is less important in CredTrust-Trust than in AverageTrust, CredTrust and PerronTrust,
- R2 is less important in CredTrust-Trust than in EigenTrust, AverageTrust, CredTrust and PerronTrust,
- EigenTrust can't detect the byzantine peer in this case since ratios R1 and R2 are equal to 1.

In this case also, CredTrust-trust ($CredTrust^2$) outperforms PerronTrust, CredTrust, AverageTrust and EigenTrust.

3) Byzantine Peers' Behaviors: Coalition: We consider that malicious peers select one attacker of the coalition. They assign to this attacker good reputation ($= 1$). Malicious collusive peers provide true reputation to hide their essences. The attacker evaluates peers in a consistent way (like an altruistic peer) in order to increase his credibility.

We will focus on the trust value of the attacker chosen by the coalition. We still use the three algorithms, EigenTrust and AverageTrust to compute the trust vectors.

Results when we vary the fraction of malicious peers are given in Fig. 5. We can see that it is still possible to detect the cheater with all algorithms especially for CredTrust-trust and PerronTrust. For CredTrust and averageTrust, when the fraction of malicious peers reaches 44%, it becomes difficult to detect the attacker chosen by the coalition since its trust is above the threshold of decision is 0.5.

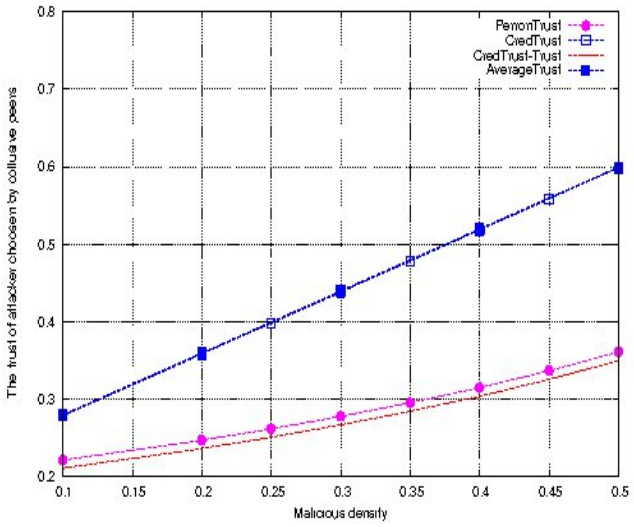

Figure 5. Trust under coalition

Observe that CredTrust and AverageTrust give almost the same estimation of trust for the chosen attacker.

Suppose that we want to compare our three algorithms to EigenTrust and AverageTrust in term of $RTC = \frac{Trust\ of\ the\ chosen\ attacker\ peer}{Trust\ of\ altruistic\ peer}$.

Results are shown in Fig. 6 where we consider the three algorithms CredTrust-trust, PerronTrust and CredTrust compared to EigenTrust and AverageTrust, with different densities of malicious peers. Observe that CredTrust and AverageTrust show high value of RTC because the attacker evaluates peers in a consistent way (like an altruistic peer) in order to increase his credibility. As CredTrust is only based on credibility the gap between altruistic and the attacker is small.

However, EigenTrust, PerronTrust and CredTrust-Trust are performing well but CredTrust-Trust considering credibility and trust gives better value of RTC.

4) Rational Peers' Behaviors: Less Efficient Peers: In this attack, we have byzantine inverting peers and rational less efficient peers. IT means that we simulate the inverting attack and we suppose that in the system we have efficient and 20 % less efficient rational peers. Less efficient peers are cooperative but they are not able to evaluate correctly malicious behaviors: they give 0.4 as an estimation for malicious behaviors (0.8 for altruistic and other less efficient peers). In addition, we have efficient altruistic peers that evaluate correctly other peers: estimating inverting peers with 0.2 and others with 0.8.

