# Detecting false testimonies in reputation systems using self-organizing maps

Z. BANKOVIC , J. C. VALLEJO , D. FRAGA , AND J. M. MOYA

## Abstract

It has been demonstrated that rating trust and reputation of individual nodes is an effective approach in distributed environments in order to improve security, support decision-making and promote node collaboration. Nevertheless, these systems are vulnerable to deliberate false or unfair testimonies. In one scenario, the attackers collude to give negative feedback on the victim in order to lower or destroy its reputation. This attack is known as bad mouthing attack. In another scenario, a number of entities agree to give positive feedback on an entity (often with adversarial intentions). This attack is known as ballot stuffing. Both attack types can significantly deteriorate the performances of the network. The existing solutions for coping with these attacks are mainly concentrated on prevention techniques. In this work, we propose a solution that detects and isolates the abovementioned attackers, impeding them in this way to further spread their malicious activity. The approach is based on detecting outliers using clustering, in this case self-organizing maps. An important advantage of this approach is that we have no restrictions on training data, and thus there is no need for any data pre-processing. Testing results demonstrate the capability of the approach in detecting both bad mouthing and ballot stuffing attack in various scenarios.

*Keywords*: Reputation systems, bad mouthing, ballot stuffing, self-organizing maps.

## 1 Introduction

Trust and reputation [18] have recently been suggested as an effective security mechanism for open and distributed environments (Ad Hoc networks [2], WSNs [1, 2], P2P networks [3], etc.). In essence, the nodes that do not behave properly (according to the established policy of 'proper' behaviour) will have low reputation, so the rest of the nodes will avoid any collaboration with them, which is equivalent to its isolation from the network. Extensive research has been done on modelling and managing trust and reputation, and it has been demonstrated that rating trust and reputation of individual nodes is an effective approach in distributed environments in order to improve security, support decision-making and promote node collaboration.

There are many different definitions of trust and reputation, but in essence trust is a belief about future behaviour that one node holds in others and it is based on its own experience, thus its main characteristic is subjectivity. On the other hand, reputation is considered to be the global perception of the behaviour of a node based on the trust that others hold in it, thus considered to be objective [2]. The common way of defining a trust value is by using policies or credentials, so a node that behaves according to the established policy system will have high reputation and vice versa. Different variations of reputation systems have been developed for various purposes. The distinguishing characteristics of all of them are the following: representation of information and

classification, use of second-hand information, definition of trust and redemption and secondary response.

Alternatives to reputation systems can be incentive systems [4], where it is advantageous for the nodes to act in a way that the resulting global welfare is optimal. In these systems, the nodes receive some sort of payment if they behave properly. It is assumed here that rational peers have no reason (in other words, incentive) to deviate from the equilibrium behaviour. However, this kind of behaviour cannot be expected from malicious peers, so reputation systems have advantage in the cases their purpose is improved security.

Among the principal threats on reputation systems we can include colluding attacks. Here we distinguish two opposite scenarios: *ballot stuffing,* where a number of entities agree to give positive feedback on an entity (often with adversarial intentions), resulting in it quickly gaining high reputation, and the opposite case, known as *bad mouthing,* where the attackers collude to give negative feedback on the victim, resulting in its lowered or destroyed reputation. In this work, we concentrate on detecting both attacks.

Available solutions for coping with colluding attacks mostly rely on prevention techniques, such as cryptography. However, these methods only increase the effort the attacker has to make in order to make his attack successful. For this reason, we present a machine learning-based solution for detecting the sources of the attack. In this work, we are mainly concerned about the reputation systems used in distributed systems such as wireless sensor or mesh networks, where these are mainly used for isolating anomalous behaviour. However, the approach can easily be adapted for the situations where the entities in the network are humans. The main idea consists in detecting great inconsistencies in appraising an entity by other entities that have been in contact with it. The inconsistencies are treated as data outliers and are being detected using self-organizing maps (SOM).

