Theodorakopoulos, G.; Baras, J.S.; , "On trust models and trust evaluation metrics for ad hoc networks," Selected Areas in Communications, IEEE Journal on , vol.24, no.2, pp. 318- 328, Feb. 2006
doi: 10.1109/JSAC.2005.861390
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1589111&isnumber=33490

On Trust Models and Trust Evaluation Metrics for Ad-Hoc Networks

George Theodorakopoulos, and John S. Baras, *Fellow, IEEE*

***Abstract*— Within the realm of network security, we interpret the concept of trust as a relation among entities that participate in various protocols. Trust relations are based on evidence created by the previous interactions of entities within a protocol. In this work, we are focusing on the evaluation of trust evidence in Ad Hoc Networks. Because of the dynamic nature of Ad Hoc Networks, trust evidence may be uncertain and incomplete. Also, no pre-established infrastructure can be assumed. The evaluation process is modeled as a path problem on a directed graph, where nodes represent entities, and edges represent trust relations. We give intuitive requirements and discuss design issues for any trust evaluation algorithm. Using the theory of semirings, we show how two nodes can establish an indirect trust relation without previous direct interaction. We show that our semiring framework is flexible enough to express other trust models, most notably PGP's Web of Trust. Our scheme is shown to be robust in the presence of attackers.**

***Index Terms*— trust evaluation, trust metric, semiring, trust model**

I. Introduction

THE notion of trust, in the realm of network security, will for our purposes correspond to a set of relations among entities that participate in a protocol [1]. These relations are based on the evidence generated by the previous interactions of entities within a protocol. In general, if the interactions have been faithful to the protocol, then trust will "accumulate" between these entities. Exactly how trust is computed depends on the particular protocol (application). The application determines the exact semantics of trust, and the entity determines how the trust relation will be used in the ensuing steps of the protocol. Trust influences decisions like access control, choice of public keys, etc. It could be useful as a complement to a Public Key Infrastructure (PKI), where an entity would accept or reject a public key according to the trustworthiness of the entities that vouch for it (i.e. have signed a certificate for it) – this is the idea behind PGP's Web of Trust [2]. It can also be used for routing decisions: Instead of the shortest path, we could be looking for the most trusted path between two nodes (this has been already proposed in P2P networks [3]).

Manuscript received May 30, 2005; revised August 15, 2005. Work prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U.S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. Research also supported by the U.S. Army Research Office under grant No DAAD19-01-1-0494. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. This work was presented in part at WiSE'04, Philadelphia, PA, October 2004

The authors are with the Electrical and Computer Engineering Department, and the Institute for Systems Research, University of Maryland, College Park, MD 20742 USA (email: gtheodor@isr.umd.edu; baras@isr.umd.edu)

For an illustration associated with Public Keys, suppose that entity A wants to determine the public key that entity B controls. In this case, the trust relation would be : "A does (or does not) believe that B's key is K_B". A and B have had no previous interactions, hence no trust relation, so A has to contact entities that have some evidence about B's key. Relevant pieces of evidence in this case are certificates binding B's key to B's identity. Also, the trustworthiness of the entities that issued these certificates should be taken into account.

In a regular PKI with a Trusted Third Party (TTP), A would now contact the TTP for B's key. Since the TTP is trusted by everyone, A would believe that B's key is what the TTP provided, and that would be the end of the story. In this work, however, we do not assume the existence of any globally trusted entity: on the contrary, everything is up to the individual nodes of the network. They themselves sign certificates for each other's keys, and they themselves have to judge how much to trust these certificates and, essentially, their issuers. If A has had previous interactions with these issuing entities, then their public keys as well as their trustworthiness will be known to A, who will now decide whether to accept K_B as B's key or not. Otherwise, the same steps will have to be repeated to establish a trust relation with the issuing entities, recursively, until A can reach a decision, which could very well be that there is not (trustworthy) enough evidence to establish the relation. This is what the trust computation algorithm does (Sec III-D.4), but in a forward way: A first computes trust values for his one-hop neighbors, then two-hop, and so on until the destination is reached (or, in the general case, until A has computed a trust value for all other entities).

The specification of admissible types of evidence, the generation, distribution, discovery and evaluation of trust evidence are collectively called Trust Establishment. In this work, we are focusing on the evaluation process of trust evidence in Ad-Hoc Networks, i.e. we are focusing on the trust metric itself. In particular, we are not dealing with the collecting of evidence from the network, and the accompanying communication and signaling overhead. This issue is important, and obviously needs to be addressed in a complete system.

We will be using the terms "trust evaluation", "trust computation", and "trust inference" interchangeably. The evaluation process is formulated as a path problem on a weighted, directed graph. In this graph, nodes represent users, and edges represent direct trust relations, weighted by the amount of trust that the first user places on the second. Each user has direct relations only towards the users he has interacted with, so all interactions are local (in the trust graph). The aim is to establish an indirect relation between two users that have not previously interacted; this is achieved by using the direct trust relations that intermediate nodes have with each other. Hence, we assume that trust is transitive, but in a way that takes into account edge weights, too.

Ad Hoc networks are envisioned to have dynamic, sometimes rapidly-changing, random, multihop topologies which are composed of bandwidth-constrained wireless links. The nodes themselves form the network routing infrastructure in an ad hoc fashion [4]. Based on these characteristics, we are imposing the following three main constraints on our scheme:

First, there is no preestablished infrastructure. The computation process cannot rely on, e.g., a Trusted Third Party. There is no centralized Public Key Infrastructure, Certification Authorities, or Registration Authorities with elevated privileges.

Second, evidence is uncertain and incomplete. Uncertain, because it is generated by the users on the fly, without lengthy processes. Incomplete, because in the presence of adversaries we cannot assume that all friendly nodes will be reachable: the malicious users may have rendered a small or big part of the network unreachable. Despite the above, we require that the results are as accurate as possible, yet robust in the presence of attackers. It is desirable to, for instance, identify all allied nodes, but it is even more desirable that no adversary is misidentified as good.

Third, the trust metric cannot impose unrealistic communication/computation requirements. Although we are not modeling or measuring the communication in any detail, we are looking for a scheme that would lend itself to an efficient implementation. In other words, it should be as light as possible since it is a complement to the real operation of the network.

We use a general framework for path problems on graphs as a mathematical basis for our proposed scheme, and also give intuitive requirements that any trust evaluation algorithm should have under that framework. The formalism of semirings highlights that our algorithm is a member of a larger family of well studied algorithms, collectively described under the term Factor Graphs [5], or Generalized Distributive Law [6]. Such algorithms include Dijkstra's shortest path algorithm, the Viterbi decoding algorithm, the Kalman filter, etc. So, analytical results about these algorithms can be directly used. Moreover, because of a particular property of semirings (distributivity, see Sec. III-D), we can do in-network processing of trust evidence, thus reducing the amount of data that needs to reach the source. In other words, local computation and message exchange is possible, which is a feature of all algorithms under the Factor Graph umbrella. We argue that it is especially useful in the context of ad-hoc networks.

This work is organized in five sections. After this Introduction, the second section describes and comments on trust design issues that frequently appear in related work. The third section explains our approach, proposes a flexible mathematical modeling framework for trust computation, and describes intuitive properties that any scheme under this framework should have. In the fourth section, our proposed scheme is used for actual computation scenarios, and the results are discussed. The fifth section concludes the paper and suggests future directions for improvement.

II. Taxonomy of Related Work

In this section we are examining important issues that should be considered by designers of trust metrics. For more specific examples of related work, please see [7].

A. *System Model*

The most commonly used model is a labeled, directed graph. Nodes represent entities, and edges represent binary trust relations. These relations can be (for an edge $i \to j$): a public key certificate (issued by i for j's key), the likelihood that the corresponding public key certificate is valid, the trustworthiness of j as estimated by i, etc.

B. *Centralized vs decentralized trust*

By centralized trust we refer to the situation where a globally trusted party calculates trust values for every node in the system. All users of the system ask this trusted party to give them information about other users. The situation described has two important implications: First, every user depends on the trustworthiness of this single party, thus turning it into a single point of failure. Second, it is reasonable to assume that different users are expected to have different opinions about the same target; this fact is suppressed here.

