Defense of Trust Management Vulnerabilities in Distributed Networks

Yan (Lindsay) Sun*, Zhu Han†, and K. J. Ray Liu‡

* Department of Electrical and Computer Engineering
University of Rhode Island, RI 02881

† Department of Electrical and Computer Engineering
Boise State University, ID 83725

‡ Department of Electrical and Computer Engineering
University of Maryland, College Park, MD 20742

Abstract—**Establishing trust among distributed network entities has been recognized as a powerful tool to secure distributed networks such as MANET and sensor networks. Similar to most security schemes, trust establishment methods themselves can be vulnerable to attacks. In this paper, we investigate the benefits of introducing trust into distributed networks, the vulnerabilities in trust establishment methods, and the defense mechanisms. Five attacks against trust establishment methods are identified and the defense techniques are developed. Effectiveness of the attacks and the defense is demonstrated in the scenarios of securing routing protocols and detecting malicious nodes in MANET.**

I. Introduction

It is well known that mobile ad hoc networks (MANET) and sensor networks face many security challenges [1]. Many of the challenges are due to the fact that those networks inherently rely on cooperation among distributed entities. However, cooperation is fragile and can be easily damaged by selfish behaviors, malicious attacks and even unintentional misconfiguration. The bottom-line problem is that distributed entities take actions without knowing whether they can trust the entities that they are collaborating with.

When network entities do not know how to trust each other, they either naïvely believe in good intentions of other entities or are paranoid. The naïve users can suffer badly from malicious attacks, whereas the paranoid users can cause the network to suffer from low availability and efficiency.

Without trust, a network entity has to delegate a task, such as sending data to a destination, to someone that may not be trustworthy. This would lead to failures of critical network functions, such as routing. Furthermore, the unknown risk of interacting with untrustworthy parties will reduce the incentive for cooperation in distributed systems. It is well known that trust is the driving force for cooperation in social networks. A similar principle can also be applied to distributed networks, especially when the network entities do not belong to a single authority.

The research on the subject of trust in computer networks has been extensively performed for a wide range of applications, including authorization and access control, electronics commerce, P2P networks, web-based services selection, distributed computing, and pervasive computing [2], [3]. Incorporating the notion of trust into MANET and sensor networks has recently gained a large amount of research attention [4]–[8]. Whereas traditional security approaches are inadequate or too complicated to protect such autonomous networks from possibly compromised nodes, trust-based approaches are thus investigated as a complementary security mechanism.

The basic idea is to generate trust values describing the trustworthiness, reliability or competence of individual nodes, based on some monitoring schemes. Such trust information is then used to assist routing [5], data aggregation [7], malicious node detection [6], and even time-synchronization. Another direction is to understand how trust stimulates cooperation in autonomous wireless networks [4].

However, there still exists a big gap between existing solutions and a systematically designed trust infrastructure. There are many open questions. What is the meaning of trust metrics? What are the mathematical properties of trust metrics? How can

This work is partially supported by NSF Award #0643532.

the trust establishment approaches be analyzed and validated? Is trust establishment process vulnerable to attacks? Among these questions, the last one, attack and defense, receives the least amount of research attention. Although there are a few work studying one or several possible vulnerabilities [9] in e-commerce and P2P applications, there is a lack of systematic treatment to this problem.

In this paper, we investigate the attacks against distributed trust establishment approaches and defense mechanisms. In particular, we first summarize the roles of trust and the core design issues of trust establishment mechanisms in distributed networks, in Section II and III, respectively. The attacks and protection methods will be described in Section IV. Simulation results in MANET are shown in Section V, followed by a conclusion in Section VI.

II. Role of Trust

There has been a great deal of confusion about the topic of trust. Many researchers recognize trust as an essential element in security solutions for distributed systems [2]. However, it is still not clear *what trust is* and *how exactly trust can benefit network security* [10]. We synthesize the roles that trust can play in MANET and sensor networks.

Prediction and Diagnosis When a network entity establishes trust in other network entities, it can *predict* the future behaviors of others and *diagnose* their security properties. This prediction and diagnosis can solve or partially solve the following four important problems.

- *Assistance in decision-making to improve security and robustness:* With a prediction of the behaviors of other entities, a network entity can avoid collaborating with untrustworthy entities, which can greatly reduce the chance of being attacked. For example, a node can choose the most trustworthy route to deliver its packets in MANET.
- *Adaptation to risk, leading to flexible security solutions:* The prediction of nodes' future behavior directly determines the risk faced by the network. Given the risk, the network can adapt its operation accordingly. For example, stronger security mechanisms should be employed when risk is high.
- *Misbehavior detection:* Trust evaluation leads to a natural security policy that the network participants with low trust values should be investigated or eliminated. Thus, trust information can be used to detect misbehaving network entities.
- *Quantitative assessment on system-level security properties:* With the assessment on trustworthiness of individual network entities, it is possible to evaluate the trustworthiness of the entire network. For example, the distribution of the trust values of network entities can be used to represent the healthiness of the network.

Simplification and Abstraction With raising security threats, the design of many network protocols and applications must consider the possibility that some participants will not follow the protocols honestly. Currently, this issue is considered by individual protocols or applications, which leads to repetitive monitoring and high complexity. When trust information is produced by an infrastructure managed by the network, the designer of network protocols can simply take trust values and integrate them into the design, without worrying about how to determine whether a node is trustworthy or not.

