Enforcing Trust-based Intrusion Detection in Cloud Computing Using Algebraic Methods

Amira Bradai, Afifi Hossam

Enforcing Trust-based Intrusion Detection in Cloud Computing Using Algebraic Methods

Amira Bradai* and Hossam Afifi*
* Institut Mines-Telecom,Telecom SudParis(TSP)

***Abstract*—A trust-based intrusion detection scheme for hybrid cloud computing is proposed. We consider a trust metric based on honesty, cooperation and efficiency for detecting malicious machines. We use Perron-Frobenius theorem to detect intrusion based on trust and observations.**
By statistically analyzing peer-to-peer trust distributed results, we apply trust-based intrusion detection to assess the trustworthiness and maliciousness.
An analytical model and simulation for performance are developed. We analyze the sensitivity of false alarms with respect to the minimum trust threshold below where a node is considered malicious.
Results confirm that our proposal is flexible enough to detect malicious behaviours in various context of executing application in hybrid cloud. With this work, we can guide future execution in the cloud resource.

***Index Terms*—trust management, intrusion detection, Perron Frobenius, cloud computing, hybrid execution, false alarms, security scores.**

I. Introduction

The concept of the cloud computing has rapidly emerged as an efficient method for service delivery. There are three major cloud service models: infrastructure-as-service (IaaS), software-as-a-service (SaaS), and platform-as-a-service (PaaS). IaaS is the model that we consider in this paper.
Cloud computing is usually deployed for 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. Public clouds run by third parties, and applications from different customers are likely to be mixed together on the cloud's servers, storage systems, and networks. Hybrid clouds that combine both public and private cloud models can help to provide on-demand, externally provisioned scale. In a Cloud Security Survey [5], 32% of enterprise 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. Applications like PSAs(A Parameter Survey Application) need hybrid methods. IaaS systems offer infrastructures for applications shared by multiple tenants. These tenants are with different security domains. Therefore, our work focuses on the possibility of using the IaaS model shared with private cloud and still guaranteeing a certain degree of security, trust and prevention .

The issue of security in clouds has been addressed in many publications [2], [3] and [4]. Clearly, this new model is exposed to many attacks. We can classify threats in cloud computing based on relative location of attacks. Attacks can be located at the user side, at the network or at the service provider side. An attacker can listen to network traffic, insert malicious traffic, probe cloud structure and launch DoS (Denial of Service), DDos (Distributed Denial of Service), ARP spoofing and IP spoofing. With clouds, we are also more exposed to the risk of identity theft and the loss of personal data. Attack methods such as phishing, fraud, brute force dictionaries and exploitation of software vulnerabilities can be used with more dramatic results. On the other hand, the implementation characteristics of IaaS, such as multi-tenancy, sharing of virtual servers, virtualization, elasticity and programmable APIs bring new attack vectors

In this paper, we aim to detect possible attacks in different locations : at the user side and IaaS cloud. Enterprises should not trust their applications to IaaS providers and rely on the security processes put in place by providers. Commercial offers don't give visibility to IaaS users of advanced security primitives such as Intrusion Detection and Prevention (IDS/IPS). For this, we need to adopt a pair trust evaluation model based on three parameters: cooperativeness, honesty and efficiency. we don't use the trust directly to identify malicious machines but to feed a novel algorithm that detects intrusions rapidly and with more efficiency than other means. Our proposal is evaluated via simulation with exhaustive scenarios. The evaluation considers several attack scenarios such as grouped attacks, non-cooperation in trust management and falsification of trust results. Our algorithm shows that we can reach minimum false alarms for different proposed methods that ca be used in different cases.

The rest of the paper is organized as follows. Section II details some contributions on trust and intrusion detection. In Section III, the system model and the evaluation of pair trust score are described . Section IV presents the proposed intrusion detection system and Section V shows some simulation results and analysis. Finally, the paper concludes and lists some possible extensions for future work.

II. TRUST MODELS AND INTRUSION DETECTION

In this section, we briefly enumerate some contributions on trust models and protocols. We also introduce some principles in intrusion detection and trust management.