The decentralized version of the trust problem corresponds to each user being the ”center of his own world”. That is, users are responsible for calculating their own trust values for any target they want. This ”bottom-up” approach is the one that has been most widely implemented and put into use, as a part of PGP [2] for public key certification.

Note that the distinction just mentioned refers to the semantics of trust. The actual algorithm used for the computation of trust is a separate issue: all data may be gathered at a single user, where the algorithm will be executed; or the computation may be done in a distributed fashion, throughout the network; or the algorithm may even be localized, in the sense that each node only interacts with his local neighborhood, without expecting any explicit cooperation from nodes further away.

C. *Proactive vs reactive computation*

This is an issue more closely related to the communication efficiency of the actual implementation. The same arguments as in routing algorithms apply: Proactive trust computation uses more bandwidth for maintaining the trust relationships accurate. So, the trust decision can be reached without delay. On the other hand, reactive methods calculate trust values only when explicitly needed. The choice depends largely on the specific circumstances of the application and the network. For example, if local trust values change much more often than a trust decision needs to be made, then a proactive computation is not favored: The bandwidth used to keep trust values up to date will be wasted, since most of the computed information will be obsolete before it is used.

D. Extensional vs intensional metrics

As mentioned in [8] one possible criterion to classify uncertainty methods is whether the uncertainty is dealt with *extensionally* or *intensionally*. In extensional systems, the uncertainty of a formula is computed as a function of the uncertainties of its subformulas. In intensional systems, uncertainty is attached to ”state of affairs” or ”possible worlds”. In other words, we can either aggregate partial results in intermediate nodes (in-network computation), or we can collect all data (opinions and trust topology) at the initiator of the trust query and compute a function that depends on all details of the whole graph.

For example, the scheme proposed by Jøsang [9] is intensional, whereas ours is extensional. As pointed out by Maurer, there seems to be a trade-off between computational efficiency and correctness. Extensional systems are more efficient, whereas intensional are more correct. For more on this, see the discussion on the distributivity property (Sec III-D).

E. Attack resistance (node/edge attacks)

Levien ([10]) suggested a criterion for measuring the resistance of a trust metric to attackers. First, he distinguished between two types of attacks: node attacks, and edge attacks. Node attacks amount to a certain node being impersonated. So, the attacker can issue any number of arbitrary opinions (public key certificates in Levien’s case) from the compromised node about any other node.

Edge attacks are more constrained: Only one false opinion can be created per each attack. In other words, an attack of this type is equivalent to inserting a false edge in the trust graph. Obviously, a node attack is the more powerful of the two, since it permits the insertion of an arbitrary number of false edges.

The attack resistance of a metric can be gauged by the number of node or edge attacks, or both, that are needed before the metric can be manipulated beyond some threshold. For instance, in [11] Reiter and Stubblebine show that a single misbehaving entity (a 1-node attack) can cause the metric proposed in [12] to return an arbitrary result.

Here an important clarification has to be made: there are trust graphs that are ”weaker” than others. When, for example, there exists only a single, long path between the source and the destination, then any decent metric is expected to give a low trust value. So, the attack resistance of a metric is normally judged by its performance in these ”weak” graphs. This line of thinking also hints at why intensional systems (group metrics) perform better than extensional: They take into account the whole graph, so they can identify graph ”weaknesses” more accurately.

F. Negative and positive evidence (certificate revocation)

It is desirable to include both positive and negative evidence in the trust model. The model is then more accurate and flexible. It corresponds better to real-life situations, where interactions between two parties can lead to either satisfaction or complaints. When a node is compromised (e.g. its private key is stolen) the public key certificates for this node should be revoked. So, revocation can be seen as a special case of negative trust evidence.

On the other hand, the introduction of negative evidence complicates the model. Specifically, an attacker can try to deface good nodes by issuing false negative evidence about them. If, as a countermeasure to that, issuing negative evidence is penalized, good nodes may refrain from reporting real malicious behavior for fear of being penalized.

G. What layer should trust be implemented in?

An important issue that is often glossed over is the layer at which the trust protocol will operate. That is, the services required by the protocol and the services it offers should be made clear, especially its relationship to other security components. As pointed out in [13], some secure routing protocols assume that security associations between protocol entities can be established with the use of a trust establishment algorithm, e.g. by discovering a public key certificate chain between two entities. However, in order to offer its services, the trust establishment algorithm may often assume that routing can be done in a secure way. This creates a circular dependency that should be broken if the system as a whole is to operate as expected.

III. Semiring-Based Trust Evaluation Metrics

A. System Model

We view the trust inference problem as a generalized shortest path problem on a weighted directed graph $G(V, E)$ (*trust graph*). The vertices of the graph are the users/entities in the network. A weighted edge from vertex i to vertex j corresponds to the *opinion* that entity i, also referred to as the *issuer*, has about entity j, also referred to as the *target*. The weight function is $l(i, j) : V \times V \longrightarrow S$, where S is the opinion space.

Each opinion consists of two numbers: the *trust* value, and the *confidence* value. The former corresponds to the issuer’s estimate of the target’s trustworthiness. For example, a high trust value may mean that the target is one of the good guys, or that the target is able to give high quality location information, or that a digital certificate issued for the target’s public key is believed to be correct. On the other hand, the confidence value corresponds to the accuracy of the trust value assignment. A high confidence value means that the target has passed a large number of tests that the issuer has set, or that the issuer has interacted with the target for a long time, and no evidence for malicious behavior has appeared. Since opinions with a high confidence value are more useful in making trust decisions, the confidence value is also referred to as the *quality* of the opinion. The space of opinions can be visualized as a rectangle (ZERO_TRUST, MAX_TRUST)×(ZERO_CONF, MAX_CONF) in the Cartesian plane (Figure 1, for $S = [0, 1] \times [0, 1]$).

Both the trust and the confidence values are assigned by the issuer, in accordance with his own criteria. This means that a node that tends to sign public key certificates without too much consideration will often give high trust and high confidence values. The opposite holds true for a strict entity.

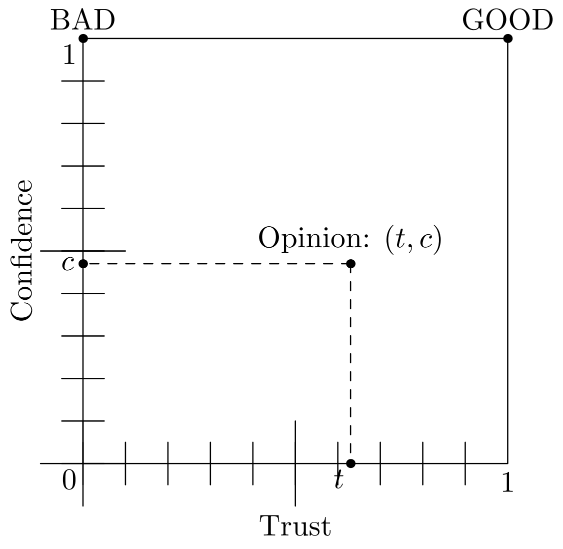

Fig. 1. Opinion space

When two such entities interact, it is important for the stricter entity to assign a low enough trust value to the less strict one. Otherwise, the less strict entity may lead the stricter one to undesirable trust decisions. This situation is easier to picture in the context of Certification Authorities and public key certification. In that context, a certification authority A will only give a high trust value to B, if B's policy for issuing certificates is at least as strict as A's and has the same durability characteristics [1].

Also, it is assumed that nodes assign their opinions based on local observations. For example, each node may be equipped with a mechanism that monitors neighbors for evidence of malicious behavior, as in [14]. Alternatively, two users may come in close contact and visually identify each other, or exchange public keys, as suggested in [15]. In any case, the input to the system is local: however, extant pieces of evidence based on, e.g., previous interactions with no longer neighboring nodes can also be taken into account for the final decision. This would come into play when two nodes that have met in the past need now to make a trust decision for each other. Of course, the confidence value for such evidence would diminish over time. One consequence of the locality of evidence gathering is that the trust graph initially overlaps with the physical topology graph: The nodes are obviously the same, and the edges are also the same if the trust weights are not taken into account. As nodes move, opinions for old neighbors are preserved, so the trust graph will have more edges than the topology graph. However, as time goes by, these old opinions fade away, and so do the corresponding edges.