First, let's focus on the credibility seen in CredTrust and CredTrust-Trust. In Fig. 7, we show that the credibility of

Table I
RESULTS UNDER DIFFERENT LEVELS OF MALICIOUSNESS

	PerronTrust	CredTrust	$CredTrust^2$	Eigen	Average
Trust for group 1 peers	0.3204	0.3239	0.2364		0.4020
Trust for group 2 peers	0.3216	0.3265	0.2371		0.4040
Trust for altruistic peers	0.6833	0.6794	0.7670		0.6020
Credibility for group 1 peers		0.2922	0.2587		
Credibility for group 2 peers		0.5844	0.5173		
Credibility for altruistic peers		0.8164	0.9523		
R1	0.468	0.476	0.308	1.002	0.667
R2	0.456	0.480	0.309	1	0.671

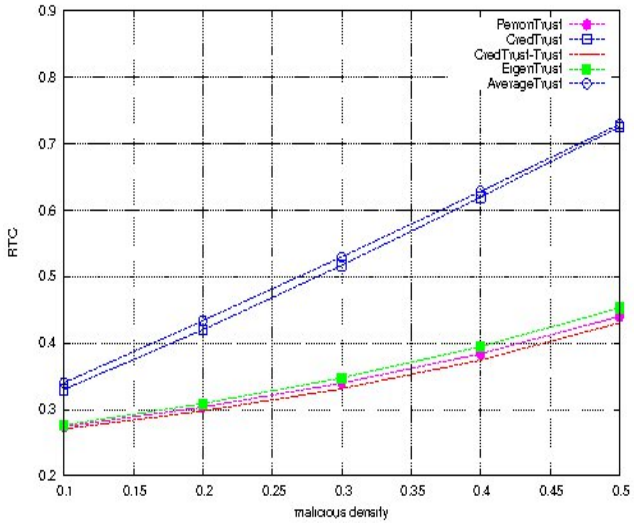

Figure 6. RTO under coalition

the altruistic efficient peers are more important in CredTrust-trust than in the CredTrust. We favorite these peers in the computation of the trust scores in the system.

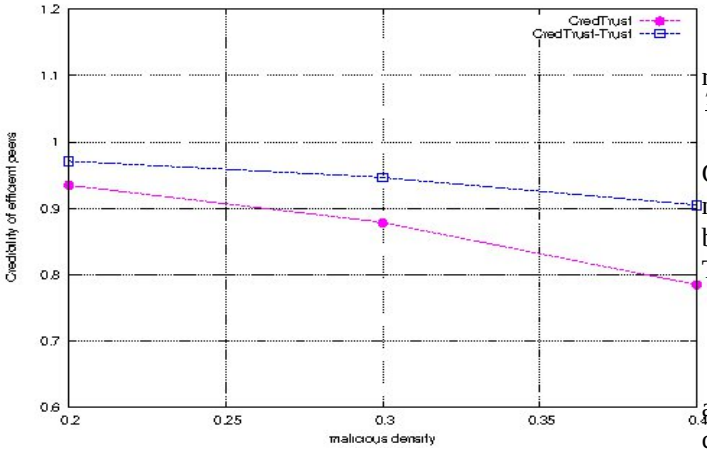

Figure 7. Comparison of credibilities

Second, results of the trust of malicious peers under this efficiency problem are shown in Fig. 8. We can notice that CredTrust-trust outperforms PerronTrust, CredTrust and AverageTrust. This model gives the most significant trust to potential malicious peers under 40% of malicious ones. This is due to the credibility effect. In fact, the credibility of less efficient peers is considered in the computation of the trust. Hence, the model considering credibility can efficiently distinguish efficient peers from less efficient ones.

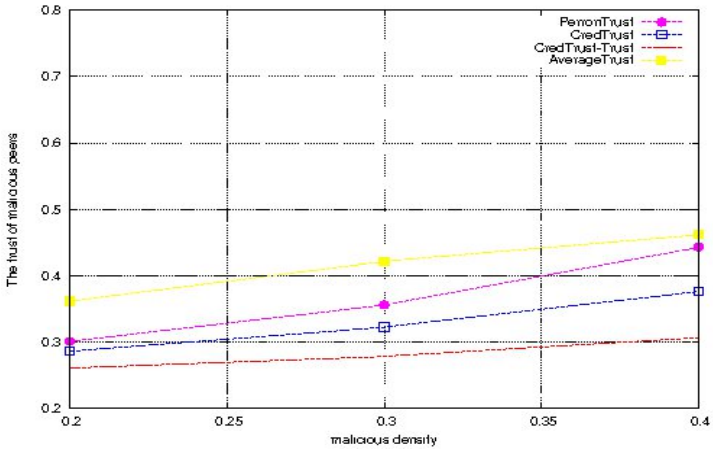

Figure 8. Comparison under efficiency problems

Third, suppose that we want to compare our three algorithms to EigenTrust and AverageTrust in term of $RTL = \frac{Trust\ of\ byzantine\ peers}{Trust\ of\ altruistic\ peer}$. Results are shown in Fig. 9.