978-0-7695-4810-4/12 $26.00 © 2012 IEEE
DOI 10.1109/CyberC.2012.38

A. Trust Protocols

Trust can be seen as the general confidence in a person or an object. Generally, it is evaluated by values on a scale from zero to one. Several trust management protocols [6], [7] and [8] have been proposed for network security, data integrity, and secure routing in different fields. In [8] a group-based trust management scheme for clustered Wireless Sensor Networks was proposed. This protocol reduced the use of memory for storing trust scores and minimizes the cost associated with trust evaluation of distant nodes compared to other works. Marsh was one of the first authors to introduce a computational model for trust [9]. He examined the concepts of trust, mistrust, and distrust. He also covered many points related to the concept of trust but this work needs more refinement. Fuzzy logic was introduced to trust models in [10] focusing on the trustworthy 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.

B. Intrusion Detection

Intrusion detection research lacks firm scientific bases as it is mainly relying on progressive knowledge of protocol flaws and system weaknesses. We still try to enumerate some contributions that have helped us in designing the proposed algorithm.
Using replicated services, a preliminary work has been proposed via Byzantine Fault Tolerance techniques (BFT) [11] to detect malicious behaviors. This work imposed an agreement protocol and limited by high overhead. Intrusion detection systems for grid can be based on host, network, data, knowledge and behavior. Host-based Intrusion Detection System(HIDS) [12] consists of an agent that identifies intrusions by analyzing activities on the host and its current state. An adapted version of this system will be used in our detection algorithm. In [13], the intrusion detection in the cloud computing is considered. In [14], a detection system is proposed to reduce the impact of DoS attacks.

C. Trust Management for Intrusion Detection

In the literature, few contributions combine trust and intrusion detection together. Wang et al. [15] proposed an intrusion detection mechanism based on trust (IDMTM). Their work considered both evidence chain (main malicious behavior forms) and trust fluctuation. They provided more accurate decision with low false alarms. Ebinger et al. [16] introduced a cooperative intrusion detection method for MANETs based on trust evaluation and reputation exchange. The reputation and confidence are combined with trustworthiness to improve intrusion detection. The contribution didn't however consider dynamic MANET environments. In [17], F.Bao et al.(2011) proposed a scheme for hierarchical trust management considering honesty, cooperativeness and energy. This model suits our requirements. Hence, it will be adapted to the first part of our work (building pair trust scores).

III. SYSTEM DESCRIPTION AND TRUST

In this section, we first describe our network architecture for cloud and application processing. Then, we briefly present the trust model that we adapt to calculate the pair trust scores.

A. Network Architecture and Assumptions

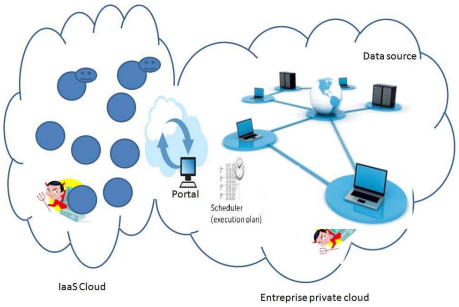

Figure 1. System overview

Applications like PSAs(A Parameter Survey Applications) need hybrid execution mechanism utilizing both local computing resources with a batch scheduler and an IaaS Cloud. We start from the assumption that cloud is used to execute some tasks of this application remotely. The scheduler organizes the application's execution. The model (in fig. 1) distinguishes between two types of equipment: devices that are local and other in the IaaS.
The local machines are supposed to be in a trusted and protected location (but they can launch internal attacks).
IaaS ones are supposed to be in different domains so different security assets, that are not necessarily well protected. Let us assume there are ***N+M*** machines in the system:

- ***N*** constitutes a finite set of local machines having a task to execute for the application.
- ***M*** represents machines allocated from the IaaS cloud computing having also tasks to accomplish for the same application.

In our system, we use both private computing resources with a portal and an IaaS Cloud as shown in Fig. 1. The user submits the application and fixes the deadlines and execution time with the portal (a trusted node in the system). The executer in the system can have two behaviors: The honest executer never cheats whereas the malicious one may cheat during the execution. The probability to have local and malicious is less than to have cloud and malicious. Each machine can decide to cooperate in the evaluation of trust or not to cooperate (this is part of the attack strategy).