Integrating Social Needs into Design "The most vexing security problems today are not just failures of technology, but result from the interaction between human behavior and technology" [1]. Trust can be a bridge between social needs and security solutions. For example, trust infrastructure can *stimulate cooperation* because there is an incentive for users/network entities to build high reputation/trust values.

III. Core Design Issues of Trust Establishment Methods

Trust can be established in a centralized or distributed manner. Obviously, MANET and sensor networks perfer distributed trust management, where each network entity maintains a *trust manager*. The basic elements of such a trust manager is illustrated in Figure 1 and described in this section.

Trust Record stores information about trust relationship and associated trust values. A *trust relationship* is always established between two parties for a specific action. That is, one party trusts the other party to perform an action. In this work, the first party is referred to as the *subject* and the second party as the *agent*. A notation $\{subject :$

$agent, action\}$ is introduced to represent a trust relationship. For each trust relationship, one or multiple numerical values, referred to as *trust values*, describe the level of trustworthiness.

There are two common ways to establish trust in computer networks. First, when the subject can directly observe the agent's behavior, *direct trust* can be established. Second, when the subject receives recommendations from other entities about the agent, *indirect trust* can be established.

Direct Trust is established upon observations on whether the previous interactions between the subject and the agent are successful. The observation is often described by two variables: s denoting the number of successful interactions and f denoting the number of failed interactions. For example, in the beta-function based method [2], the direct trust value is calculated as $\frac{s+1}{s+f+2}$.

Recommendation trust is a special type of direct trust. It is for trust relationship $\{subject$: $agent$, $making\ correct\ recommendations\}$. When the subject can judge whether a recommendation is correct or not, the subject calculates the recommendation trust from s_r and f_r values, where s_r and f_r are the number of good and bad recommendations received from the agent, respectively. This judgement is often done by checking consistence between observations and recommendations, or among multiple recommendations. When using the beta-function based methods, the recommendation trust can be calculated as $\frac{s_r+1}{s_r+f_r+2}$.

Indirect Trust: Trust can transit through third parties. For example, if A has established a recommendation trust relationship with B and B has established a trust relationship with Y, then A can trust Y to a certain degree if B tells A its trust opinion (i.e. recommendation) about Y. This phenomenon is called *trust propagation*. Indirect trust is established through trust propagations.

Two key factors determine indirect trust. The first is when and from whom the subject can collect recommendations. For example, in a sensor network, a sensor may only get recommendations from its neighbors when there is a significant change in their trust records. This affects the number of available recommendations and the overhead of collecting recommendations.

The second is to determine how to calculate indirect trust value based on recommendations. When node B establishes direct trust in node Y and node A establishes recommendation trust in node B, $A-B-Y$ is one recommendation path. One recommendation path can contain more than two hops, such as $A-B_1-B_2-\cdots-Y$, and there may exist multiple recommendation paths, such as $A-B_1-Y$, $A-B_2-Y$, $\cdots$ etc. *Trust models* determines how to calculate indirect trust between A and Y from trust propagation paths. There have been many trust models proposed for various applications [2].

IV. Attacks and Protection

As we will show in the simulation section, trust management can effectively improve network performance and detect malicious entities. Thus, it is an attractive target for attackers. In this section, we discuss attacks and protection.

A. Bad Mouthing Attack

As long as recommendations are taken into consideration, malicious parties can provide dishonest recommendations [9] to frame up good parties and/or boost trust values of malicious peers. This attack, referred to as the bad mouthing attack [6], is the most straightforward attack. In this paper, we defend against the bad mouthing attack by formally building and utilizing recommendation trust.

First, the recommendation trust is treated separately from regular direct trust, and can only be established based on previous recommendation behaviors. As discussed in Section III, recommendation trust is determined by s_r and f_r values, which are independent of whether the agent has performed the action or not.

Second, we add a necessary condition to trust propagation. That is, trust can propagate along path $A-B-Y$ if the recommendation trust between A and B is greater than a threshold.

Third, we develop a generic trust-based malicious node detection algorithm based on M trust relationships as $\{A : B, action_i\}$, for $i = 1, 2, \cdots, M$. The trust relationships can be direct, indirect, or recommendation trust. For each trust relationship, we use (α_i, β_i) to represent the trust values. There are two ways to calculate (α_i, β_i) values.

- When one can estimate the numbers of successful and failed interactions for $\{A : B, action_i\}$ as s_i and f_i, respectively, one will calculate

$\alpha_i = s_i + 1$ and $\beta_i = f_i + 1$. This is often used for direct trust relationships.

- When one can estimate mean (m_i) and variance (v_i) of the distribution of the probability (p) that the agent will perform the action, one will calculate $\alpha_i = m_i\left(\frac{m_i(1-m_i)}{v_i} - 1\right)$ and $\beta_i = (1-m_i)\left(\frac{m_i(1-m_i)}{v_i} - 1\right)$. This is often used for indirect trust relationships.

The above calculations come from the assumption that probability p follows a beta distribution [2]. (α, β) values are the parameters of the beta distribution, and (m, v) values are the mean and variance of the beta distribution. There is a one-to-one mapping between (α, β) and (m, v). The physical meanings of α and β determine that $\alpha = s+1$ and $\beta = f+1$.

Then, a node is detected as malicious if $\frac{\alpha}{\alpha+\beta} <$ threshold, where $\alpha = \sum_i w_i(\alpha_i - 1) + 1$ and $\beta = \sum_i w_i(\beta_i - 1) + 1$. Here, $\{w_i\}$ is a set of positive weigh vectors and $w_i \leq 1$. This malicious node detection algorithm considers multiple trust relationships including recommendation trust, which can lead to an detection of the bad-mouthing attackers.

