# Computing of Trust in Ad-hoc Networks

Huafei Zhu, Feng Bao, Jianwei Liu*

Institute for Infocomm Research, A-star, Singapore
Email: {huafei, baofeng}@i2r.a-star.edu.sg
*BeiHang University, China, Email: liujianwei@buaa.edu.cn

**Abstract.** Although, the notion of trust has been considered as a primitive for establishing relationships among nodes in ad-hoc networks, syntax and metrics of trust are not well defined. This paper studies computing of trust in ad-hoc networks and makes the following three contributions. Firstly, the notion of trust is formalized in terms of predict functions and strategy functions. Namely, the notion of trust in this paper is defined as a predict function that can be further evaluated by a strategy function for a pre-described action; Secondly, structures of trust are formalized as a map between a path in the underlying network graph and the corresponding edge of its transitive closure graph; Thirdly, a generic model for computing of trust in the small world is proposed.

Keywords: Ad-hoc network, Transitive graph, Trust computing

## 1 Introduction

Ad-hoc networks formed by a set of dynamic nodes without relying on a preexisting infrastructure have been a very attractive field of academic and industrial research in recent years due to their potential applications and the proliferation of mobile devices. For example, a set of self-organized nodes are selected to accomplish a designated task say, collaboratively computing a multi-variable boolean function $f(x)$ on input $x$. In this setting, all nodes involved in the computation of $f(x)$ have to access a certain resource to obtain data in order to complete the task. As a result, a node should prove its membership to a self-organized set which is supposed to have access to the resource. If traditional public key infrastructures (PKI) are assumed, then the authentication of membership should be an easy task. However, it is difficult to deploy centralized certification authorities in ad-hoc networks due to the lack of central services.

Trust is considering a primitive for the establishment of relationship in ad-hoc networks. In our opinion, Alice trusts Bob means that Alice predicates that Bob will act on some action honestly in the future. It follows that the notion of trust should be defined as a predict (by $\mathcal{PT}$, we denote the function of a prediction). For example, a verification of a signature is a predict function; If an output of the predict function is 1, Alice's trust value evaluation strategy (by $\mathcal{SG}$, we denote a strategy for evaluating trust value) is then performed. The output value is called trust degree (or trust value) of Alice to Bob for the pre-specified action. Intuitively, the output of $\mathcal{SG}$ satisfies the following properties:

– one-wayness: for a fixed action $\mathcal{A}$ (by $\mathcal{A}$, we denote an action chosen from the pre-described action space which is denoted by $\mathcal{A}^*$), the concept of trust is one-way (or asymmetric) in the sense that $N_1$ trusts $N_2$'s action $\mathcal{A}$ does not imply that $N_2$ trusts $N_1$'s action $\mathcal{A}$.
– transitivity: the concept of trust maintains transitivity for a fixed action. That is, if $S$ trusts $N_1$'s action $\mathcal{A}$, and $N_1$ trusts $N_2$'s action $\mathcal{A}$, and $N_2$ trusts $T$'s action $\mathcal{A}$, then $S$ trusts $T$'s action $\mathcal{A}$. We stress that the action $\mathcal{A}$ specified by the source node $S$, intermediate nodes and the target node $T$ must be same, otherwise there is no reason to maintain the transitivity.

If we view individual participant in ad-hoc networks as a node of a delegation graph $G$, then a mapping between a delegation path from the source node $S$ to the target node $T$ in the graph $G$ and an edge in the transitive closure graph $G^*$ of $G$ can be established. We thus study the following fundamental research problems: how to evaluate trustworthiness of participants in an edge of $G$? how to compute trustworthiness of a recommendation path of $G$? Namely, how to evaluate the trustworthiness of edges in the transitive closure graph $G^*$ of $G$?