B. The Trust Model and the Peer-to-peer Trust Score

After the interaction between machines, each machine that chooses a check strategy will give a trust value to the checked machines . Our model considers the effect of cooperativeness, honesty and efficiency(it can be task completion capability). The honesty trust component is measured through evidences of

dishonesty such as false selfreporting, trust fluctuation and abnormal trust recommendations. A compromised node usually manifests itself as being uncooperative. Malicious machines give wrong values without respecting the trust model. We adapted the trust model cited previously in [17]: We added new parameter: efficiency and we change the definition of honesty. Based on the previous indicators, the trustworthiness of machines can be estimated based on local observations of their behaviours. So, we use the peer-to-peer trust score $T_{ij}(t)$as an evaluation of machine ***i*** to machine ***j*** at time ***t***. It is represented as a real number in the range of [0, 1] where 1 indicates complete trust, 0.5 ignorance, and 0 distrust. The trust value is the weighted average of trust values related to honesty T_h, cooperativeness T_c and efficiency T_e

$$\mathrm{T}_{ij} = w_h.T^h_{ij}(t) + w_c.T^c_{ij}(t) + w_e.T^e_{ij}(t) \quad (1)$$
$$w_h + w_c + w_e = 1$$

w_c, w_e and w_h are the weights associated with each component. We will explain each term of the equation (1) where we evaluate the trust value $T_{ij}(t)$. We will start with the update of the honesty term:

$$\mathrm{T}^h_{ij}(t) = \begin{cases} p_h.T^h_{ij}(t-\delta\mathrm{t})+(1-\mathrm{p}_h).T^{dir}_{ij}(t), & \text{if i checks j;} \\ T^h_{ij}(t-\delta\mathrm{t}), & \text{else.} \end{cases} \quad (2)$$

p_h: the weight associated with the previous evaluation of the honesty.
δ***t***: update period,
$T^{dir}_{ij}(t)$:this is the belief of machine i that machine j is honest based on machine iŠs direct observations toward machine j. This refers to the peer to peer verifications and monitoring. It is updated through verifications like challenge sent periodically: it represents the fraction of positive results when machine i verifies machine j. For example after verifications between machine 1 and machine 2 (machine 1 verifies machine 2) with 5 positive results and 4 negative results, the pair trust score is 5/4+5=0.55. So, $T^{dir}_{ij}(t) = 0.55$. The weight associated with this component is $1 - p_h$.
The following explains the update of the cooperativeness term T_c:

$$\mathrm{T}^c_{ij}(t) = \begin{cases} p_c.T^c_{ij}(t-\delta\mathrm{t})+(1-\mathrm{p}_c).T^{deg}_{ij}(t), & \text{if i checks j;} \\ T^c_{ij}(t-\delta\mathrm{t}), & \text{else.} \end{cases} \quad (3)$$

p_c: The weight associated with the previous evaluation of the cooperativeness.
$T^{deg}_{ij}(t)$: refers to the degree of cooperation seen by machine i when verifying machine j at time t. Malicious machine can manifest itself as being uncooperative to avoid detection. The weight associated with this component is $1 - p_c$.

$$\mathrm{T}^e_{ij}(t) = \begin{cases} p_e.T^t_{ij}(t-\delta\mathrm{t})+(1-\mathrm{p}_e).T^{eff}_{ij}(t), & \text{if i checks j;} \\ T^t_{ij}(t-\delta\mathrm{t}), & \text{else.} \end{cases} \quad (4)$$

p_e: The weight associated to the previous evaluation of the efficiency.
$T^{eff}_{ij}(t)$: refers to the efficiency evaluated by i (after executing verification and the application task). The weight associated to this component is $1 - p_e$.

The result of the trust evaluation process is a matrix containing peer-to-peer trust values between nodes.

IV. PROPOSED DETECTION SYSTEM

In this section, we describe the proposed algorithms that calculate security scores for current and future execution in hybrid cloud. We base our intrusion detection methods on the Perron Frobenius theorem.
"***Perron Frobenius theorem*** *asserts that a real square matrix with positive entries has a unique largest real eigenvalue and that the corresponding eigenvector has strictly positive components.*"
Our algorithm using the PerronFrobenius theorem gives dynamic evaluation: with each successive iteration, we ask if the security score remains stable or will be different. Notice that this approach was also used to compute the PageRank of web documents [18].

A. Intrusion Detection Methods Description

1) The First Method: Let us first introduce some notations.