In the framework described, two versions of the trust inference problem can be formalized. The first is finding the trust-confidence value that a source node A should assign to a destination node B, based on the intermediate nodes' trust-confidence values. Viewed as a generalized shortest path problem, it amounts to finding the generalized distance between nodes A and B. The second version is finding the most trusted path between nodes A and B. That is, find a sequence of nodes $\langle v_0 = A, v_1, \ldots, v_k = B\rangle : (v_i, v_{i+1}) \in E, 0 \leq i \leq k-1$ that has the highest aggregate trust value among all trust paths starting at A and ending at B. A high level view of the system is shown in Figure 2.

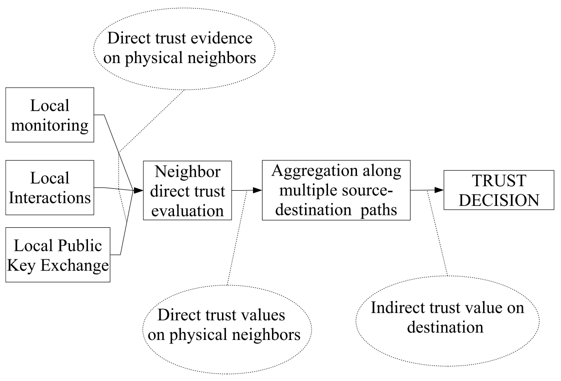
Direct trust evidence on physical neighbors
Local monitoring
Local Interactions
Local Public Key Exchange
Neighbor direct trust evaluation
Aggregation along multiple source-destination paths
TRUST DECISION
Direct trust values on physical neighbors
Indirect trust value on destination

Fig. 2. System operation

Both problems are important: finding a target's trust value is needed before deciding whether to grant him access to one's files, or whether to disclose sensitive information, or what kind of orders he is allowed to give (in a military scenario, for instance). With this approach, a node will be able to rely on other nodes' past experiences and not just his own, which might be insufficient. The second problem is more relevant when it comes to actually communicating with a target node. The target node being trustworthy is one thing, but finding a trusted path of nodes is needed, so that traffic is routed through them. Note that in the usual shortest path problem in a graph finding the distance between two nodes, simultaneously finds the actual shortest path. In the trust case, we will usually utilize multiple trust paths to compute the trust distance from the source to the destination, since that will increase the evidence on which the source bases its final estimate. The first problem is addressed with what we call ”Distance semiring” (Sec III-D.3), and the second with the ”Path semiring” (Sec III-D.2).

The core of our approach is the two operators used to combine opinions: One operator (denoted $\otimes$) combines opinions along a path, i.e. A's opinion for B is combined with B's opinion for C into one indirect opinion that A should have for C, based on B's recommendation. The other operator (denoted $\oplus$) combines opinions across paths, i.e. A's indirect opinion for X through path p_1 is combined with A's indirect opinion for X through path p_2 into one aggregate opinion. Then, these operators can be used in a general framework for solving path problems in graphs, provided they satisfy certain mathematical properties, i.e. form an algebraic structure called a semiring. More details on this general framework are in section III-B. Two existing trust computation algorithms (PGP [2] and EigenTrust [16]) are modeled as operations on two particular semirings. Note that our approach differs from PGP in that it allows the user to infer trust values for unknown users/keys. That is, not all trust values have to be directly assigned by the user making the computations. The operators are discussed in greater depth in section III-D.

B. Semirings

For a more complete survey of the issues briefly exposed here, see Rote [17], and also (for more applications in communications and other areas) Kschischang, Frey, Loeliger [5], and Aji, McEliece [6].

1) Definitions: A *semiring* is an algebraic structure $(S, \oplus, \otimes)$, where S is a set, and $\oplus, \otimes$ are binary operators with the following properties ($a, b, c \in S$):

- $\oplus$ is commutative, associative, with a neutral element $\textcircled{0} \in S$:

$$\begin{aligned} a \oplus b &= b \oplus a \\ (a \oplus b) \oplus c &= a \oplus (b \oplus c) \\ a \oplus \textcircled{0} &= a \end{aligned}$$

- $\otimes$ is associative, with a neutral element $\textcircled{1} \in S$, and $\textcircled{0}$ as an absorbing element:

$$\begin{aligned} (a \otimes b) \otimes c &= a \otimes (b \otimes c) \\ a \otimes \textcircled{1} = \textcircled{1} \otimes a &= a \\ a \otimes \textcircled{0} = \textcircled{0} \otimes a &= \textcircled{0} \end{aligned}$$

- $\otimes$ distributes over $\oplus$:

$$\begin{aligned} (a \oplus b) \otimes c &= (a \otimes c) \oplus (a \otimes c) \\ a \otimes (b \oplus c) &= (a \otimes b) \oplus (a \otimes c) \end{aligned}$$

A semiring $(S, \oplus, \otimes)$ with a partial order relation $\preceq$ that is monotone with respect to both operators is called an *ordered semiring* $(S, \oplus, \otimes, \preceq)$:

$$a \preceq b \text{ and } a' \preceq b' \Longrightarrow a \oplus a' \preceq b \oplus b' \text{ and } a \otimes a' \preceq b \otimes b'$$

An ordered semiring $(S, \oplus, \otimes, \preceq)$ is ordered by the *difference relation* if:

$$\forall a, b \in S : (a \preceq b \iff \exists z \in S : a \oplus z = b)$$

A semiring is called idempotent when the following holds:

$$\forall a \in S : a \oplus a = a$$

2) Semirings for path problems: One way we can see semirings in action is when computing a generalized shortest path weight in a weighted graph. In that case, $\otimes$ is the operator used to calculate the weight $w(p)$ of a path $p = (v_0, v_1, \ldots, v_k)$ based on the weights of the path's edges:

$$w(p) = w(v_0, v_1) \otimes w(v_1, v_2) \otimes \cdots \otimes w(v_{k-1}, v_k)$$

The $\oplus$ operator is used to compute the shortest path weight d_{ij} as a function of all paths from the source i to the destination j:

$$d_{ij} = \bigoplus_{\substack{p \text{ is a path} \\ \text{from } i \text{ to } j}} w(p)$$

Suppose we want to compute the delay of the shortest path from i to j in a network. We model the network as a weighted graph, where edge weights correspond to link transmission delays. Since link delays are non-negative, the set S is going to be $\Re_+ \cup \{\infty\}$. The total delay of a path is equal to the *sum* of all link delays (edge weights) along the path. So, the $\otimes$ operator is $+$, and $\textcircled{1}$ (neutral element for $\otimes$) is 0: if we add a zero-delay link in a path, the total delay does not change. This is regular addition. Now, we have all the path delays from i to j, and we want to somehow combine them so as to come up with the *shortest* path delay. The way to combine them is take the *minimum* among their values. So, the $\oplus$ operator is $\min$, and $\textcircled{0}$ (neutral element for $\oplus$) is ∞, since $\min(x, \infty) = x, \forall x \in \Re_+$: if we find an infinite delay path, the shortest path delay does not change. In summary, the semiring we should use for this situation is $(\Re_+ \cup \{\infty\}, \min, +)$.

Suppose now that we are given all link capacities instead of delays. We want to compute the highest possible rate of traffic from i to j along any single path (i.e. all paths are candidates, but we have to pick one). Link capacities, like link delays, are non-negative so S is again $\Re_+ \cup \{\infty\}$. The highest possible traffic rate along a path (the path capacity) is the *minimum* among all links along the path (bottleneck capacity). So, the $\otimes$ operator is $\min$, and $\textcircled{1}$ is ∞. Now, we have the capacities of all paths from i to j, and we want to find the *largest* among them. So, the $\oplus$ operator is $\max$, and $\textcircled{0}$ is 0: if we find a 0 capacity path, the maximum path capacity does not change. The semiring is now $(\Re_+ \cup \{\infty\}, \max, \min)$.

Note that the $\oplus$ operator may pick a single path weight (as is case above with $\max$ and $\min$) or it may explicitly combine information from all paths (addition or multiplication).