### 1.1 Previous Works

The pioneer work for computing of trust is due to Beth, Borcherding and Klein [2] and Yahalom, Klein and Beth [12]. In their seminal papers, models for computing of trust in distributed network are outlined. Although, a collection of genuine ideas were presented, there was no formal definition of trust presented in their papers. Following their seminal contributions, Zhu et al [14] distilled transitivity of trust by means of transitive graph and then applied their results for computing of trust in wireless networks (e.g., [15], [16] and [7]). Although, these are interesting applications of trust in the real world, the notion of trust is not well defined. For example, the term trust (and trust value/degree) defined in their previous papers does not cover the following important issues: the formalization of the notion of action (and action space), and the notion of trust; and the longer size of a recommendation path, the less trust value along a path; We stress that these issues are inherent properties of the notion of trust, and thus must be satisfied. As a result, any more satisfactory solution for computing of trust is certainly welcome.

### 1.2 This Work

The contributions of the paper are three-fold. In the first fold, the notion of trust is formalized in terms of predict functions and strategy functions. Namely, the notion of trust is defined as a predict that can be further evaluated by a strategy function for a pre-described action if a predict outputs 1; In the second fold, the structures of trust is formalized as a mapping between a path in a network graph $G$ and an edge of the transitive closure graph $G^*$ of $G$. In the third fold, a generic model for computing of trust in the small world phenomena is proposed.

The remainder work of this paper is organized as follows: In Section 2, syntax, structure of trust are introduced and formalized. In Section 3, a framework for computing of trust in ad-hoc works is proposed and analyzed. We propose an example for computing of trust in the small world phenomena in Section 4, and conclude our work in Section 5.

## 2 Trust: Syntax, Characteristics and Structures

### 2.1 Definition of Trust

Tons of definitions regarding trust have been presented in the literature. The commonly cited definition of trust is due to Golbeck[4]: Alice trusts Bob if she commits to an action $\mathcal{A}$ based on a belief that Bob's future actions will lead to a good outcome. We stress that Golbeck's definition does not capture the prediction of trust. That is, the notion of trust should be defined binary values: trust (a predict $\mathcal{PT}$ outputs 1) or distrust (a predict $\mathcal{PT}$ outputs 0). In case of trust (or distrust), we can talk about the degree of trust (or distrust). Since the notion of trust and the notion of distrust are complementary concepts, it is enough for us to define the concept of trust.

We also stress that an action $\mathcal{A}$ should be sampled by any probabilistic polynomial time (PPT) Turing machine on input of a system parameter $k$. That is, on input of a system parameter $k$, the PPT Turing machine will specify an action space ($\mathcal{A}^*$) such that on input of an index $i \in I$, an action $\mathcal{A}_i \in \mathcal{A}^*$ is selected.

Given an action $\mathcal{A} \in \mathcal{A}^*$, Alice runs a predict function $\mathcal{PT}$ which outputs 0 or 1. Once $\mathcal{PT}(\mathcal{A})$=1, Alice can preform her strategy function $\mathcal{SG}$ to obtain a trust value with the help of her auxiliary information *aux* (intuitively, the auxiliary information *aux* is a cumulative history record of Bob maintained by Alice herself).

Thus, to formalize the notion of trust, we first need to provide a formal definition of an action. Let $\mathcal{A}$ be a disjunction $c_1 \vee \cdots \vee c_m$ of clauses, where each clause $c_i$ is a conjunction $l_1 \wedge \cdots \wedge l_{t_i}$ of $t_i$ literals. Each literal $l_j$ is either a Boolean variable $X_i$ or its negation $\bar{X}_i$. Without loss of generality, we may assume that each variable occurs at once in any given clause.

**Definition 1.** *An action $\mathcal{A}$ is a disjunctive normal form over $k$ Boolean variables $X_1$, $\cdots$, $X_k$. The set of all actions is call action space which is denoted by $\mathcal{A}^*$.*

To define the trust value of an action, we need to make the following assumptions:

- the underlying network is an unknown fixed-identity graph $G$, where each node has a unique identity $N_i$ which cannot be forged. And each node knows the identities of its neighbors in $G$. Such an assumption is necessary since if a node forges its node id, then it is impossible for one to distinguish a forged id from a genuine id (as there is no public key infrastructure assumption involved in our model);

– a keyed-identity of node $N_i$ is of form $k_i$:=$(N_i, g(N_i))$ where $g(N_i)$ is a claimed public key of the node $N_i$.