- T: In order to obtain the global view of the trust management, we construct a matrix T of the pair trust score. This matrix is initialized to be 0.5 (ignorance). Hence, $T_{ij} =$ depends on time and represents how much a node i trusts a nodes j.

- π: We define the security score vector sized ***N+M***. This objective security score is initialized as π ***= (0.5, 0.5, 0.5 ...)***

Given some approximate values of the security scores, the idea is to get a better estimation of π_j using these approximate values. This can be done through the formula: $\pi_j \leftarrow \frac{\sum_i (T_{ij}.\pi_i)}{||\pi||_1}$.
In other words, for each machine j, the new security score is obtained by combining the trust values T_{ij}. If a node i has a high security score, then it is natural to give more importance to the trust values that it assigns to other nodes.

The general form of the iterative formula is given by: $\pi \leftarrow \frac{T^t.\pi}{||\pi||_1}$

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.

2) The Second Method: This method is a refinement of the first method. In fact some nodes might have a good security score, while they are not really able to give a good estimation of the security of the other nodes. In this case, it is important to assign less importance to their judgment.

Let us add some parameters:

- **d**: is a vector representing the ability to evaluate correctly the security score of machines (d_i corresponds to node i).
- **B**: is the privilege matrix (where the portal gives different weights to machines in the system). To estimate whether a node is really a good trust evaluator, it may be important to focus on what it said about some special nodes. This is expressed through this privilege matrix by putting a value in the diagonal decided by the portal.

More precisely, starting from an estimation of the security vector, one can estimate the ability of nodes to evaluate the trust using the following formula:

$d_i \leftarrow diag(\frac{1}{||T_i B^t||_2})\frac{T.B^t.B.\pi}{||B.\pi||_2}$ where $diag(\frac{1}{||T_i B^t||_2})$ is a diagonal matrix where the i^{th} term is given by the norm of the i^{th} line of T multiplied by B^t. In fact, the ability of node i can be seen as the inner product of $B.\pi$ and $B.{T_i}^t$. The diagonal matrix and the division by $||B.\pi||_2$ allow to normalize this inner product.

The resulting vector d, gives an estimation of the ability values of each machines. Given these abilities, it is now natural to estimate the scores using the formula :

$\pi \leftarrow \frac{T^t.d}{||d||_1}$

An iterative algorithm where both formulas are repetitively and alternatively used to estimate the ability and the security score will converge. In fact, this holds if we assume that the matrix $T^t T B^t B$ is positive (which is true here). The Perron-Frobenius can again be used in the same way as before to ensure the convergence of the algorithm.

3) The Third Method: In this method, we add the vector h and the value v. **v**: is the security score for trusted machines This method does not use the Perron theorem. We assume that the portal has a knowledge of some trusted nodes from previous executions for example. It does not update their trust score from external calculations. The rest of nodes that are unknown to it have their trust calculated as before. Hence, the initial security score is presented as a vector h: h=(0.5,0.5,0.5,0.9,0.5,0.9,0.5) as an example. We adopt zeros in the diagonal of B to the untrusted machines and a value different to zero to the trusted machines.

We assume that the value 0.9 is given to the local machines.

$d \leftarrow diag(\frac{1}{||T_i B^t||_2})\frac{T.B^t.B.h}{||B.h||_2}$

$\pi \leftarrow \frac{T^t.d}{||d||_1}$

Here, π is obtained in one step.

4) The Fourth Method: In this method, the portal uses the recursive convergence method (Perron). It maintains however unchanged, the security scores for machines that it trusts, in each step. With this method, we can benefit from the trusted machines by fixing the initial security scores. Also, we affect at the end the modified security score according to the knowledge that the portal have.

π_c: is the security vector with trusted machines values.

$d \leftarrow diag(\frac{1}{||T_i B^t||_2})\frac{T.B^t.B.\pi_c}{||B.\pi_c||_2}$

$\pi \leftarrow \frac{T^t.d}{||d||_1}$

5) General Algorithm for intrusion detection: We describe in this section the steps needed to obtain the security score value of each machine in the hybrid cloud. The algorithm switches between the four methods described before based on intervals of π.