B. *On-off Attack*

On-off attack means that malicious entities behave well and badly alternatively, hoping that they can remain undetected while causing damage. This attack exploits the dynamic properties of trust through time-domain inconsistence. Next, we first discuss the dynamic properties of trust and then demonstrate this attack and its solution.

Trust is a dynamic event. A good entity may be compromised and turned into a malicious one, while an incompetent entity may become competent due to environmental changes. In wireless networks, for example, a mobile node may experience bad channel condition at a certain location and has low trust value associated with forwarding packets. After it moves to a new location where the channel condition is good, some mechanisms should be in place to recover its trust value.

In order to track this dynamics, the observation made long time ago should not carry the same weight as that made recently. The most commonly used technique that addresses this issue is to introduce a forgetting factor. That is, performing K good actions at time t_1 is equivalent to performing $K\hat{\beta}^{t_2-t_1}$ good actions at time t_2, where $\hat{\beta}(0 < \hat{\beta} \leq 1)$ is often referred to as the *forgetting factor*. In the existing schemes, using a fixed forgetting factor has been taken for granted. We discover, however, the existing forgetting scheme can facilitate the on-off attack on trust management.

Let's demonstrate such an attack through an example. Assume that an attacker behaves in the following four stages: (1) first behaves well for 100 times, (2) then behaves badly for 100 times, (3) then stops doing anything for a while, (4) and then behaves well again. Figure 2 shows how the trust value of this attacker changes. The horizontal axis is the number of good behaviors minus the number of bad behaviors, while the vertical axis is the estimated probability that the attacker will perform a good action in the next round. This probability, denoted by p, is estimated as $p = \frac{s+1}{s+f+2}$, where s is the number of good actions and f is the number of bad actions. In fact, p is the mean of the beta distribution discussed in Section IV-A. In this section, p is also called the probability-based trust value.

In Figure 2, the dashed line is for $\hat{\beta} = 1$ and the solid line is for $\hat{\beta} = 0.0001$. We observe

1. When the system does not forget, i.e. $\hat{\beta} = 1$, this attacker has high trust value in stage 2. That is, the attacker can have good trust value even after it turns bad.
2. When using a small forgetting factor, the attacker can regain trust by simply waiting in stage 3, or regain trust quickly after behaving well for just a few times in stage 4.

From the attackers' point of view, they can take advantages of the system one way or another, no matter what value of the forgetting factor is chosen.

To defend against the on-off attack, we propose a scheme that is inspired by a social phenomenon – while it takes long-time interaction and consistent good behavior to build up a good reputation, only a few bad actions can ruin the reputation. This implies that bad behavior is remembered for a longer time than good behavior. We mimic this social phenomenon by introducing an *adaptive forgetting scheme*. Instead of using a fixed forgetting factor, $\hat{\beta}$ is a function of the current trust value. For example, we can choose

$$\hat{\beta} = 1 - p \tag{1}$$

$$\text{or, } \hat{\beta} = \beta_1 \text{ for } p \geq 0.5; \text{ and } \hat{\beta} = \beta_2 \text{ for } p < 0.5, \quad (2)$$

where $0 < \beta_1 << \beta_2 \leq 1$. Figure 3 demonstrates the probability-based trust value changes when using these two adaptive forgetting schemes. One can see that the trust value keeps up with the entity's current status after the entity turns bad. An entity can recover its trust value after bad behaviors, and this recovery requires many good actions. The adaptive forgetting schemes solve the problems in the traditional forgetting schemes.

To integrate the dynamic forgetting scheme into the trust manager, the trust record needs to maintain s and f values, as well as the time t when this record was last updated, for each trust relationship. Assume there are Δ_s and Δ_f *additional* success and failed interactions between time t and t_2. Then, at time t_2, s is updated to $(s\hat{\beta}^{t_2-t} + \Delta_s)$ and f is updated to $(f\hat{\beta}^{t_2-t} + \Delta_f)$, where $\hat{\beta}$ is determined by the adaptive forgetting scheme.

C. Conflicting Behavior Attack

While an attacker can behave inconsistently in the time domain, it can also behave inconsistently in the user domain. In particular, malicious entities can impair good nodes' recommendation trust by performing differently to different peers. This attack is referred to as the conflicting behavior attack.

For example, the attacker X can always behave well to one group of nodes, denoted by G_1, and behave badly to another group of nodes, denoted by G_2. When a node $A \in G_1$ provides recommendation about X to a node $B \in G_2$, this recommendation will disagree with node B's observation about X. As a result, B will lower its recommendation trust in A. If many collaborative attackers launch this attack, the nodes in G_1 will assign low recommendation trust to the nodes in G_2. This results in inaccurate recommendation trust. The influence of this attack and a simple defense method will be shown in Section V.

D. Sybil Attack and Newcomer Attack

If a malicious node can create several faked IDs, the trust management system suffers from the *sybil attack*. The faked IDs can share or even take the blame, which otherwise should be given to the malicious node.

If a malicious node can easily register as a new user, trust management suffers from the *newcomer attack*. Here, a malicious node can easily remove its bad history by registering as a new user.

The defense to the sybil attack and newcomer attack does not rely on the design of trust management, but the authentication and access control, which make registering a new ID or a faked ID difficult. In this paper, we point out these two attacks in order to have an inclusive discussion on vulnerabilities in trust establishment systems.