We can notice that CredTrust-trust outperforms PerronTrust, CredTrust, EigenTrust and AverageTrust. In this case, we can notice again the benefit of credibility to make clear decisions between behaviors. That is why, CredTrust and CredTrust-Trust outperform PerronTrust, EigenTrust and AverageTrust.

VI. CONCLUSION

In this paper, we designed new algorithms to detect rational and byzantine (malicious) peers in the context of hybrid cloud. We considered a dynamic reputation as a peer to peer evaluation. Three models are proposed and compared under different cheating strategies. First, PerronTrust, a model considering the computation of trust based on Perron algorithm. Second, CredTrust improves the first approach by introducing the concept of credibility when the trust vector is updated. A third model, CredTrust-trust is proposed by combining the

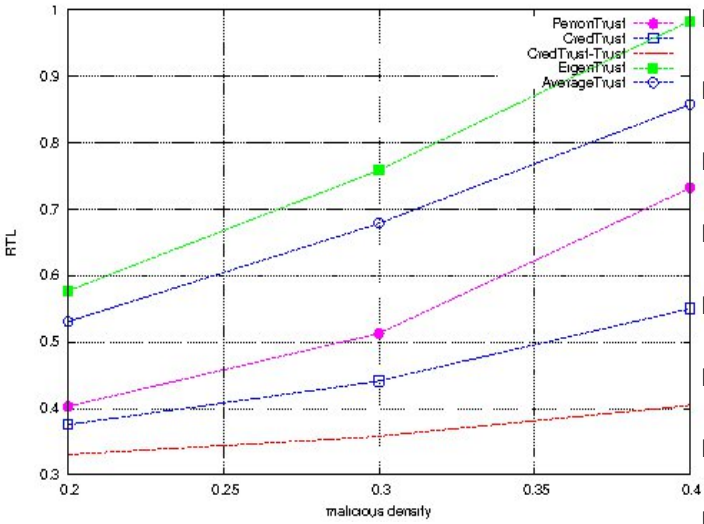

Figure 9. RTL under under efficiency problems

trust and the credibility parameters in the iterative updating process.

Simulations and comparison with two well-known existing works are performed. The results of the first part of experiments show that the proposed model selects the dependable and reliable peers in cloud environment. Second part of simulations confirms also that the credibility is well suited to clarify the behaviors of malicious (byzantine) and rational peers.

We also noticed that many collective attacks and compromised peers can be clearly detected with high accuracy and low false alarm probability. This is particularly true when CredTrust-trust is used. It outperforms the well-known EigenTrust scheme and AverageTrust significantly. The combination of trust and credibility improves the performances of the method in a very significant way. Moreover, the complexity of the algorithm is very limited allowing it to be used in large scale systems.

Our framework results can help in making decision on whether to purchase execution in resources from an unknown supplier or not.

One possible future research direction is to propose a gossip-based algorithm for reputation aggregation in a completely distributed system.

REFERENCES

[1] P.Mell, and T.Grance, "The NIST Definition of Cloud Computing", 2009

[2] Trend micro, "Cloud security survey", Global executive summary, June 2011

[3] V.D.Cunsolo, S.Distefano , A.Puliafito , M.Scarpa , "Volunteer Computing and Desktop Cloud: The Cloud@Home Paradigm", Eighth IEEE International Symposium on Network Computing and Applications, 2009.

[4] Abhishek Chandra and Jon Weissman, "Nebulas: Using Distributed Voluntary Resources to Build Clouds", HotCloud'09 Proceedings of conference on Hot topics in cloud computing, 2009.

[5] F.Azzedin, and M.Maheswaran, "Towards Trust-Aware Resource Management in Grid Computing Systems", 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID02), 2002.

[6] K.-J. Lin et al., "Reputation and Trust Management Broker Framework for Web Applications", Proc. IEEE Conf. e-Technology, e-Commerce, and e-Services, IEEE CS Press, pp. 262-269, 2005.