The rest of the work is organized as follows. Previous work is surveyed in Section 2. Section 3 details the principles of the proposed solution, while Section 4 provides its evaluation. Finally, conclusions are drawn in Section 5.

## 2 Previous work

As already mentioned, the majority of the existing solutions for coping with colluding attacks rely on prevention techniques. A typical solution is the one given in [5] that relies on cryptography. However, with the existence of side channel attacks [6], the attacker can easily guess the secret keys and compromise the cryptography-based protocols. Another solution proposes to use 'controlled anonymity' [7], where the identities of the communicating entities are not known to each other. In this way, each entity has to provide ratings based on the quality of service provided, and as they can no longer identify their 'victims', bad-mouthing and negative discrimination can be avoided. However, this is not always possible, e.g. in the case of online hotel ratings.

Another approach [14] is based on establishing credibility of ratings based on the accuracy of their past recommendations. However, consistent behaviour is assumed, which is not always realistic. Other ideas include giving more weight to the ratings of more reliable entities [15], or removing extreme ratings [16], or even avoiding second-hand information in the process of calculating reputation [15].

All the previous solutions in essence increase the effort the attacker needs to introduce in order to compromise the system, but it will not protect the system from all the attacks. Thus, a second line of defense that would detect the attacks and stop their further spreading is necessary.

TABLE 1. Example of recommendations

| | $n1$ | $n2$ | $n3$ | $n4$ | $n5$ |
|---|---|---|---|---|---|
| 1 | 100 | 99 | 100 | 95 | 99 |
| 2 | 100 | 99 | 100 | 95 | 99 |
| 3 | 100 | 99 | 100 | 95 | 99 |
| 4 | 98 | 99 | 98 | 98 | 99 |
| 5 | 98 | 99 | 98 | 98 | 99 |
| 6 | 98 | 99 | 98 | 98 | 99 |
| 7 | 98 | 99 | 98 | 98 | 99 |
| 8 | 95 | 95 | 97 | 97 | 98 |
| 9 | 95 | 95 | 97 | 97 | 98 |
| 10 | 95 | 95 | 97 | 97 | 98 |

For this reason, in the recent past solutions for detecting these attacks on reputation systems started to appear. Machine learning has always been an attractive solution, given that it copes well with the uncertainties that exist in security. A representative solution using hierarchical clustering is given in [8]. This solution, as many others, after the training assign the clusters that contain the majority of the data as 'good' clusters. However, this imposes restrictions on training data, as if the algorithm does not process the 'unclean' data during the training, it will not be able to detect attacks. A solution based on graph theory is given in [9]. This solution, instead of using the count-based scheme that considers the number of accusations, uses the community-based scheme that achieves the detection of up to 90% of the attackers, which permits the correct operation of the system.

Thus, our aim is to design a detection-based solution that would overcome the abovementioned issues. Namely, we want to provide a solution that would not have any restrictions regarding training data, and that would be capable of detecting up to 100% of malicious entities.

## 3 Proposed solution

### *3.1 Feature definition and generation of the model*

For each entity, the feature vector is formed of the recommendation others give on it. The main idea is to find inconsistencies in recommendations. In the case the reputation system considers separately different services each entity has to offer, each service is characterized and examined independently. The characterization is based on the idea of $k$-grams and it is performed in equidistant moments of time using the recommendations between the consecutive moments. The features are different sets of recommendations ($k$-grams) and their occurrence or their frequency during the characterization period. Let the recommendations issued for the node $n$ from five different nodes during 10 sample periods be those given in Table 1.

In this case, the extracted $k$-grams, i.e. features, and their corresponding feature values are given in Table 2. From this example, it is obvious that the extracted number of different $k$-grams does not have to be the same in all characterization period. Thus, we cannot use any of the standard distance measurements.