In the beginning, as we don't have any precise knowledge of this value, we start with the first method. When this parameter is relatively high, we consider that the nodes are quite trusted and we adopt hence the third or fourth methods. When however, we have doubts about the trustiness of the node (values between .4 and .6) and we can put privilege in some machines after some observations(network behavior), we use the method number two. We will see after in the simulation section why we are choosing this algorithm.

1) At t=t_0+δt execute 4,3, 2 then 1;

2) Each machine reports its peer-to-peer trust evaluation of other machines in the matrix T to the portal;

3) Execute Method 1;

4) Calculate π,

if $\pi_i > 0.7$ then update B and h and execute Method 3 else if $0.6 < \pi_i < 0.7$ then update B and execute Method 4 else if $0.4 < \pi_i < 0.6$ then update B and execute Method 2 else still execute method 1;

end

end

end

5) After δt, return to trust evaluation(1);

V. Simulation Analysis

In this section, we first evaluate robustness of each method against attacks and we evaluate positive/false alarms. The simulation experiments are implemented using MATLAB. We consider first an hybrid execution with ***20*** machines in an

Table I
FALSE ALARMS FOR DIFFERENT METHODS

Maliciousness	0.1	0.2	0.32	0.45	0.55	0.77
False Alarms 1	0	0	0	0	0.54	0.77
False Alarms 2	0	0	0	0.43	0.55	0.77
False Alarms 3	0	0	0	0.08	0.55	0.77
False Alarms 4	0	0	0	0.08	0.55	0.77

IaaS cloud and ***80*** machines running on local resources. We consider also 80% of cooperation in the verification process. Machines can hence be from either side: normal or attacker. The selection of machines strategy is chosen randomly. For trust model parameter, we choose to weight more the honesty with ***0.8***. For the others, we put ***0.15*** for cooperation and ***0.05*** for efficiency. Once security score obtained is lower than a threshold (e.g 0.5 for our simulation), the corresponding machine can be detected and considered as malicious. Unless previous works, we account for malicious collaborations among nodes that create coalitions for misreputation. Due to paper size restrictions, we focus on :

- the possible attacks and numerical results,
- positive/false alarm probability when we vary the malicious machines density,
- the effect of machine's ability (30% of maliciousness).

A. Attacks Analysis

- Coalition of nodes with the same objective
  We consider that malicious nodes select one attacker of the coalition. They assign to this attacker the total trust (1). The attacker evaluates nodes with correct peer-to-peer trust scores: 1 for the honest and 0 for the malicious (to increase the ability and not to be detected).
  We implement the attack with 30% of malicious nodes and we analyze the best method to discover it. As the attacker is hidden, it is difficult to detect it. We have tested the four methods. Our results show the following: $S^4_{attacker} < S^3_{attacker} < S^2_{attacker} < S^1_{attacker} < 0.44$. ($S^i_{attacker}$: security score for the supposed attacker in method i)
  So, we can detect the attacker with all the methods but especially with the method 4 (making trusted machines).
- Attack between malicious nodes

$Malicious - malicious$	1
$Malicious - honest$	1
$Honest - malicious$	0
$Honest - honest$	1

  We obtain the result represented below (the malicious nodes percentage is varied):

  In this attack, table I shows that 1^{st} method can perform better than the others. We can say here that we can't detect malicious nodes in all methods with 45% of malicious machines and more because malicious machines evaluate honest machines with 1 and they obtain good d_i.
- Attack against honest nodes

$Malicious - malicious$	0
$Malicious - honest$	0

In this case, the malicious nodes have ability "0" (in the three last methods). This is due to the malicious machines that evaluate privileged and trusted machines with zero. In this attack, all methods don't send false alarms (even with 70% maliciousness).

B. False Alarm Probability

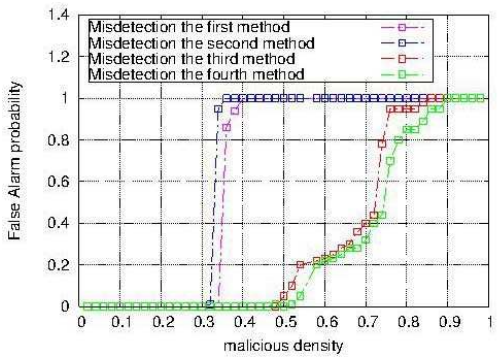

Figure 2. False alarm probability

In Fig. 2, we show that the misdetection ratio in the third and the fourth methods is 0 in the beginning and it increases only 48% of malicious machines for the third method and 52% for the fourth method. The first method sends false alarms after only 32% of maliciousness (only 36% for the second method).