[7] Francesco Ricci and Lior Rokach and Bracha Shapira, "Introduction to Recommender Systems Handbook", Recommender Systems Handbook, Springer, pp. 1-35, 2011.

[8] D. Zeng. "Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering". ACM Trans. Inf. Syst., 22(1):116Ű142, 2004.

[9] P.Zhang and al. "Survey of trust management on various networks", International Conference on Complex Intelligent and Software Intensive Systems, 2011.

[10] Marsh, S., "Formalising Trust as a Computational Concept. PhD thesis", University of Stirling, Department of Computer Science and Mathematics, 1994.

[11] A. Mejia et al., "A game theoretic trust model for on-line distributed evolution of cooperation in MANETs" Elsevier Journal of Network and Computer Applications.34:39-51, 01/2011.

[12] S. Ganeriwal, L.K. Balzano, and M.B. Srivastava, "Reputation-based framework for high integrity sensor networks," ACM Transitions on Sensor Network, vol. 4, no. 3, May 2008.

[13] K. Liu, N. Abu-Ghazaleh, and K.-D. Kang, "Location verification and trust management for resilient geographic routing," J. Parallel and Distributed Computing, vol. 67, no. 2, pp. 215-28, 2007.

[14] R.A. Shaikh, H. Jameel, B.J. d'Auriol, H. Lee, S. Lee, and Y.J. Song, "Group-based trust management scheme for clustered wireless sensor networks," IEEE Transactions on Parallel and Distributed Systems, vol. 20, no. 11, pp. 1698-1712, Nov.2009.

[15] T.Kim ,and H.Seo, "A trust model using fuzzy logic in wireless sensor network", World academy of science, engineering and technology, 42, 2008.

[16] Y. Zhang, H. Chen, and Z.Wu, "A social network-based trust model for the semantic web. In Autonomic and trusted computing", Springer, pp. 183Ű192, Berlin 2006:

[17] L. Xiong , and L.Liu," PeerTrust: Supporting Reputation-Based Trust for Peer-to- Peer Electronic Communities", IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 7, July 2004.

[18] S. Kamvar, M. Schlosser and H. Garcia-Molina , "The EigenRep Algorithm for Reputation Management in P2P Networks". In Proceedings of WWW, Budapest, Hungary, 2003.

[19] R. Chen, X. Zhao, L. Tang, J. Hu, and Z. Chen, "CuboidTrust: a global reputation-based trust model in peer-to-peer networks," Fourth Int. Conf. on autonomic and Trusted Computing, pp. 203-215, 2007

[20] G.Silaghi and al., "Reputation-based Trust management systems and their applicability to Grids (Technical report), Network of Excellence 2007.

[21] H. Liu and J. Shen, "A Mission-Aware Behavior Trust Model for Grid Computing Systems", Int'l workshop on Grid and Cooperative Computing (GCC2002), Sanya, China, Dec. 26, 2002.

[22] S. Song and K. Hwang, "Fuzzy trust integration for security enforcement in grid computing," in International Symposium on Network and Parallel Computing (NPC2004), March. 2004.

[23] Wenjuan Li, Lingdi Ping, and Xuezeng Pan, "Use trust management module to achieve effective security mechanisms in cloud environment," in International Conference on Electronics and Information Engineering (ICEIE), vol. 1, Kyoto, Japan, pp. 14-19, 2010

[24] P.D. Manuel, T. Selve, and M.I. Abd-EI Barr, "Trust management system for grid and cloud Resources," in First International Conference on Advanced Computing (ICAC 2009), Chennai. India, pp. 176-181, 2009

[25] W.Li, and L.Ping, "Trust Model to Enhance Security and Interoperability of Cloud Environment", CloudCom 2009

[26] C. Lin, V. Varadhrajan, Y. Wang, and V. Pruthi, "Enhancing Grid Security with Trust Management", IEEE International Conference On Services Computing, SCC 2004

[27] A.Bradai and H.Afifi, "Enforcing Trust-based Intrusion Detection in Cloud Computing Using Algebraic Methods", IEEE International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2012.

[28] L. Lamport, R. Shostak , and M. Pease, "The Byzantine Generals Problem", ACM Transactions on Programming Languages and Systems, pp. 382-401, July.1982

[29] Mohit Agrawal, "The Perron-Frobenius Theorem and Google's PageRank Algorithm", 2010