3) Semirings for systems of linear equations: An equivalent way to describe the previous shortest path problem is by way of a system of equations that the shortest path weights and the edge weights should satisfy. If a_{ij} is the weight of the edge (i, j), with $\textcircled{0}$ being the weight of non-existent edges, and x_{ij} is the shortest path weight from i to j, then the following equation has to hold (assume there exist n nodes):

$$x_{ij} = \bigoplus_{k=1}^{n} (a_{ik} \otimes x_{kj})$$

For example, when edge weights are transmission delays, this equation becomes:

$$x_{ij} = \min_{1 \le k \le n} (a_{ik} + x_{kj})$$

Note, also, that if $\oplus$ and $\otimes$ are the usual addition and multiplication, respectively, then the first of the above equations becomes exactly matrix multiplication.

$$x_{ij} = \sum_{k=1}^{n} a_{ik} x_{kj} \quad \Leftrightarrow \quad X = AX$$

where $X = [x_{ij}]_{n \times n}, A = [a_{ij}]_{n \times n}$

We will use this fact in a later section to model an existing trust computation algorithm.

C. Semirings as a model for trust computations

In order to show the modeling power of this framework, we now model PGP's web of trust computations [2] as a semiring. Remember that PGP computes the validity of an alleged key-to-user binding, as seen from the point of view of a particular user, henceforth called the source. The input to the computation algorithm consists of three things: The source node, the graph of certificates issued by users for each other, and the trust values for each user as assigned by the source. Note that the validity of all key-to-user bindings has to be verified, since only certificates signed by valid keys are taken into account, and any certificate may influence the validity of a key-to-user binding.

The validity of the key-to-user binding for user i will be deduced from the vector $d_i \in \mathbb{N}^k$, where k is the number of different trust levels defined by PGP. It seems that k is 4 (”unknown”, ”untrusted”, ”marginally trusted”, ”fully trusted”), but some include a fifth level : ”ultimately trusted”. Our analysis is independent of the exact value of k. The vector d_i will hold the number of valid certificates for user i that have been signed by users of each trust level. For example, $d_i = (0, 1, 2, 3)$ means that one ”untrusted”, two ”marginally trusted”, and three ”fully trusted” users have issued certificates for user i’s public key. In addition, all six of these certificates are signed by valid keys, i.e. keys for which the key-to-user binding has been verified.

In order to verify the actual validity of the binding, we will use the function $\mathtt{val} : \mathbb{N}^k \to \mathbb{V}$, where $\mathbb{V}$ is the space of admissible results. For simplicity, we will be assuming that $\mathbb{V}$ ={”invalid”, ”valid”}, although values such as ”marginally valid” have also been proposed. The output of val for a specific input is determined by thresholds such as: ”A key-to-user binding is valid if at least two ”marginally trusted” users have issued a certificate for it”. These thresholds are incorporated in val and will be transparent to our analysis. Finally, for computation simplicity we will be assuming that $\mathbb{V} = \{0, 1\}$, where ”invalid”$= 0$, and ”valid”$= 1$.

The edge weights $w_{ij} \in \mathbb{N}^k, 1 \leq i, j \leq n$, where n is the number of users, correspond to the certificate from i about j’s alleged public key. A weight can only have one of $k + 1$ possible values. Either it consists only of 0s, or of exactly $k - 1$ 0s and one 1. An all-zero weight means that there is no certificate from i about j’s key. An 1 in the position that corresponds to trust level t means that the source has assigned trust level t to i, and i has issued a certificate for j.

The $\otimes$ operator is defined as follows ($a, b \in \mathbb{N}^k$):

$$a \otimes b = \mathtt{val}(a)b \in \mathbb{N}^k$$

The $\oplus$ operator is defined exactly as vector addition in $\mathbb{N}^k$.

Verification of the semiring properties: For $\otimes$, the absorbing element is $⓪ = (0, \ldots, 0) \in \mathbb{N}^k$, and the neutral element is $① = \{x \in \mathbb{N}^k : \mathtt{val}(x) = 1\}$. That is, all such vectors are mapped to ①; for our purposes, they are equivalent. It is trivial to prove that ⓪ is a neutral element for $\oplus$.

The $\otimes$ operator is associative:

$$a \otimes (b \otimes c) = a \otimes (\mathtt{val}(b)c) = \mathtt{val}(a)\mathtt{val}(b)c$$

$$(a \otimes b) \otimes c = (\mathtt{val}(a)b) \otimes c = \mathtt{val}(\mathtt{val}(a)b)c$$

and these two are equal because val(⓪)=0.

The $\oplus$ operator is commutative and associative, because it is vector addition.

The $\otimes$ operator distributes over $\oplus$:

$$a \otimes (b \oplus c) = \mathtt{val}(a)(b + c)$$

$$(a \otimes b) \oplus (a \otimes c) = \mathtt{val}(a)b + \mathtt{val}(a)c$$

The computation algorithm below uses the above semiring to compute the validity or otherwise of all keys in the certificate graph G. The source node is s and the function w maps edges to edge weights.

```
PGP-SEMIRING-CALCULATION($G, w, s$)
 1  for $i \leftarrow 1$ to $|V|$
 2      do $d[i] \leftarrow ⓪$
 3  $d[s] \leftarrow ①$
 4  $S \leftarrow \{s\}$
 5  while $S \neq \emptyset$
 6      do $u \leftarrow$ DEQUEUE($S$)
 7          for each $v \in Neighbors[u]$ with $\mathtt{val}(d[v]) = 0$
 8              do
 9                  $d[v] \leftarrow d[v] \oplus (d[u] \otimes w(u, v))$
10                  if $\mathtt{val}(d[v]) = 1$
11                      then ENQUEUE($S, v$)
```

The computation starts at the source s, and progressively computes the validity of all keys reachable from s in the certificate graph. The queue S contains all valid keys for which the outgoing edges (certificates signed with these keys) have not been examined yet. When a key is extracted from S, its certificates to other keys are examined, and their d-vectors are updated. Only certificates to so-far-invalid keys are examined, since adding a certificate to the d-vector of a key already shown to be valid is redundant. If a so-far-invalid key obtains enough certificates to become valid, it is added to the queue for future examination. Each key is enqueued at most once (when it becomes valid), and all keys in the queue are eventually dequeued. Ergo, the algorithm terminates. After termination, all valid keys have been discovered.

Note that if s is only interested in the validity of a particular key-to-user binding, then the algorithm can stop earlier: as soon as its validity is determined, or after all certificates for that key have been examined.

We can also model the EigenTrust algorithm [16] as a semiring. Using the system of linear equations interpretation of a semiring, the EigenTrust algorithm solves the following matrix equation for T:

$$T = CT \quad \Leftrightarrow \quad t_{ij} = \sum_{k=1}^{n} c_{ik} t_{kj}$$

where the semiring operators are the usual addition and multiplication.

D. Trust Semirings

1) Trust Interpretation of Semiring Properties: Based on intuitive concepts about trust establishment, we can expect the binary operators to have certain properties in addition to those required by the semiring structure.

Since an opinion should deteriorate along a path, we require the following for the $\otimes$ operator ($a, b \in S$):

$$a \otimes b \preceq a, b$$

where $\preceq$ is the difference relation defined in Section III-B. Note that the total opinion along a path is ”limited” by the source’s opinion for the first node in the path.

Regarding aggregation across paths with the $\oplus$ operator, we generally expect that opinion quality will improve, since we have multiple opinions. If the opinions disagree, the more confident one will weigh heavier. In a fashion similar to the $\otimes$ operator, we require that the $\oplus$ operator satisfies $(a, b \in S)$:

$$a \oplus b \succeq a, b$$

The $⓪$ element (neutral element for $\oplus$, absorbing for $\otimes$) corresponds to the opinion ”I don’t know” (*not* the most negative opinion). This corresponds to non-existent trust relations between nodes. The rationale is that if a $⓪$ is encountered along a path, then the whole path ”through” this opinion should have weight equal to $⓪$. Also, such opinions should be ignored in $\oplus$-sums.

The element $①$ (neutral element for $\otimes$) is the ”best” opinion that can be assigned to a node. This can also be seen as the opinion of a node about itself. If encountered along a path, $①$ effectively contracts the corresponding edge and identifies the nodes at its endpoints for the purposes of the concatenation.

The associativity property for both operators allows the incremental calculation of results: If one more opinion needs to be aggregated into the current ”total”, then it can be done in one step, without having to recall all opinions that were aggregated for the current total. The same goes for concatenation. Commutativity for aggregation makes irrelevant the order in which opinion are taken into account (i.e. which one is first, which one is second, etc.)