TABLE 2. The characterization of the previous example

| Features | Occurrence | Frequency |
|---|---|---|
| 100 99 100 95 99 | 3 | 0.3 |
| 98 99 98 98 99 | 4 | 0.4 |
| 95 95 97 97 98 | 3 | 0.3 |

TABLE 3. Example of calculating distance

| V1 | | V2 | | Distance |
|---|---|---|---|---|
| Feature | Feature value | Feature | Feature value | $\|\Delta f\|$ |
| 100 99 100 95 99 | 0.33 | 100 99 100 95 99 | 0.33 | 0 |
| 98 99 98 98 99 | 0.33 | 98 99 98 98 99 | 0.33 | 0 |
| 95 95 97 97 98 | 0.33 | 95 95 97 97 98 | 0 | 0.33 |
| 98 98 97 97 96 | 0 | 98 98 97 97 96 | 0.33 | 0.33 |

Distance $= \sum |\Delta f| = 0.66$

The distance between the instances of the presented model is taken from [10]. It is designed to calculate the distance between two sequences. We have elected this one (among all given in [10]) since it is proven to be the most efficient in the terms of the absolute execution time. The deployed distance function is actually is equivalent to Manhattan distance after making the following assumption: the feature that does not exist in the first vector while exists in the second (and vice versa) actually exists with the value equal to 0, since we can say that it occurs with 0 frequency. In this way, we get two vectors of the same size and the distance between the centre and an input is between 0 (the vectors have the same features with the same feature values) and 2 (the vectors have different features with the values greater than 0). In the same way, if the set of the features of one is the subset of the feature set of the other, the distance will be between 0 and 1. An example of calculating distance is given in Table 3.

In many situations this can result in having huge number of combinations. In this case, it is necessary to apply one of the following possibilities for reducing this number. One possibility is to divide the range [0,100] into few equidistant ranges (usually three to five), and assign a unique value or meaning to all the values that belong to one range. This significantly reduces the number of possible $k$-grams. Another possibility is to take an average of the values that belong to a certain range.

## *3.2 Detection and isolation of collusion attack*

As previously mentioned, we treat attacks as data outliers and deploy clustering techniques. In this work, we will use the SOM algorithm, as they are relatively fast and inexpensive when the dimensionality of the data is huge, which can happen in our case.

There are two possible approaches for detecting outliers using clustering techniques [11] depending on the following two possibilities: detecting outlying clusters or detecting outlying data that belong to non-outlying clusters. For the first case, we calculate the average distance of each node to the rest of the nodes (or its closest neighborhood) (*MD*). In the latter case, we calculate quantization error (*QE*) of each input as the distance from its group centre. If we train the SOM algorithm with clean data, it is obvious that we will have the second scenario. On the other hand, if the traces of attacks existed during the training, both situations are possible. Thus, we can detect attacks in the situation we do or do not have the traces of attacks during the training, which means that we do not have any restrictions on the training data. This further means that we avoid time consuming and error-prone process of pre-processing the training data.

In the first step, we examine the recommendations for each node in order to find the inconsistencies. Having in mind that the attacks will often result in creating new $k$-grams, it is reasonable to assume that the extracted vector in the presence of attackers will not be a subset of any vector extracted in normal situation, thus the distance will never be lower than 1. Thus, the suspicious values of both *QE* and *MD* are greater than 1. Upon detecting suspicious values, we pass to the second step.

In the second step, the aim is to find the origin(s) of the suspicion. First, the deviations from either the mode, median or mean values of all the recommendations assigned to the node are calculated. The rationale behind this is that the majority of the nodes will assign correct recommendations, which will result in higher deviations from each of the above values in the cases of the wrong recommendations. This information can further be used in various ways, yet we choose to couple it with the reputation system, in the way the origins of the bad mouthing will result in lowered reputation, and their recommendations will not be considered. Thus, the calculated deviations are normalized to the range [0, 1], in the way the entity that is the origin of the maximal deviation has the lowest reputation, i.e. 0, whereas the origin of the minimal one has the highest reputation, i.e. 1. The reputations of the rest of the nodes are linearly interpolated between 0 and 1.