V. Performance Evaluation

A. Trust Management in MANET

In mobile ad hoc networks, securing routing protocols is one of the fundamental challenges. In this paper, the impact of the attacks and anti-attack methods is evaluated in the application of trust-assisted ad hoc routing. The scheme in [5] is chosen. In this scheme, trust information is used to handle and detect the gray hole attack against routing, in which malicious nodes selectively drop data packets. The key elements of this scheme are summarized as follows.

- The trust values associated with two actions: forwarding packets and making recommendations, are investigated.
- When a source node wants to establish a route to the destination node, the source node first find multiple routes to the destination. Then, the source node checks its own trust record to see whether it has trust relationship with the nodes on the routes. If not, the source node broadcast a recommendation request message to its neighbors and waits for replies.
- Upon receiving a recommendation request message, the other nodes in the network will reply if they have information needed by the source node. They will also check whether the request message has propagated over more than a certain number of hops. If not, they will forward the request message to their neighbors.
- The source node collects replies and calculate/update the trust values of the nodes on the routes using a trust model.
- The source node calculates the trustworthiness of a route as the multiplication of the trust values of the nodes on the route. The source

node then transmits packets through the most trustworthy route.

- During data transmission, the source node observes the packet forwarding behavior of the nodes on the route, through a lightweight self-reporting mechanism.
- After data transmission, the source node compares its observation and the recommendations it received previously. If the difference between a recommendation and the observation is smaller than a threshold, this recommendation is marked as good. Otherwise, this recommendation is marked as bad. Then, recommendation trust is updated accordingly.
- Finally, the source node updates its direct trust in the nodes that have forwarded packets for it. The nodes with trust values lower than a threshold can be detected as malicious.

B. Simulation Results

An event-driven simulator is built to simulate the trust-assisted routing in MANET. In the physical layer, a fixed transmission range of 300m is used. The MAC layer protocol is IEEE 802.11 distributed coordination function (DCF), and the routing protocol is dynamic source routing (DSR). 50 honest nodes are randomly located in a 1000m by 1000m rectangular area. 50 traffic pairs with Poisson packet arrival are randomly generated. The routing protocol finds up to 5 routes between source and destination. Maximal route length is 10 hops. Mobility model is the random way point model with a slight modification. A node starts at a random position, waits for a duration called the pause time that is modeled as a random variable with exponential distribution, then randomly chooses a new location and moves towards the new location with a velocity uniformly chosen between 0 and $v_{max} = 10$ meters/second. When it arrives at the new location, it waits for another random pause time and repeats the process. The average pause time is 300 seconds. For trust management, the recommendation request messages propagate no more than 3 hops.

Next, we show the advantages of trust management, and the effects of the bad mouthing attack and the conflicting behavior attack. The performance of the on-off attack and the adaptive forgetting scheme has been shown previously in Section IV-B.

1) Advantage of Trust Management: In Figure 4, three schemes are compared: (1) baseline system without attackers; (2) baseline system without trust management but with 5 attackers launching the gray-hole attack, in which they randomly drop about 90% of packets passing through them; (3) the system with 5 gray-hole attackers and trust management.

Figure 4 shows the percentage of the packets that are successfully transmitted, which represents network throughput, as a function of time. Three observations are made. First, network throughput can be significantly degraded by malicious attackers. Second, after using trust management, the network performance can be recovered because it enables the route selection process that avoids untrustworthy nodes. Third, when the simulation time increases, trust management can bring the performance close to that in the case of no attackers because more accurate trust records are built up over time.

2) Bad Mouthing Attack: The attackers launch the gray-hole attack and the bad-mouthing attack, in which they provide good (bad) recommendations for bad (good) nodes.

We introduce a metric called MDP to describe the <u>m</u>alicious node <u>d</u>etection <u>p</u>erformance. Each node performs malicious node detection locally. Let D_i denote the number of good nodes that have detected that node n_i is malicious, $\mathbf{M}$ denote the set of malicious nodes, and $\mathbf{G}$ denote the set of good nodes. Then, MDP is defined as $\frac{\sum_{i:n_i \in \mathbf{M}} D_i}{|\mathbf{M}|}$, which represents the average detection rate. Similarly, we can define another metric as $\frac{\sum_{i:n_i \in \mathbf{G}} D_i}{|\mathbf{G}|}$, which describes the false alarm rate. For all simulations, we choose the detection threshold such that the false alarm rate is sufficiently small. Thus, we only show MDP as the performance index.

We compare two malicious node detection methods. In the first method, only direct and indirect trust is used. In the second case, direct, indirect and recommendation trust is used. Figure 5 shows the MDP of this two detection methods. It is seen that using recommendation trust in the detection process can significantly increase the detection rate and therefore hold back the bad mouthing attack.

3) Conflicting-behavior Attack: In this attack, the attackers drop packets that are originated from one group of nodes, denoted by G_2, and do not drop packets that are originated from another group of nodes, denoted by G_1. The *attack percentage* is defined as the number of nodes in G_2 divided by the total number of nodes.

While launching this attack, the attackers have four strategies to provide recommendations:

R1: no recommendations to G_2, and honest recommendations to G_1;

R2: no recommendations to G_2, and no recommendations to G_1;

R3: bad recommendations to G_2, and no recommendations to G_1;

R4: bad recommendations to G_2, and honest recommendations to G_1.