The distributivity property is potentially more double-edged, in the sense that its desirability has been disputed by Jøsang [9]. Briefly stated, the argument is that distributivity ignores opinion dependence when aggregating. To visualise the situation, consider the following two graphs (Figs. 3 and 4). If distributivity holds, then the source cannot distinguish between the two topologies, and, in fact, for the source all topologies are indistinguishable from the one in Fig. 4. So, even though the intermediate nodes are depending on the same node (the question-marked) for information on the destination, this fact is hidden from the source. This is called opinion dependence, and it is a problem because the real trust topology becomes equivalent to the topology that is perceived by the source (for an appropriate assignment of numerical values to the opinions). In other words, the question-marked node becomes a single point of failure, borrowing a term from the distributed systems terminology.

The situation at hand is an example of the extensional versus intensional approach (see Section II). Clearly, if complete information about the graph is available to the source, then the decision will be better. However, this means that all opinions in the trust graph will have to be sent to the source. On the other hand, if we make use of the distributivity property, in-network computation is possible: each node will only pass a single aggregate opinion to the upstream neighbor (the node on the way back to the source). This will save a significant amount of bandwidth, which is particularly suitable for resource-constrained ad-hoc wireless networks.

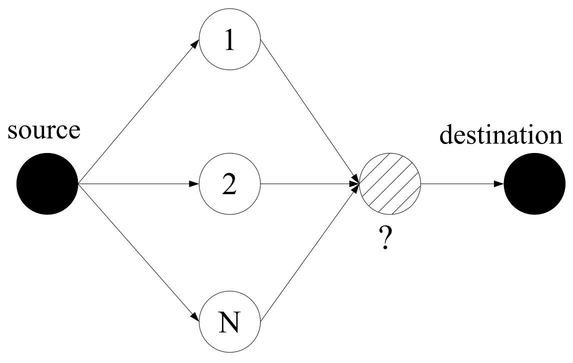

Fig. 3. Real topology

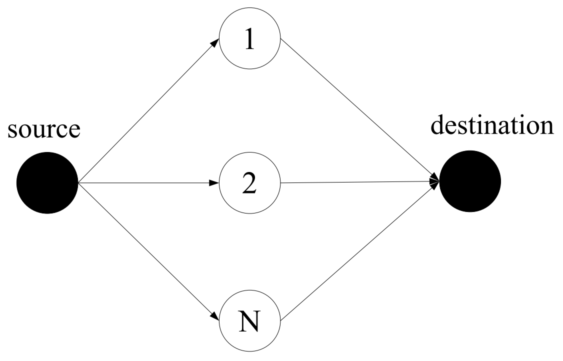

Fig. 4. Topology as perceived by the source

Finally, distributivity is defensible from the trust perspective, too. Namely, the shaded node is indeed a single point of failure, on which the source’s opinion depends, but it is also a node with multiple independent trust paths leading to it. So, if this node would turn out to be, say, malicious, it would mean that all other nodes *independently* vouching for it would be simultaneously wrong. That said, it is certainly better if there exist completely independent paths all the way from the source to the destination. But this is not always the case.

2) Path semiring: In our first semiring, the opinion space is $S = [0, 1] \times [0, 1]$ Our choice for the $\otimes$ and $\oplus$ operators is as follows (Figure 5):

$$(t_{ik}, c_{ik}) \otimes (t_{kj}, c_{kj}) = (t_{ik}t_{kj}, c_{ik}c_{kj}) \quad (1)$$

$$(t_{ij}^{p_1}, c_{ij}^{p_1}) \oplus (t_{ij}^{p_2}, c_{ij}^{p_2}) = \begin{cases} (t_{ij}^{p_1}, c_{ij}^{p_1}) & \text{if } c_{ij}^{p_1} > c_{ij}^{p_2} \\ (t_{ij}^{p_2}, c_{ij}^{p_2}) & \text{if } c_{ij}^{p_1} < c_{ij}^{p_2} \\ (t_{ij}^{*}, c_{ij}^{p_1}) & \text{if } c_{ij}^{p_1} = c_{ij}^{p_2} \end{cases} \quad (2)$$

where $(t_{ij}^{p_1}, c_{ij}^{p_1})$ is the opinion that i has formed about j along the path p_1, and $t_{ij}^{*} = \max(t_{ij}^{p_1}, t_{ij}^{p_2})$.

Since both the trust and the confidence values are in the $[0, 1]$ interval, they both decrease when aggregated along a path. When opinions are aggregated across paths, the one with the highest confidence prevails. If the two opinions have equal confidences but different trust values, we pick the one with

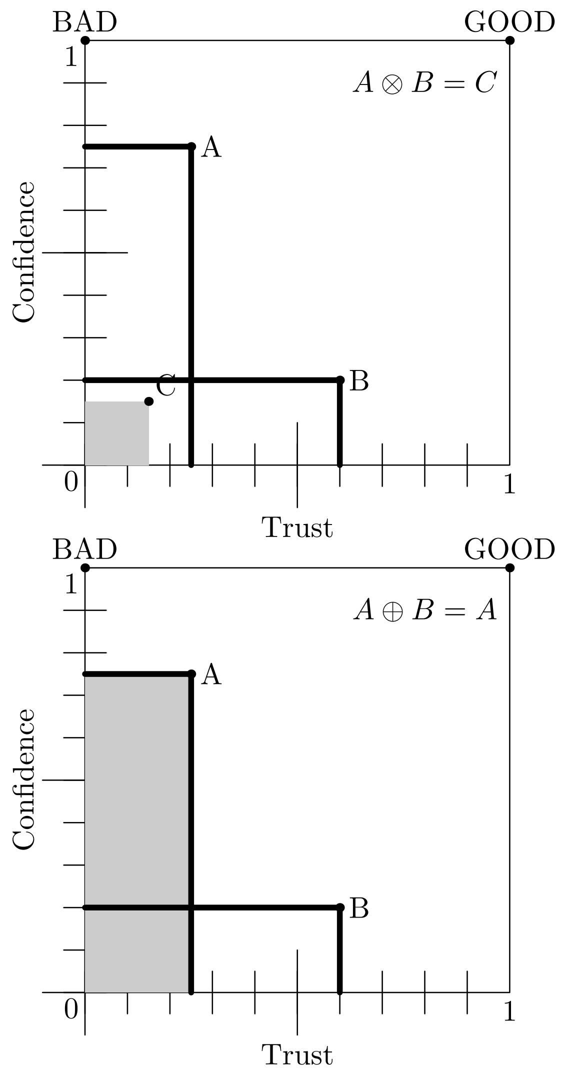

Fig. 5. ⊗ and ⊕ operators for the Path semiring

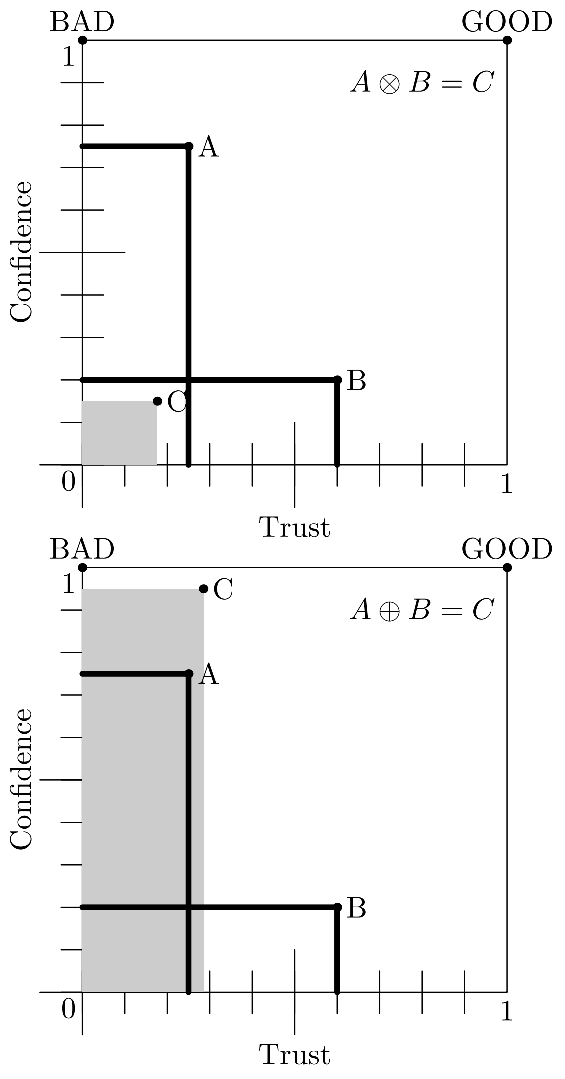

Fig. 6. ⊗ and ⊕ operators for the Distance semiring

the highest trust value. We could have also picked the lowest trust value; the choice depends on the desired semantics of the application.