### *3.3 SOM algorithm*

The SOM algorithm [12, 17] follows the standard steps of SOM execution, as given in [12]. The only problem-specific point is the centre, i.e. node representation and updating. Each centre is implemented as a collection whose size can be changed on the fly and whose elements are the $k$-grams defined in the previous text with assigned occurrence or frequency. The adjustment of nodes (that belong to the map area to be adjusted) is performed in the following way:

- if an $n$-gram of the input instance $v(t)$ exists in the node, its value is modified according to the centre update given in [12];
- if an $n$-gram of the instance $v(t)$ does not exist in the cluster centre, the $n$-gram is added to the centre with occurrence equal to 1.

## 4 Experimental evaluation

The proposed algorithm has been tested on the reputation systems simulator developed by our research group and designed using the C++ programming language. The network consists of a number of distributed entities, and it can simulate a number of distributed systems, such as wireless sensor networks, wireless mesh networks, etc. The reputation is implemented in the class *ReputationServer*.

In the testing scenario, the entities assign recommendations to their neighbours. The attacks (bad mouthing and ballot stuffing) are initiated at a certain moment and are composed of a number of malicious nodes that assign false reputation (low in the case of bad mouthing and high in the case of ballot stuffing) to some of their neighbours in a random fashion. The time in the simulator is measured in time ticks, where the ticks are the moments of time the recommendations are being published to the rest of the world.

In the following experiments, we will present the performance of the approach in various scenarios, varying the attack strength and the starting point of the attack. There will be two typical situations: in the first case the attack will start after the end of training, so the training will be performed with 'clean' data, while in the second case we will have the situations where the training data contain the traces of attacks as well. The aim of these experiments is to show that no constraints on training data exist and that the approach is capable of detecting 100% of the malicious entities even in the cases they make a significant part of the network.

The scenario is based on 50 entities that can take one of the possible 300 positions. In the experiments concerning bad mouthing scenario, we fix the number of the entities that take part in the attack, where each entity gives false accusations about 10% of the entities in its closest neighbourhood. On the other hand, in the experiments concerning ballot stuffing, few nodes collude to give positive feedback on a certain node that has adversarial activities. Thus, we can say that bad mouthing is implemented in a distributed manner, while ballot stuffing is rather local.

## *4.1 Bad mouthing detection*

In the first experiment the attack starts after the end of training, so the training is performed with so-called clean data. In Figure 1a, we present the reputation evolution of the situation of the most aggressive attack that can be detected with 100% detection rate with 0% false positive rate, which is the case when the bad nodes make 28.6% of all the nodes. We can observe that the reputation of the malicious nodes has been lowered to 0. In Figure 1b, the dependence of both detection rate (DR) and false positive rate (FPR) on the number of bad nodes is presented. As expected, as the number of bad nodes increases, detection rate decreases, while false positive rate increases. The simulation stops at the moment the total number of bad entities makes 61.5% of total, when the undetected bad entities make the majority of non-isolated entities. In this case, the reputation system becomes compromised and stops providing correct reputation values.

In the second experiment, we will present the results of the situation where the SOM algorithm is being trained with the data that contain the traces of attacks. Bad nodes make 28.6% of all the nodes. The results are presented in Figure 2. We can observe that the detection rate decreases, as the amount of the clean data during the training also decreases, while the false positive rate increases. This could be expected, as the 'unclean' data makes bigger part of the training data, it is harder to distinguish it.