In R1 and R4, the attackers can in fact help the network performance by providing good recommendations, especially when the attack percentage is low or at the beginning (when most good nodes have not established reliable recommendation trust). In R1, malicious nodes can have higher recommendation trust than good nodes. Thus, it is harmful to use the recommendation trust in the malicious node detection algorithm. The similar phenomenon exists in R4 when the attack percentage is low. In R3, malicious nodes always have much lower recommendation trust than good nodes. Thus, they can be easily detected as long as the threshold in the malicious node detection algorithm is properly chosen. The similar phenomenon exists in R4 when the attack percentage is high. Based on the above discussion, if the attackers do not want to help the network by providing honest recommendations and do not want to be detected easily, the best strategy for them providing recommendation is R2.

When the attackers launch the conflicting-behavior attack and use recommendation strategy R2, the MDP performance of the two detection methods (using or not using recommendation trust) is shown in Figure 6. The data is for the simulation time 1500. It is observed that using recommendation trust in malicious node detection yields a lower detection rate. This is because the good nodes' recommendation trust is deteriorated. This result is opposite to the result when the bad-mouthing attack is launched.

In practice, when conflicting-behavior attack is suspected, one should not use recommendation trust in the detection algorithm. This is a simple defense to the conflicting behavior attack. When it is not clear what types of attacks are launched, using recommendation trust in the malicious node detection is still a good idea because of its obvious advantages in defeating the bad-mouthing attack.

VI. Conclusion

This paper describes trust evaluation mechanism in distributed networks such as MANET and sensor networks, with a focus on protecting such systems against malicious attacks. In particular, the advantage of integrating trust in distributed networks is demonstrated through a synthesis on the roles of trust and simulations. Three attacks against trust evaluation are investigated in depth. The main results are summarized as follows. For the bad mouthing attack, the most effective defense is to incorporate recommendation trust in the malicious node detection algorithm. To defeat the on-off attack, the adaptive forgetting scheme is better than using fixed forgetting factors. In the conflicting-behavior attack, when the attackers do not provide recommendations to anyone, this attack is most effective. Under the conflicting-behavior attack, using recommendation trust in malicious node detection can reduce the detection rate. The joint effects of various attacks can be an interesting future research topic.

References

[1] A. Perrig, D. Clark, and S. Bellovin (Editors), "Secure next generation internet," NSF Workshop Report, July 2005.

[2] A. Josang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision Support Systems*, vol. 43, no. 2, pp. 618–644, 2005.

[3] Y. Wang and J. Vassileva, "A review on trust and reputation for web service selection," in *Proceeding of the first Int. Workshop on Trust and Reputation Magnement in Massively Distributed Computing Systems (TRAM'07)*, Toronto, Canada, June 2007.

[4] Pietro Michiardi and Refik Molva, "Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks," in *Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security*, 2002, pp. 107–121.

[5] Y. Sun, W. Yu, Z. Han, and K. J. Ray Liu, "Information theoretic framework of trust modeling and evaluation for ad hoc networks," *IEEE JSAC Special Issue on Security in Wireless Ad Hoc Networks*, vol. 24, pp. 305– 317, February 2006.

[6] S. Buchegger and J. L. Boudec, "Coping with false accusations in misbehavior reputation systems for mobile ad-hoc networks," EPFL Technical Report IC/2003/31, EPFL-DI-ICA, 2003.

[7] S. Ganeriwal and M. B. Srivastava, "Reputation-based framework for high integrity sensor networks," in *Proceedings of ACM Security for Ad-hoc and Sensor Networks (SASN)*, Washington, D.C., USA, Oct. 2004.

[8] Y. Sun, Z. Han, W. Yu, and K. J. Ray Liu, "A trust evaluation framework in distributed networks: Vulnerability analysis and defense against attacks," in *Proceeding of the 27th Conference on Computer Communications (INFO-COM'06)*, Barcelona, Spain, April 2006.

[9] C. Dellarocas, "Mechanisms for coping with unfair ratings and discriminatory behavior in online reputation reporting systems," in *Proceddings of the twenty first international conference on Information systems (ICIS)*, Brisbane, Queensland, Australia, December 2000.

[10] M. Langheinrich, "When trust does not compute - the role of trust in ubiquitous computing," in *Proceedings of the 5th International Conference on Ubiquitous Computing(UBICOMP)*, Seattle, Washington, October 2003.