This semiring essentially computes the trust distance along the most confident trust path to the destination. An important feature is that this distance is computed along a single path, since the $\oplus$ operator picks exactly one path. Other paths are ignored, so not all available information is being taken into account. One of the advantages is that if the trust value turns out to be high, then a trusted path to the destination has also been discovered. Also, fewer messages are exchanged for information gathering.

3) Distance semiring: Our second proposal, the distance semiring, is based on the *Expectation semiring* defined by Eisner in [18], and used for speech/language processing:

$$\begin{aligned}(a_1,b_1)\otimes(a_2,b_2) &= (a_1b_2+a_2b_1,b_1b_2)\\ (a_1,b_1)\oplus(a_2,b_2) &= (a_1+a_2,b_1+b_2)\end{aligned}$$

The opinion space is $S=[0,\infty]\times[0,1]$. Before using this semiring, the pair (trust, confidence)=(t,c) is mapped to the weight $(c/t,c)$. The motivation for this mapping becomes clear when we describe its effect on the results of the operators. The binary operators are then applied to this weight, and the result is mapped back to a (trust, confidence) pair. For simplicity, we only show the final result without the intermediate mappings.

$$\begin{aligned}(t_{ik},c_{ik})\otimes(t_{kj},c_{kj}) &\rightarrow \left(\frac{1}{\frac{1}{t_{ik}}+\frac{1}{t_{kj}}},c_{ik}c_{kj}\right)\\ (t^{p_1}_{ij},c^{p_1}_{ij})\oplus(t^{p_2}_{ij},c^{p_2}_{ij}) &\rightarrow \left(\frac{c^{p_1}_{ij}+c^{p_2}_{ij}}{\frac{c^{p_1}_{ij}}{t^{p_1}_{ij}}+\frac{c^{p_2}_{ij}}{t^{p_2}_{ij}}},c^{p_1}_{ij}+c^{p_2}_{ij}\right)\end{aligned}$$

So, when aggregating along a path, both the trust and the confidence decrease. The component trust values are combined like parallel resistors. Recall that two resistors in parallel offer lower resistance than either of them in isolation. Also, a zero trust value in either opinion will result in a zero trust value in the resulting opinion (absorbing element), while a trust value equal to infinity will cause the corresponding opinion to disappear from the result (neutral element). On the other hand, the component confidence values are between 0 and 1, and they are multiplied, so the resulting confidence value is smaller than both.

When aggregating across paths, the total trust value is the weighted harmonic average of the components, with weights according to their confidence values. So, the result is between the two component values, but closer to the more confident one. Note, also, the behavior caused by extreme (zero or infinity) trust values: A zero trust value dominates the result (unless its corresponding confidence is zero); a trust value equal to infinity results in an increase in the trust value given by the other opinion. In order for the resulting trust value to be the maximum possible, both opinions have to assign the maximum. So, in general, we can say that this operator is conservative. A zero confidence value (neutral element) causes the corresponding opinion to disappear from the result.

4) Computation algorithm: The algorithm below, due to Mohri [19], computes the $\oplus$-sum of all path weights from a designated node s to all other nodes in the trust graph $G = (V, E)$.

Revisiting the illustrative example described in the Introduction, remember that entity A wanted to judge the validity of entity B's public key based on certificates signed by other entities in the network. Using the trust computation algorithm, A will compute those entities' trustworthiness. If they are trustworthy enough, then A will believe their certificates are true, and will accept B's key. If not, A will not accept it. Of course, the above assumes that A already knows the public keys of the entities that issued the certificates for B. If that is not the case, then the whole process will be repeated for the public key of each one of the unknown issuers.

```
GENERIC-SINGLE-SOURCE-SHORTEST-DISTANCE($G, s$)
 1  for $i \leftarrow 1$ to $|V|$
 2      do $d[i] \leftarrow r[i] \leftarrow$ ⓪
 3  $d[s] \leftarrow r[s] \leftarrow$ ①
 4  $S \leftarrow \{s\}$
 5  while $S \neq \emptyset$
 6      do $q \leftarrow head(S)$
 7         DEQUEUE($S$)
 8         $r' \leftarrow r[q]$
 9         $r[q] \leftarrow$ ⓪
10         for each $v \in Neighbors[q]$
11             do if $d[v] \neq d[v] \oplus (r' \otimes w[(q, v)])$
12                  then $d[v] \leftarrow d[v] \oplus (r' \otimes w[(q, v)])$
13                       $r[v] \leftarrow r[v] \oplus (r' \otimes w[(q, v)])$
14                       if $v \notin S$
15                          then ENQUEUE($S, v$)
16  $d[s] \leftarrow$ ①
```

This is an extension to Dijkstra's algorithm [20][1]. S is a queue that contains the vertices to be examined next for their contribution to the shortest path weights. The vector $d[i], i \in V$ holds the current estimate of the shortest distance from s to i. The vector $r[i], i \in V$ holds the total weight added to $d[i]$ since the last time i was extracted from S. This is needed for non-idempotent semirings, such as the one proposed. Its computational complexity depends on the semiring used, and also on the actual topology of the network. As the reader can see in [19], the crucial parameter of the topology is the number of paths from the source to the other nodes. So, the more sparse the network, the more efficient the algorithm. But, in any case, the algorithm can be executed in a distributedfashion just like OSPF [21] with local data exchanges only.

[1]Note that Dijkstra's algorithm is essentially the base for the OSPF protocol [21], as pointed out by Reviewer 1. The extra r vectors take care of the contributions of additional paths to the trust value of a node.

Our computation algorithm is based on Mohri's, but with three adjustments which are needed when considering the problem from the perspective of trust. Lines 11-13 of the algorithm will be referred to as "node q votes for node v".

First of all, some nodes may be prevented from voting. Only if a node's trust value exceeds a predefined trust threshold, is the node allowed to vote. This is motivated from the common sense observation that only good nodes should participate in the computation, and bad nodes should be barred. Note that there is no restriction on the corresponding confidence. This will initially lead to bad nodes being allowed to vote, but after some point they will be excluded since good nodes will acquire evidence for their maliciousness.

Second, no node is allowed to vote for the source (s). Since it is s that initiates the computation, it does not make sense to compute s's opinion for itself.

Third, no cyclic paths are taken into account. If that were the case, we would be allowing a node to influence the opinion about itself, which is undesirable. Unfortunately, there is no clear way to discard any single edge-opinion of the cycle. So, the approach taken is to discard any edges that would form a cycle if accepted. As a result, the order in which the voters are chosen in line 6 is important. We argue that it makes sense to choose the node for which the confidence is highest.

These adjustments introduce characteristics from the Path semiring into the Distance semiring. For example, the node with the maximum confidence gets to vote first. Moreover, some paths are pruned which means that fewer messages are exchanged, thus saving bandwidth, but also some of the existing information is not taken into account.

IV. Evaluation and Experimental Results

In this section, we are describing the scenarios that were examined in the simulations. The results obtained are discussed, and explained in terms of the parameters and properties of the algorithms.

A. Good and Bad Nodes

We assume that some nodes are Good, and some are Bad. Good nodes adjust their direct opinions (opinions for their neighbors) according to some predefined rules (explained in Section IV-B). Bad nodes, however, always have the best opinion $(1, 1)$ for their neighboring Bad nodes, and the worst opinion $(0, 1)$ for their neighboring Good nodes.

We expect that the opinions of a Good node for all other nodes would evolve as in Figure 7. That is, all Good and all Bad nodes will be identified as Good and Bad, respectively.