## *4.2 Ballot stuffing detection*

The conducted experiments concerning ballot stuffing attack are equivalent to the previous ones. In the first experiment, the attack starts after the end of training, so the training is performed with so-called clean data. In Figure 3a, we present the reputation evolution of the situation of the most aggressive attack that can be detected with 100% detection rate with 0% false positive rate, which is the case when the bad nodes make 20% of all the nodes. In this can we can observe one more time

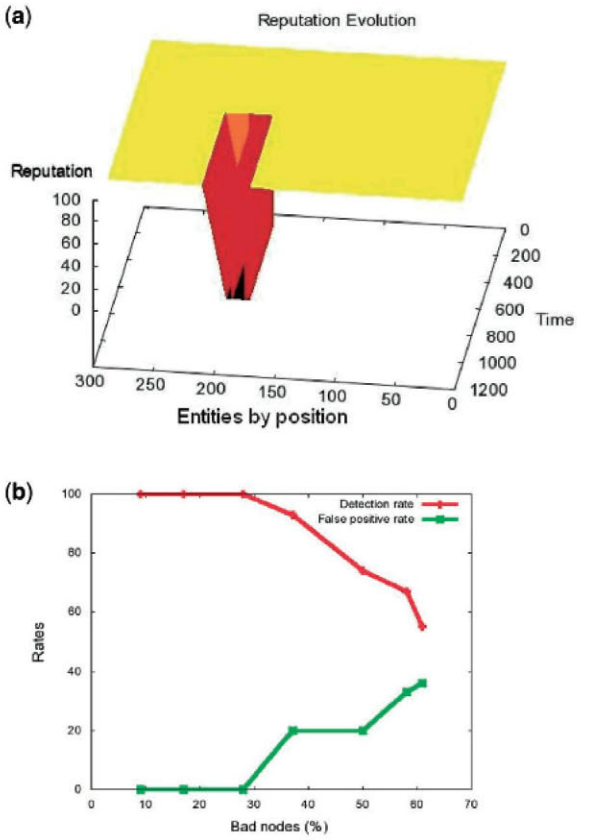


FIG. 1. Experiments with different number of bad nodes after training with clean data—bad mouthing attack. (a) Bad nodes—28.6% of all; (b) DR and FPR versus number of bad nodes.

that the reputation of the malicious nodes has been lowered to 0. In Figure 3b, the dependence of both detection rate (DR) and false positive rate (FPR) on the number of bad nodes is presented. As expected, as the number of bad nodes increases, detection rate decreases, while false positive rate increases. Comparing to the bad mouthing case, in this case we can observe that the detection close to perfect is maintained loner (almost 40% of the nodes are bad). However, soon after it the detector becomes useless, while in the previous case it can function properly if up to 60% of the nodes are bad. This occurs due to the implementation of the attacks in the simulations, i.e. to the fact bad mouthing is global and distributed, while ballot stuffing is local. For this reason, ballot stuffing is easier to detect, resulting in higher DR and lower FPR. However, as bad nodes become significant part on the local level, the attack cannot be detected.

In the second experiment, we will present the results of the situation where the SOM algorithm is being trained with the data that contain the traces of attacks. Bad nodes make 20% of all the nodes. The results are presented in Figure 4. In the same way as in the previous case, we can observe that the detection rate decreases, as the amount of the clean data during the training also decreases, while the false positive rate increases. However, relatively good detection (100% DR and low false positive rate) is maintained longer than in the case of bad mouthing detection. However, the change