**Definition 2.** *Let $k_A$ and $k_B$ (for convenience, we sometime will write $k_A$ simply as $A$) be two nodes in a graph $G$. An auxiliary information $aux^A(B) \in \{0,1\}^{poly(\lambda)}$ is a string that cumulatively records the state of $B$ by $A$.*

**Definition 3.** *An auxiliary information is called samplable if there is a deterministic polynomial time algorithm $\mathcal{I}$ such that on input $\lambda$, $k_A$ and $k_B$, it outputs $aux^A(B) \in \{0,1\}^{poly(\lambda)}$. By $\mathcal{I}^*$, we denote operators set $\mathcal{I}$.*

**Definition 4.** *On input $k_A$ and $k_B$, an action $\mathcal{A} \in \mathcal{A}^*$, and auxiliary information $aux^A(B)$, a deterministic predict function $\mathcal{PT}$ outputs a bit $b \in \{0,1\}$. Once $\mathcal{PT}$ outputs 1, $\mathcal{PT}$ then runs a trust evaluation strategy algorithm $\mathcal{SG}$ which outputs a positive value $\alpha \in \{0,1\}$. This value $\alpha$ is called a trust value of $k_A$ regarding the action $\mathcal{A}$ associated with $k_B$.*

### 2.2 Trust Structures

**Definition 5.** *A graph $G = (V, E)$ has a finite set $V$ of vertices and a finite set $E \subseteq V \times V$ of edges. The transitive closure $G^* = (V^*, E^*)$ of a graph $G = (V, E)$ is defined to have $V^* = V$ and to have an edge $(u, v)$ in $E^*$ if and only if there is a path from $u$ to $v$ in $G$.*

Based on the above assumptions, we can now define the structure of trust. For a given path $S \rightarrow N_1 \rightarrow \cdots \rightarrow N_k \rightarrow T$, we define the trust values of individual edges $S \rightarrow N_1$, $N_1 \rightarrow N_2$, $\cdots$ and $N_k \rightarrow T$. And we then compute the edge $S \rightarrow T$ in the transitive closure graph $G^*$. A a result, two types of trust structures can be defined: a direct trust and a recommended trust. Intuitively, a direct trust is an edge between two nodes in a graph $G$ while recommended trust is an edge defined in its corresponding transitive closure graph $G^*$. As a result, the notion of recommended trust can be viewed as a natural extension of the notion of the direct trust (if the number of intermediate nodes in a path is zero). Generally, for any path of length $k$ defined over $G$, a recommended trust $RT$ is defined of the following form: $\Pi_{i=1}^{k} DT_i$, where $DT_i$ is a direct trust of $N_i$ to $N_{i+1}$.

## 3 Computing of Trust

### 3.1 Computing of Direct Trust Values

Let $dtv^A(B)$ be a direct trust value assigned to $B$ by $A$; The range of $dtv^A(B)$ is $[0,1]$. If the trust value $dtv^A(B)$ is 0, it means that $A$ does not trust $B$ at all; if $dtv^A(B)$=1, it means that $A$ trusts $B$ completely; if $dtv^A(B)$ =$\alpha$, it means that $A$ trusts $B$ with degree $\alpha$, where $\alpha \in (0,1)$. Computing of direct trust value $dtv^A(B)$ can be performed as follows:

– Input $\Theta$:= $(k_A, k_B, aux^A(B), \mathcal{A})$, where $k_A$ (resp. $k_B$) is a key-identity of node $A$ (resp. $B$) and $aux^A(B)$ is auxiliary information regarding the node $B$ maintained by the node $A$;
– Computing $u \leftarrow \mathcal{PT}$ $(\Theta)$;
  - if $u$=0, then $v \leftarrow 0$;
  - if $u$=1, then $v \leftarrow \mathcal{SG}$ $(\Theta|u = 1)$
– Output $dtv^A(B) \leftarrow v$.

We stress that the above computation of the direct trust value captures two things. The first one is the notion of predict. This means that $A$ either trusts $B$ or distrusts $B$. The second one is the computation of direct trust value under the condition that $A$ trusts $B$.

## 3.2 Computing of Recommended Trust Values over Bounded-Disjoint-Paths

Suppose $p_1$, $\cdots$, $p_k$ be $k$ paths connected between $S$ and $T$. These paths are referred to as delegation paths. Let $N^i$ =$\{N_1^i, \cdots, N_{l_i}^i\}$ be a set of intermediate recommenders (not including $S$ and $T$) in the path $p_i$.