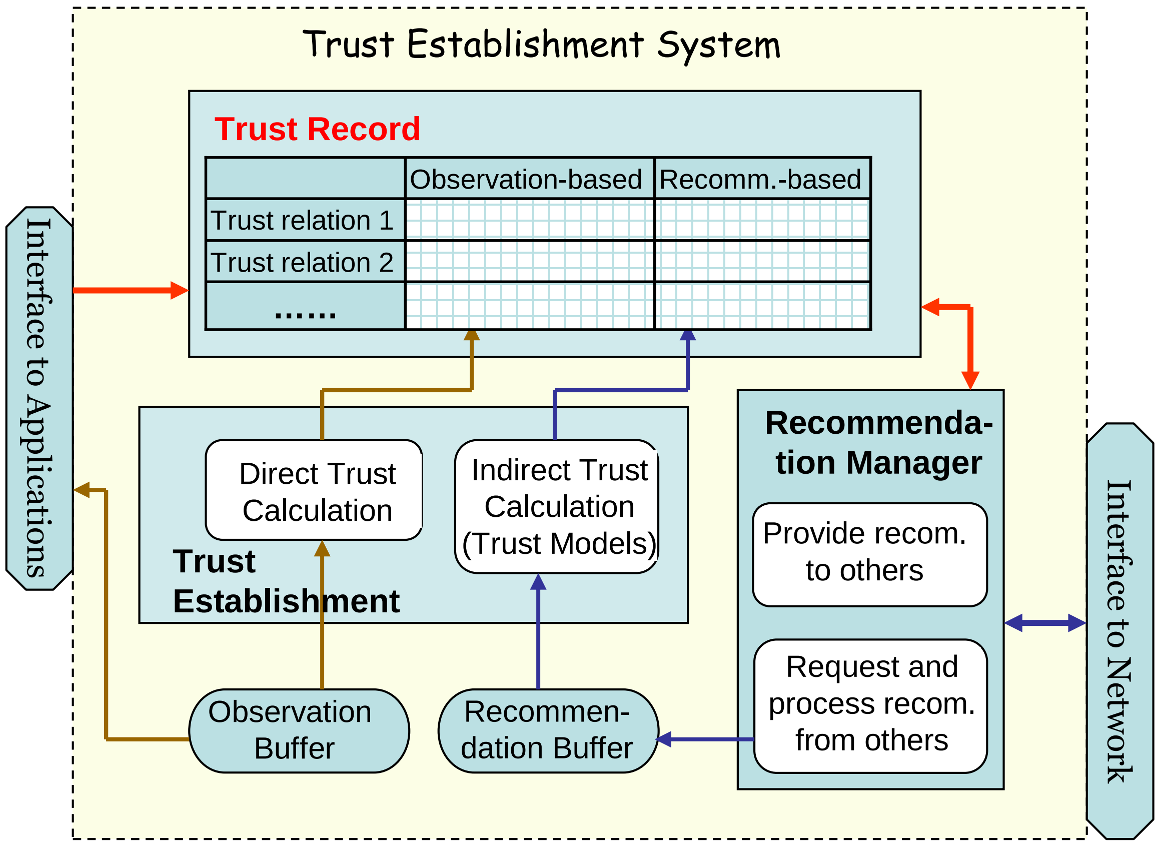

Fig. 1: Basic elements in trust establishment systems

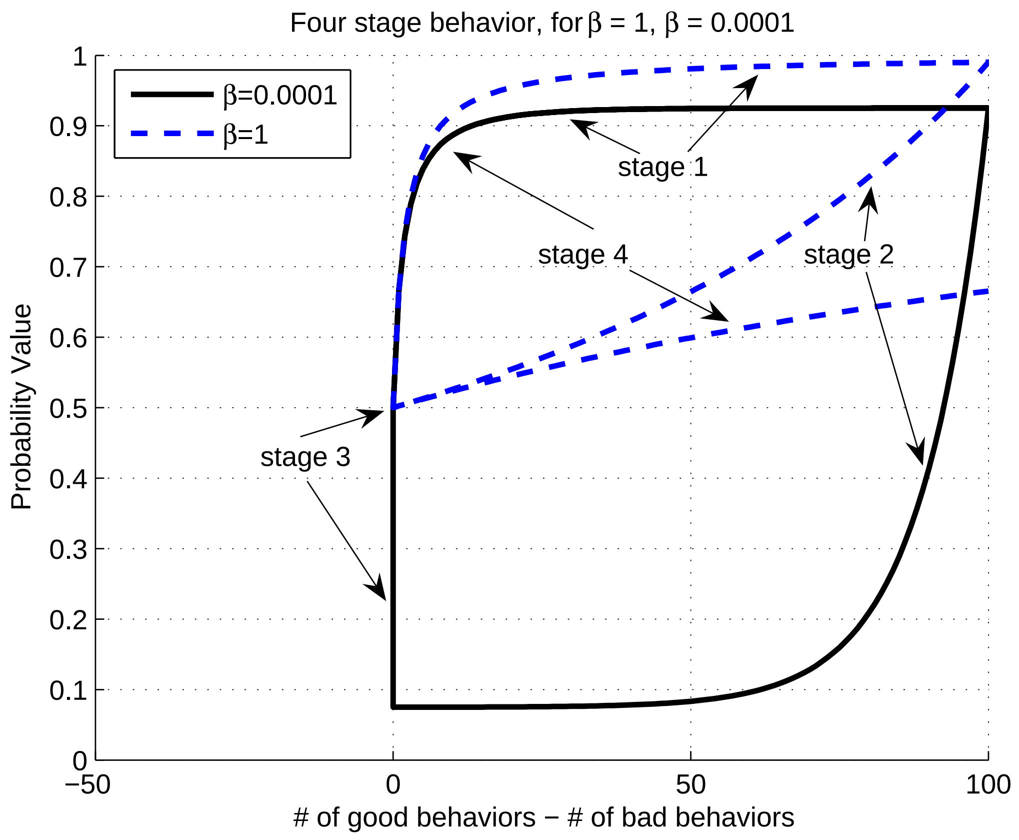

Fig. 2: Probability-based trust value under on-off attack with fixed forgetting factors