C. Ability Evaluation

First, we consider method one and method two and we compare them in the same simulation conditions as explained before.

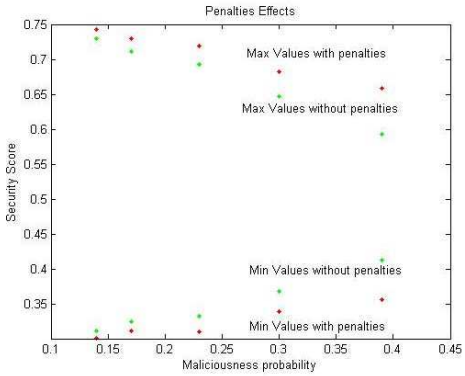

Figure 3. The benefit of ability calculation

In Fig. 3, we show that the detection and the global estimation of the machines'strategies are more clear with methods considering the calculation of machines'ability(red

points) than the others(green ones).

Second, we analyze the evolution of ability affected to machines. We notice that in all cases (considering less than 40% of malicious machines) $D^3_{max} > D^4_{max} > D^2_{max}$ and $D^2_{min} < D^4_{min} < D^3_{min}$ ($D^i_{min/max}$: min/max ability values in the method *i*). The third method has hence the best performance owing to the consideration of security score of trusted machines and the value assigned in $\boldsymbol{B}$ matrix.

VI. Conclusion

In this paper, we designed a new algorithm based on Perron theorem to detect malicious machines in the context of hybrid cloud. We considered a composite trust derived from honesty, cooperativeness and efficiency as a peer-to-peer evaluation. Our system uses four methods in different contexts to detect malicious machines considering different parameters (trust, privilege and ability). We claim that ability calculation and privilege parameters have many benefits. We show that many collective attacks can be detected. We analyzed the false alarm probabilities with respect to a detection threshold. Compromised nodes can be detected clearly with high accuracy and low false alarm probability. we show that by using privileges we can improve the detection process. The algorithm results help in making decision on whether to purchase execution resources from a supplier or not.

For future research directions, an appropriate optimization for effective detection is a challenging problem for prevention. Hence, a dynamic system using game theory and learning mechanism needs to be employed to detect attacks while minimizing the consumption of resources of defender nodes.

VII. Acknowledgement

The authors would like to thank Pr.Walid Ben Ameur for his help.

References

[1] P.Mell and T.Grance, "The NIST Definition of Cloud Computing (Draft)", January 2011

[2] X. Jing, "A brief survey on the security Model of Cloud Computing ", ISDCAB 2010.

[3] B.Kandukuri, " Cloud Security Issues", 2009 IEEE

[4] H.Takabi, " Security and privacy challenges in cloud computing environments " 2010 IEEE

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

[6] 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.

[7] 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, 2007, pp. 215-28.

[8] 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, Nov. 2009, pp. 1698-1712.

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

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

[11] M. Castro and B. Liskov, "Practical byzantine fault tolerance", In Proc.OF OSDI, New orleans, LA, 1999.

[12] L.Vokorokos and A.Balaz, "Host-based Intrusion Detection System",Intelligent Engineering Systems (INES), 14th International Conference, 2010

[13] S.Roschke, F.Cheng, C.Meinel, "Intrusion Detection in the Cloud", IEEE 2010

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

[15] F. Wang, C. Huang, J. Zhang, and C. Rong, "IDMTM: A novel intrusion detection mechanism based on trust model for ad hoc networks," 22nd IEEE International Conference on Advanced Information Networking and Applications, 2008, pp. 978-84.

[16] P. Ebinger and N. Bibmeyer, "TEREC: Trust evaluation and reputation exchange for cooperative intrusion detection in MANETs," 7th Annual Comm. Networks and Services Research Conf. 2009, pp. 378-385

[17] F.Bao, I.Chen, M.Chang and J.Cho, "Trust-based intrusion detection in wireless sensor networks",IEEE International Conference on Communications (ICC 2011), Kyoto, Japan, June 2011.

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