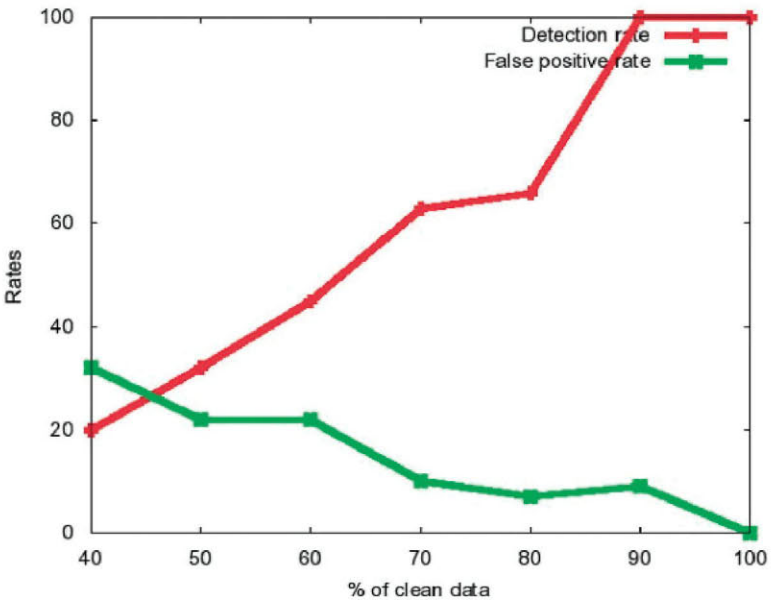


FIG. 2. DR and FPR versus amount of clean data during the training—bad mouthing attack.

in this case is more abrupt. This again occurs due to the previously mentioned implementation of the attacks.

## 5 Conclusions

In this work, we have presented an approach for detecting two collusion attacks on reputation systems of different kind (bad mouthing and ballot stuffing). The approach is based on outlier detection using SOM algorithm. We have proven that in the case of bad mouthing detection, it is capable of achieving 100% detection rate with 0% false positive rate if trained with clean data and up to 28.6% of the nodes are malicious. For the last case, when 28.6% of the nodes are malicious, the detection of the attack is possible if at least 40% of the data are clean.

On the other hand, in the case of ballot stuffing detection, we were able to achieve perfect detection if up to 20% of the nodes are bad, while false positive rate lightly increases if less than 40% of the nodes are bad.

In the future we will test the performances of other clustering techniques instead of SOM, such as Growing Neural Gas [13].

## Acknowledgements

This work was funded by the Spanish Ministry of Science and Innovation, under Research Grant (AMILCAR TEC2009-14595-C02-01), and the P8/08 within the National Plan for Scientific Research, Development and Technological Innovation 2008-2011.