B. Simulation details

When the network is "born", the nodes are partitioned into Good and Bad. We pick a Good node, which will be computing

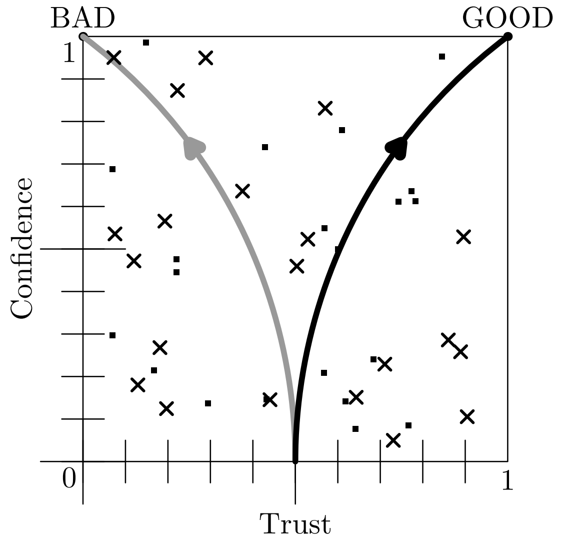

Fig. 7. Opinion convergence. Opinions for good nodes are drawn as crosses, opinions for bad nodes as squares.

indirect opinions to all other nodes. Initial direct opinions are all set to values randomly distributed in $(0.5, 0.1)$, i.e. medium trust and low confidence. The trust threshold, which decides which nodes are allowed to vote, is empirically set to 0.3. Time is discrete and is measured in rounds.

At each round, two things happen. First, the direct opinions of each node for his neighbors approach the correct opinion, which is $(0, 1)$ for the bad neighbors, and $(1, 1)$ for the good neighbors. The motivation is that the longer two nodes interact, the better they can estimate each other's trustworthiness. Second, the designated good node calculates his indirect opinions for all other nodes. These indirect opinions are the experimental results shown inFigures 8 and 9. Also, the confidence for some indirect opinions may be too low (within $\epsilon = 0.01$ of zero), so these nodes are not assigned any opinion.

The most important evaluation metric is whether the nodes are correctly classified as good and bad. In other words, we want the opinions for all bad nodes to be close to $(0, 1)$ and the opinions for all good nodes close to $(1, 1)$. Moreover, we want this to happen as soon as possible, i.e. before all direct opinions converge to the correct ones, since the users in the real network may be forced to make an early trust decision. Furthermore, a failsafe is desirable: If trust evidence is insufficient, we prefer not to make any decision about a node, rather than make a wrong one. Of course, we have to evaluate the robustness of each of the above mentioned metrics as the proportion of bad nodes increases.

The trust topology we are using is a Small World-type topology: The total number of nodes is 100, a few of which have a high degree, and all the rest have many fewer neighbors. The average degree is 8, but the highest is 19. The Small World topology for trust has also been used in [22]. A comparison with two other topologies, as well as the complete list of obtained results, is in [7].

C. Results and Discussion

We now present and discuss some representative results obtained through simulations. The percentage of bad nodes is increased from 10% to 50% to 90%. Figures 8 and 9 show the opinions of the source node (s) for every other node after the computations of rounds 30 and 70 for a 50% percentage of Bad nodes. The nodes originally designated as Good are pictured as crosses, whereas the Bad ones as squares. The aim is, first and foremost, for the Good nodes to be separated from the Bad ones. Also, the Good nodes should be as close as possible to the upper right corner (GOOD corner, corresponding to the $(1, 1)$ opinion), and the Bad nodes to the upper left corner (BAD corner, $(0, 1)$ opinion).

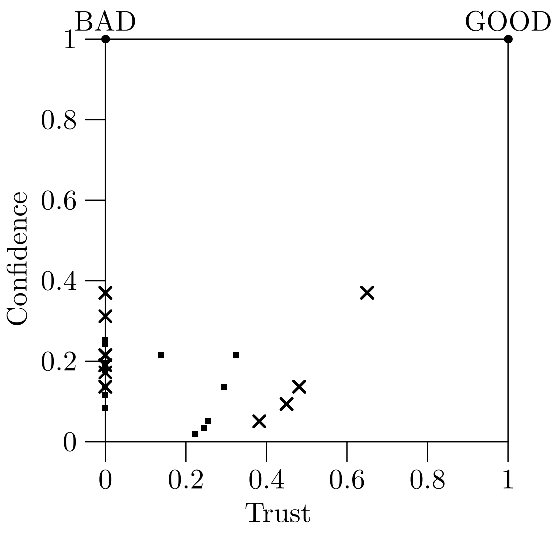

Fig. 8. 50% Bad nodes, round 30.

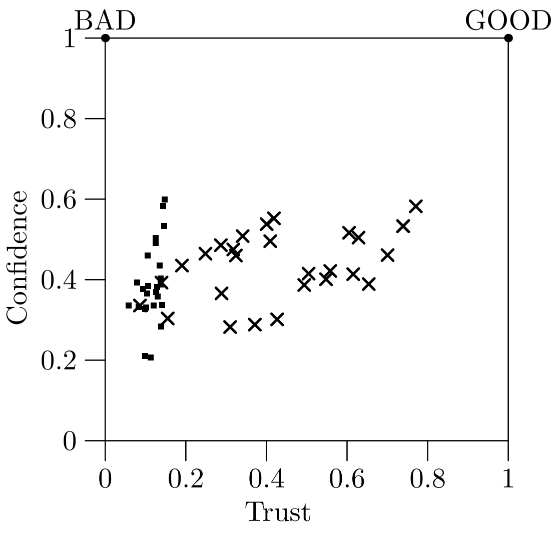

Fig. 9. 50% Bad nodes, round 70.

We were able to observe some general trends in the results obtained. First of all, in the early rounds Good and Bad nodes are intermixed: there is no clear separating line. Moreover, Bad nodes seem to be given better opinions than Good nodes, which is clearly undesirable. The explanation for this is based on two aspects of the scheme; namely, the trust threshold and the Bad nodes' way of assigning direct opinions. Initially, Bad nodes are allowed to vote, since the trust threshold (0.3) is lower than the initial default trust value (0.5), i.e. they have not been "discovered" yet. So, their $(0, 1)$ opinions for Good nodes are taken into account and the result is that Good nodes appear to be bad. Also, Bad nodes give $(1, 1)$ opinions to each other, hence reinforcing each other.

The situation in later rounds improves. The Good nodes move towards the upper right corner, the Bad ones towards the upper left. There is also a clear separating line between the two groups of nodes. For an actual implementation a practical guideline could be derived from the above observation, i.e. to be especially careful when making important trust decisions in early rounds. The trust computation may be based on too little raw evidence (direct opinions) to be relied upon. In all cases, however, the Good and Bad nodes are separated eventually (in the last rounds). This serves as a sanity check for the algorithm.

As the percentage of Bad nodes increases, we can see that the separation is still successful sooner or later, but the main observation is that the number of classified nodes is decreasing. Classified nodes are those for which the evidence was sufficient, i.e. the confidence of the source's opinion for them was more than $\epsilon = 0.01$. The following graphs show the number of nodes classified, for different percentages of Bad nodes, after every round of computation. The general effect of Bad nodes on the number of classified nodes is that, after they are discovered, they block the trust paths they are on since they are not allowed to vote. So, nodes that are further away from the source than these Bad nodes can be reached by fewer paths. They may even be completely isolated. In any case, the confidence in the source's opinion for them is decreased, so some of them cannot be classified.

In our Small World topology the average path length is short, since there are some highly connected nodes. However, it is exactly these highly connected nodes that degrade the performance of the computation when they are Bad. The reason is, again, that they block many paths and affect opinions for most nodes. If the majority of these highly connected nodes are Bad, few trust paths will be able to be established.

For the 50% and 90% Bad node cases, there is a noticeable drop in the number of classified nodes between rounds 30 and 40. This is so, because at this point the opinions for Bad nodes acquire trust values that are lower than the trust threshold, so they become ineligible to vote and block the paths they are on. This effect is more pronounced in the 90% case, but despite the Bad node preponderance, almost 40 nodes are classified. This happens because the source node is one of the highly connected nodes (19 neighbors, when the average degree is 8). So, all of the 19 neighbors, and some of the nodes that are two hops away are classified for a total of about 40 nodes. A practical guideline for the Small World topology would then be that highly connected nodes should be protected, better prepared to withstand attacks, or, in general, less vulnerable.