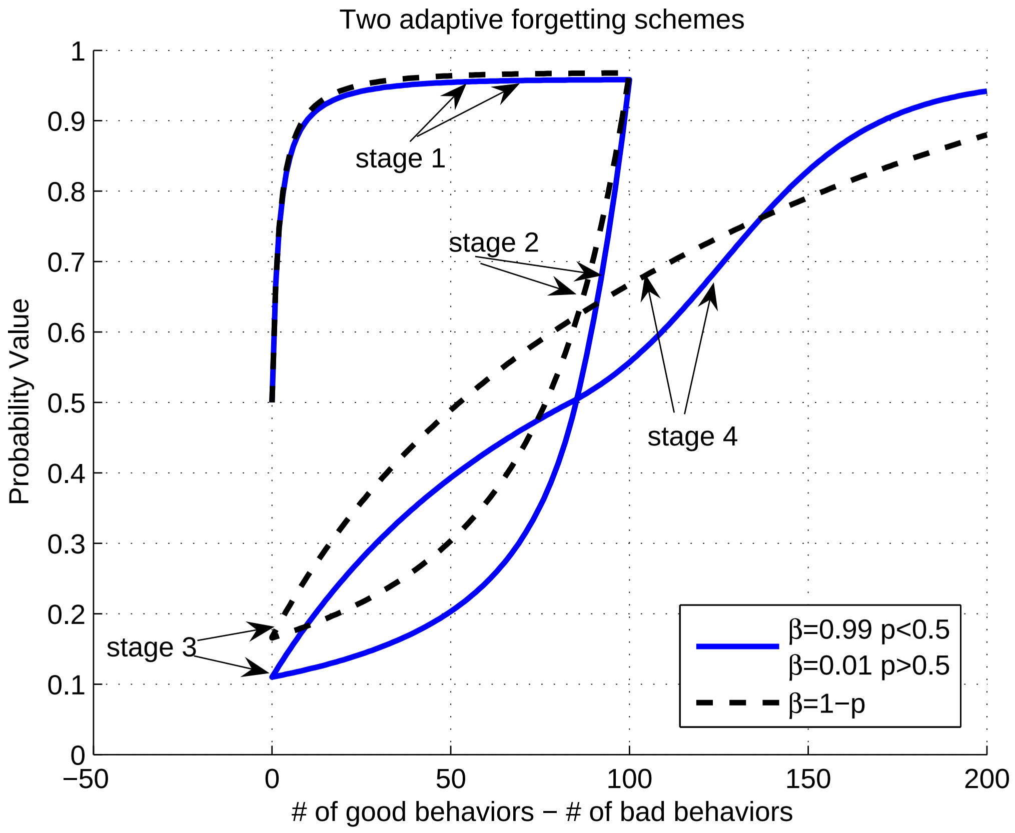

Fig. 3: Probability-based trust value under on-off attack with adaptive forgetting factors