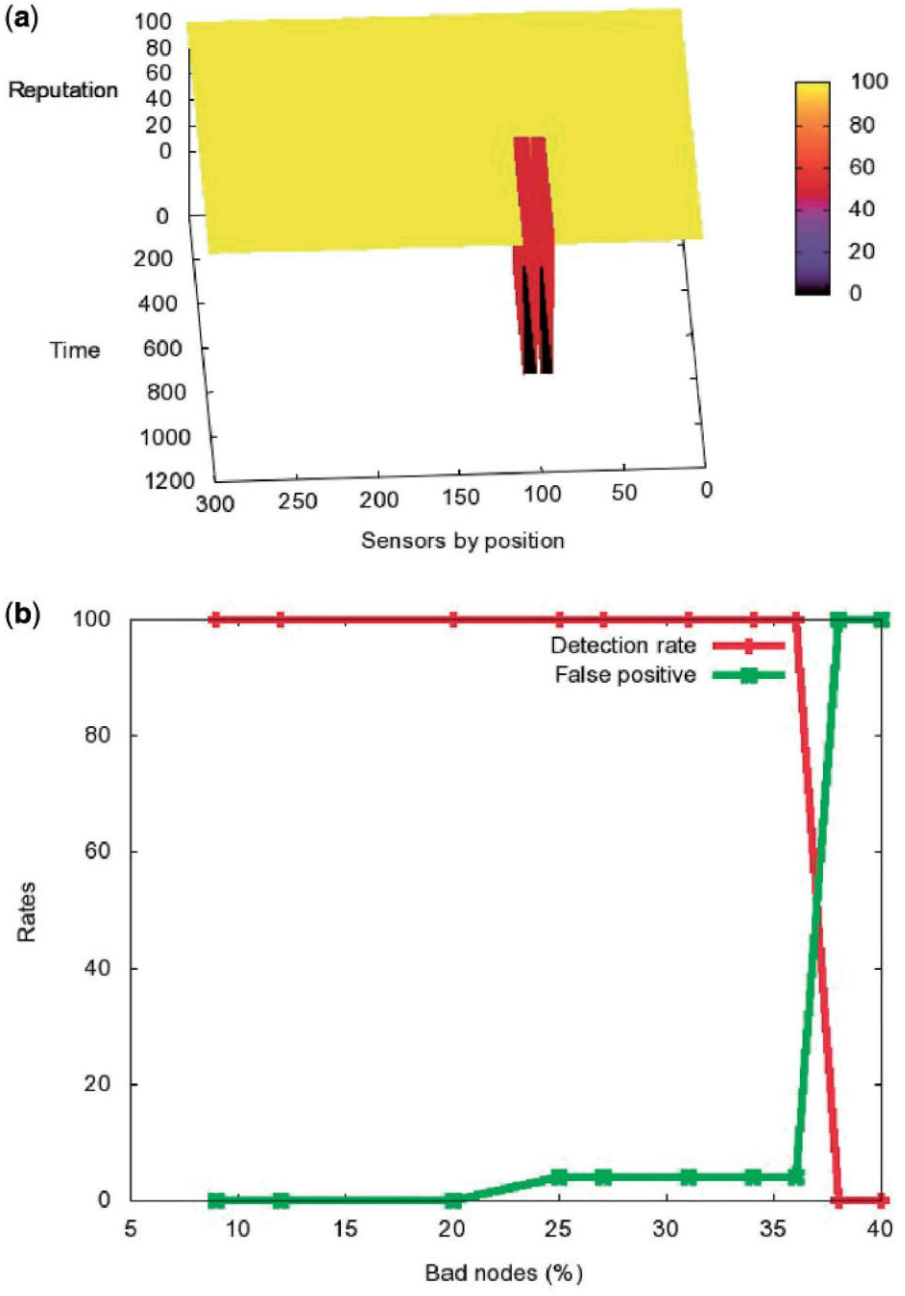


FIG. 3. Experiments with different number of bad nodes after training with clean data—Ballot Stuffing attack. (a) Bad nodes—20% of all; (b) DR and FPR versus number of bad nodes.

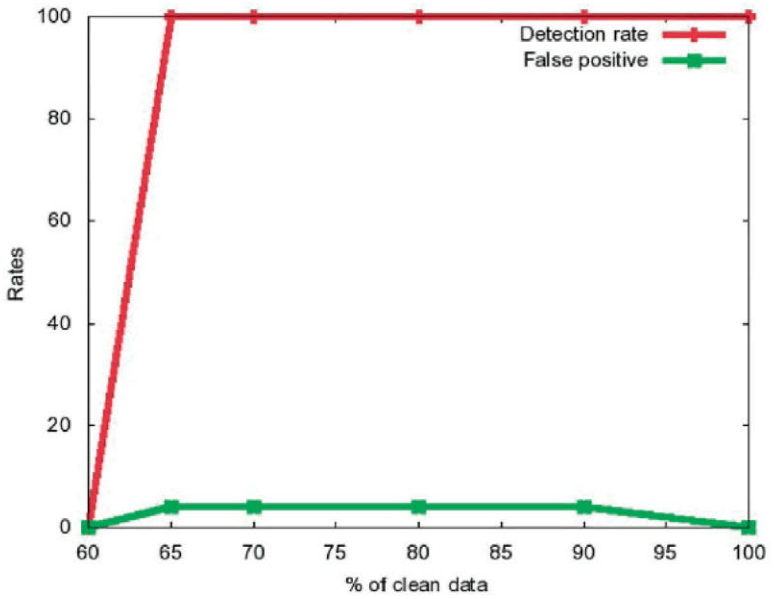


FIG. 4. DR and FPR versus amount of clean data during the training—Ballot stuffing attack.