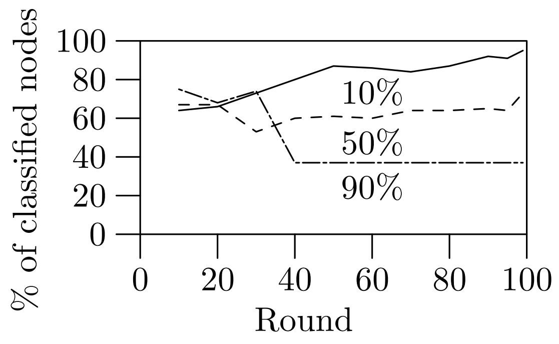

Fig. 10. Node classification, $10\% - 50\% - 90\%$ bad nodes

V. Conclusion and Future Work

We have presented a scheme for evaluating trust evidence in Ad-Hoc networks. Our scheme is entirely based on information originating at the users of the network. No centralized infrastructure is required, although the presence of one can certainly be utilized. Also, users need not have personal, direct experience with every other user in the network in order to compute an opinion about them. They can base their opinion on second-hand evidence provided by intermediate nodes, thus benefitting from other nodes' experiences. Of course, we are taking into account the fact that second-hand (or third, or fourth...) evidence is not as valuable as direct experience. In this sense, our approach extends PGP, since PGP only uses directly assigned trust values.

At each round of computation, the source node computes opinions for all nodes. This means that information acquired at a single round can be stored and subsequently used for many trust decisions. If there is not enough evidence to determine an opinion, then no opinion is formed. So, when malicious nodes are present in the network they cannot fool the system into accepting a malicious node as benevolent. A failsafe state exists that ensures graceful degradation as the number of adversaries increases.

In future work, we plan to implement more elaborate models for the attackers' behavior, and for the measures taken against nodes that are being assigned low trust values (i.e., detected to be bad). So, the attackers will be facing a tradeoff between the amount of damage they can inflict, and the possibility of being, for instance, isolated from the rest network. Suitable strategies will be developed for Good as well as Bad nodes.

Acknowledgment

The authors would like to thank the editors of the journal, and the anonymous reviewers for their cooperation and their comments that helped improve the presentation of the material.

References

[1] L. Eschenauer, V. D. Gligor, and J. Baras, "On trust establishment in mobile ad-hoc networks," in *10th International Security Protocols Workshop, Cambridge, UK, April 2002*, ser. Lecture Notes in Computer Science, B. Christianson, B. Crispo, J. A. Malcolm, and M. Roe, Eds., vol. 2845. Springer-Verlag, 2004, pp. 47–66.
[2] P. R. Zimmermann, *The Official PGP User's Guide*. MIT Press, 1995.
[3] S. Marti, P. Ganesan, and H. Garcia-Molina, "Sprout: P2P routing with social networks," Stanford University, Tech. Rep., January 2004.

[4] S. Corson and J. Macker, "Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations ,rfc 2501, IETF," January 1999.

[5] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, February 2001.

[6] S. M. Aji and R. J. McEliece, "The generalized distributive law." *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 325–343, 2000.

[7] G. Theodorakopoulos, "Distributed trust evaluation in ad-hoc networks," Master's thesis, University of Maryland, 2004. [Online]. Available: http://techreports.isr.umd.edu/ARCHIVE/

[8] U. Maurer, "Modelling a public-key infrastructure," in *Proc. 1996 European Symposium on Research in Computer Security (ESORICS' 96)*, ser. Lecture Notes in Computer Science, E. Bertino, Ed., vol. 1146. Springer-Verlag, 1996, pp. 325–350.

[9] A. Jøsang, "An algebra for assessing trust in certification chains," in *Proceedings of the Network and Distributed Systems Security (NDSS'99) Symposium*, 1999. [Online]. Available: citeseer.nj.nec.com/200003.html

[10] R. Levien and A. Aiken, "Attack-resistant trust metrics for public key certification," in *Proceedings of the 7th USENIX Security Symposium, San Antonio, Texas*, January 1998, pp. 229–242. [Online]. Available: http://www.usenix.org/publications/library/proceedings/sec98/levien.html

[11] M. K. Reiter and S. G. Stubblebine, "Authentication metric analysis and design," *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 2, pp. 138–158, May 1999.

[12] T. Beth, M. Borcherding, and B. Klein, "Valuation of trust in open networks," in *Proceedings of the European Symposium on Research in Computer Security, ESORICS94*, 1994, pp. 3–18.

[13] R. B. Bobba, L. Eschenauer, V. Gligor, and W. Arbaugh, "Bootstrapping security associations for routing in mobile ad-hoc networks," in *Proceedings of IEEE Globecom 2003*, San Francisco, CA, December 2003.

[14] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad-hoc networks," in *Proceedings of MOBICOM 2000*, 2000, pp. 255–265.

[15] S. Čapkun, J.-P. Hubaux, and L. Buttyán, "Mobility helps security in ad hoc networks," in *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC 2003)*, Annapolis, MD, June 2003.

[16] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The EigenTrust algorithm for reputation management in p2p networks," in *WWW2003*, May 2003.

[17] G. Rote, "Path problems in graphs," *Computing Supplementum*, vol. 7, pp. 155–189, 1990. [Online]. Available: http://www.inf.fu-berlin.de/ rote/Papers/postscript/Path+problems+in+graphs.ps

[18] J. Eisner, "Parameter estimation for probabilistic finite-state transducers," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, July 2002.

[19] M. Mohri, "Semiring frameworks and algorithms for shortest-distance problems," *J. Autom. Lang. Comb.*, vol. 7, no. 3, pp. 321–350, 2002.

[20] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[21] J. Moy, "OSPF version 2, RFC 2704," July 1991.

[22] J.-P. Hubaux, L. Buttyán, and S. Čapkun, "The quest for security in mobile ad hoc networks," in *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC 2001)*, 2001.

George Theodorakopoulos received the B.S. degree from the National Technical University of Athens, Greece, and the M.S. degree from the University of Maryland, College Park, MD, both in Electrical Engineering, in 2002 and 2004, respectively.

He is the co-recipient of the Best Paper award at the ACM Workshop on Wireless Security, October 2004. He is currently working towards the Ph.D. degree at the University of Maryland, focusing on ad-hoc and peer-to-peer network security issues.

John S. Baras received the B.S. in Electr. Eng. from National Technical University of Athens, Greece, 1970, and the M.S. and Ph.D. in Applied Math. from Harvard University 1971, 1973.

Professor Baras was the founding Director of the Institute for Systems Research (one of the first six NSF Engineering Research Centers) from 1985 to 1991. Since August 1973 he has been with the Electrical and Computer Engineering Department, and the Applied Mathematics Faculty, at the University of Maryland, College Park. In 1990 he was appointed to the Lockheed Martin Chair in Systems Engineering. Since 1991 Dr. Baras has been the Director of the Center for Hybrid and Satellite Communication Networks (a NASA Research Partnership Center).

Among his awards are: the 1980 Outstanding Paper Award, IEEE Control Systems Society; 1978, 1983, 1993 Alan Berman Research Publication Awards, NRL; 1991, 1994 Outstanding Invention of the Year Awards, University of Maryland; the Mancur Olson Research Achievement Award, University of Maryland; 2002, Best Paper Award 23rd Army Science Conference; 2004, Best Paper Award 2004 WiSe Conference. Dr. Baras holds three patents. He is a Fellow of the IEEE.

Professor Baras' research interests include: wireless networks and MANET, wireless network security and information assurance, integration of logic programming and nonlinear programming for trade-off analysis, multi-criteria optimization, non-cooperative and cooperative dynamic games, robust control of nonlinear systems and hybrid automata, mathematical and statistical physics algorithms for control and communication systems, distributed asynchronous control and communication systems, object oriented modeling of complex engineering systems, satellite and hybrid communication networks, network management, fast Internet services over hybrid wireless networks, stochastic systems, planning and optimization, intelligent control and learning, biologically inspired algorithms for signal processing and sensor networks.