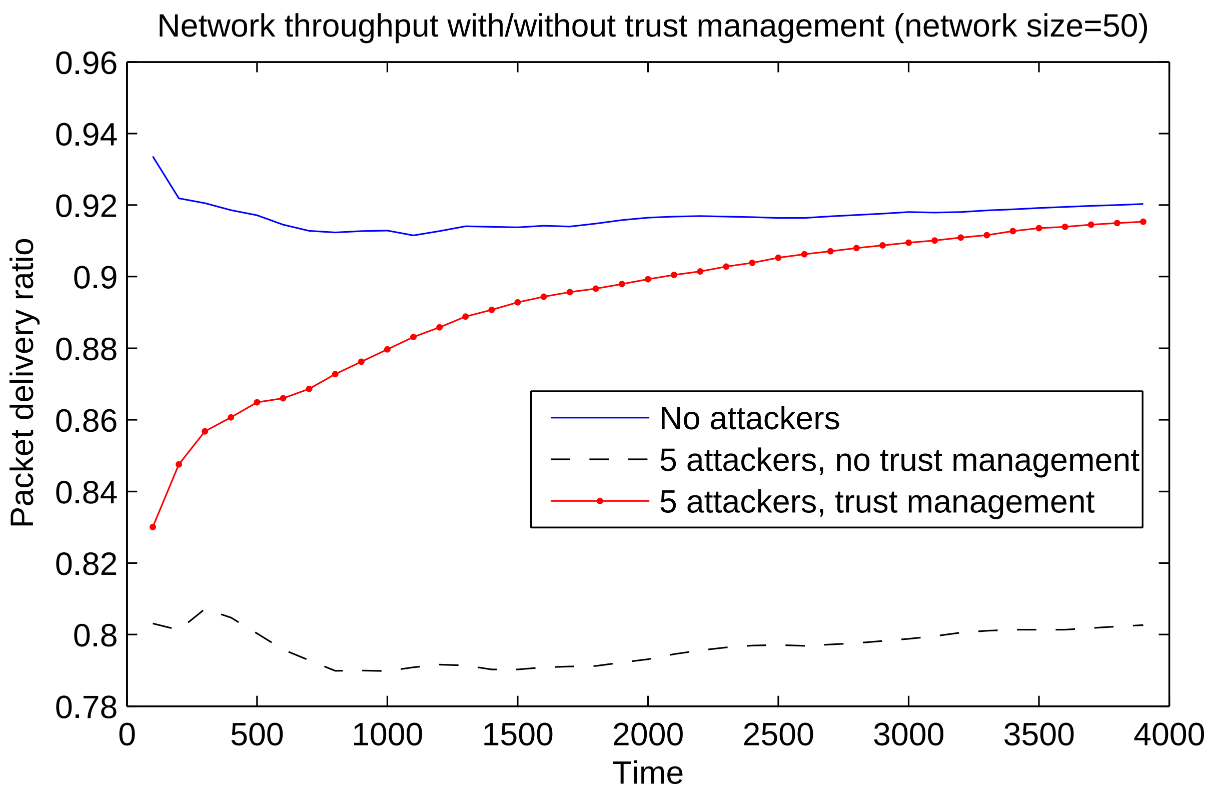

Fig. 4: Network throughput with and without trust management

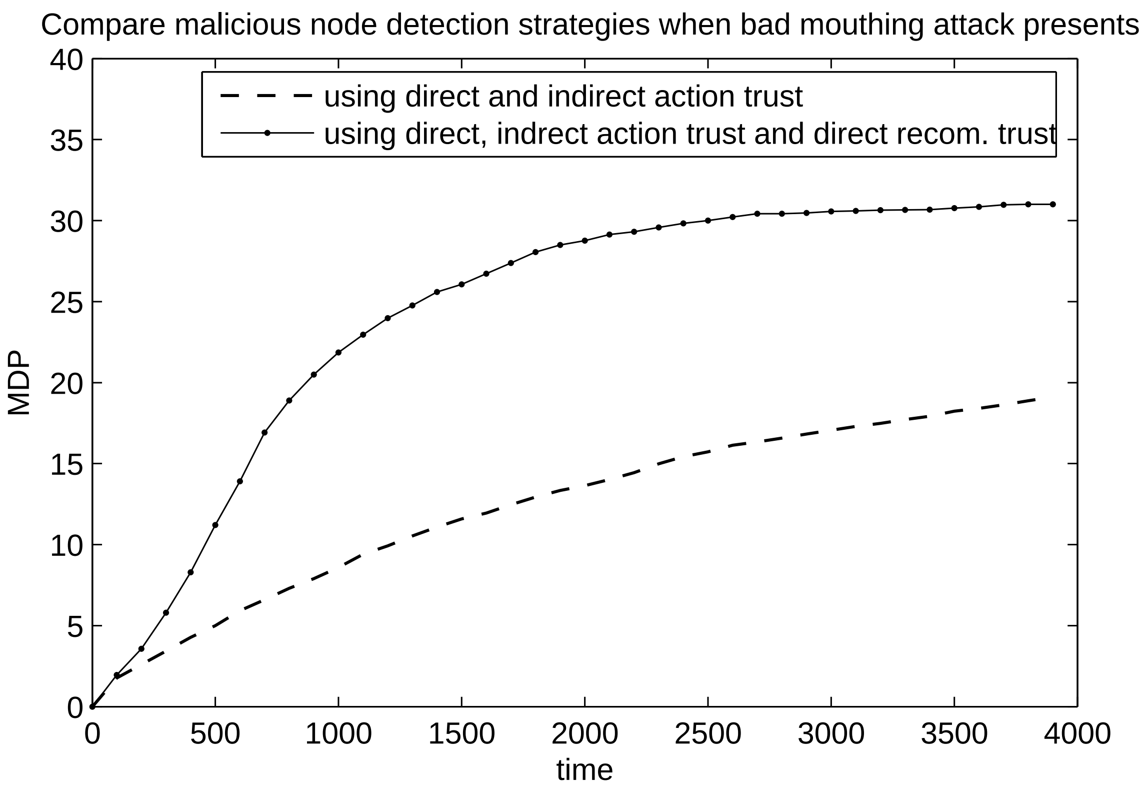

Fig. 5: Comparison between two malicious node detection methods under bad mouthing attack

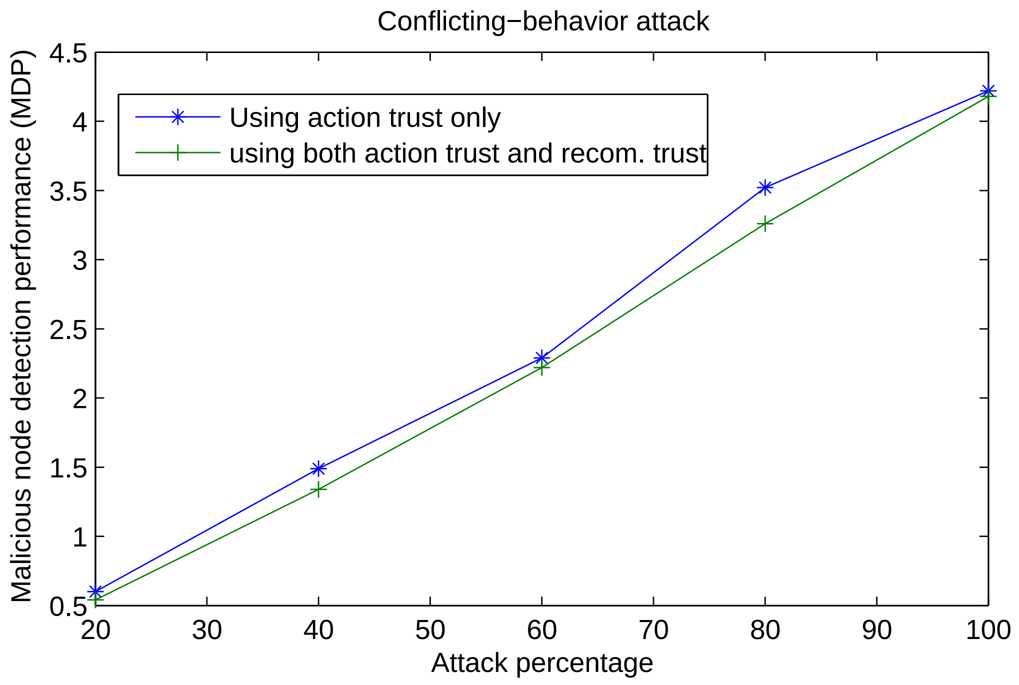

Fig. 6: Comparison between two malicious node detection methods under conflicting-behavior attack