## References

[1] J. M. Moya, A. Araujo, Z. Bankovic, J. M. de Goyeneche, J. C. Vallejo, P. Malagon, D. Villanueva, D. Fraga, E. Romero, and J. Blesa. Improving security for SCADA sensor networks with reputation systems and Self-Organizing maps. *Sensors*, **9**, 9380–9397, 2009.

[2] A. Boukerch, L. Xu, and K. EL-Khatib. Trust-based security for wireless ad hoc and sensor networks. *Computer Communications*, **30**, 2413–2427, 2007.

[3] T. G. Papaioannou and G. D Stamoulis. Effective use of reputation of peer-to-peer environments. In *Proceedings of IEEE/ACM CCGRID 2004, GP2PC Workshop. IEEE Computer Society: Chicago, IL, USA, April 19–22*. pp. 259–268, 2004.

[4] P. Antoniadis, C. Courcoubetis, E. Efstathiou, G. Polyzos, and B. Strulo. Peer-to-Peer wireless LAN consortia: economic modeling and architecture. In *3rd IEEE International Conference on Peer-to-Peer Computing*, pp. 198–199. IEEE Computer Society, 2003.

[5] J.-K. Lou, K.-T. Chen, and C.-L Lei. A collusion-resistant automation scheme for social moderation systems. In *6th IEEE Conference on Consumer Communications and Networking Conference*, pp. 571–575. IEEE Press, 2009.

[6] H. Bar El. *Introduction to Side Channel Attacks*. White Paper, Discretix Technologies Ltd, 2003.

[7] C. Dellarocas. Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In *2nd ACM conference on Electronic commerce*, pp. 150–157. ACM, 2000.

[8] S. Liu, C. Miao, Y.-L. Theng, and A. C. Kot. A clustering approach to filtering unfair testimonies for reputation systems (Extended Abstract). In *International Conference on Autonomous Agents and Multiagent Systems*, W. van der Hoek, G. Kaminka, Y. Lespérance, M. Luck and S. Sen, eds, pp. 1577–1578, 2010.

[9] J.-K. Lou, K.-T. Chen, and C.-L. Lei.. A collusion-resistant automation scheme for social moderation systems. In *6th IEEE Conference on Consumer Communications and Networking Conference*, pp. 571–575. IEEE Press, 2009.

[10] K. Rieck and P. Laskov. Linear time computation of similarity for sequential data. *Journal of Machine Learning Research*, **9**, 23–48, 2008.

[11] J. Lopez, R. Roman, I. Agudo, and C. Fernandez-Gago. Trust management systems for wireless sensor networks: best practices. *Computer Communications*, **33**, 1086–1093, 2010.

[12] S. Haykin. *Neural Networks - A Comprehensive Foundation*, 2nd edn. Prentice-Hall, 1999.

[13] B. Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems*, G. Tesauro, D. S. Touretzky, T. K. Leen, eds, pp. 625–632, MIT Press, 1995.

[14] W. T. Teacy, J. Patel, N. R Jennings, and M. Luck. Travos: trust and reputation in the context of inaccurate information sources. *Autonomous Agents and Multi-Agent Systems*, **12**, 183–198, 2006.

[15] G. Zacharia, A. Moukas, and P. Maes. Collaborative reputation mechanisms in electronic marketplaces. In *HICSS '99: Proceedings of the 32nd Annual Hawaii International Conference on System Sciences*, Vol. 8, p. 8026, IEEE Computer Society, 1999.

[16] A. Whitby, A. Josang, and J. Indulska. Filtering out unfair ratings in Bayesian reputation systems. In *Proceeding of the 7th International Workshop on Trust in Agent Societies*, Autonomous Agents & Multi Agent Systems, 2004.

[17] E. Corchado and Á. Herrero. Neural visualization of network traffic data for intrusion detection. *Applied Soft Computing 2008*, **11**, 2042–2056, 2011.

[18] A. Herzig, E. Lorini, J. F. Hübner, and L. Vercouter. A logic of trust and reputation. *Logic Journal of IGPL* **18**, 214–244, 2010.