**Definition 6.** *Two paths from $S$ to $T$, say $S \rightarrow N_1^i \rightarrow \cdots \rightarrow N_{l_i}^i \rightarrow T$ and $S \rightarrow N_1^j \rightarrow \cdots \rightarrow N_{l_j}^j \rightarrow T$ are disjoint if $N_a^i \neq N_b^j$, for all $a$, $1 \leq a \leq l_i$ and all $b$, $1 \leq b \leq l_j$.*

**Definition 7.** *Suppose $p_1$, $\cdots$, $p_k$ be $k$ paths connected between $S$ and $T$, $p_1$, $\cdots$, $p_k$ are called mutually disjoint if paths are pair-wise disjoint.*

**Definition 8.** *A path $p$ is $\rho$-bounded if its length is at most $\rho$.*

Given a directed graph $G$ (we distinguish the node $S$ and the node $T$) and a path bound $\rho$, we are interested in finding the maximum set of mutually disjoint $\rho$-bounded paths from $S$ to $T$ $-$ an interesting research problem first introduced and formalized by Reiter and Stubblebine in [9], where the Bounded-Disjoint-Paths (BDP) problem is shown to be difficult if P $\neq$ NP. As a result, there is no polynomial approximation algorithm APP for BDP such that BDP($(G, \rho, S, T)$ -APP$(G, \rho, S, T) \leq C$ for a fixed constant $C$. This means that it is hard for one to find almost bounded disjoint paths in the graph $G$. Thus, for computing of trust in ad-hoc networks, we only consider a set of incomplete Bounded-Disjoint-Paths. As a result, to define the trust value for a set of bounded disjoint paths (say, $p_1$, $\cdots$, $p_k$), we need to consider the following two cases:

– Case 1: given a path $p$=$\{N_1, \cdots, N_l\}$ (excluding the source node $S$ and the target node $T$), how to define the trust value associated with the path $p$?
– Case 2: given a collection of paths(say, $p_1$, $\cdots$, $p_k$), how to define the trust value associated with the paths?

To compute trust value in Case 1, we first informally define the recommended trust value of $S$ to $T$ by the following formula for a given path $p$=: $\{S, N_1, \cdots, N_l, T\}$:

$$rtv^S(T,p) = dtv^S(N_1) \diamond dtv^{N_1}(N_2) \diamond \cdots \diamond dtv^{N_{l-1}}(N_l) \diamond dtv^{N_l}(T)$$

We stress that the direct trust value $dtv^{N_{i-1}}(N_i)$ has been defined in the last section. The remaining question is thus to define the exact meaning of the operator $\diamond$. Intuitively, a larger size $l$ implies that the smaller recommended trust values $rtv^S(T)$. Furthermore, if there is a faulty node that provides a fault recommendation, the resulting recommended trust value should be low. Consequently, the operator $\diamond$ can be defined in a simple way: $x \diamond y = min\{x, y\}$.

To compute the trust value in Case 2, we first introduce the following notations. By $min\{a_{i,1}, a_{i,2}, \cdots, a_{i,l_i}\}$, we denote the recommended trust value of $p_i$, i.e. $rtv^X(Y,p_i) = min\{a_{i,1},\ a_{i,2},\ \cdots,\ a_{i,l_i}\}$. By $max_{i=1}^t rtv^X(Y,p_i)$, we denote the recommended trust value computed from the path set $\{p_1,\ \cdots,\ p_t\}$. The recommended trust value computed from $\{p_1,\ \cdots,\ p_t\}$ is defined below

$$rtv^X(Y,p_1,\cdots,p_t) = max_{i=1}^t min_{j=1}^{l_i}\{a_{i,j}\}$$

We stress that the recommended trust value defined above captures the intuition of the trust value:

- if there is $dtv^{N_{i-1}}(N_i)$=0, then $rtv^S(T)$ =0; This means that if there is a fault node in a given path, the recommendation path should not be trusted at all.
- if $p'$=$p\cup\{N_{k+1}\}$, then $rtv^S(T,p') \leq rtv^S(T,p)$, where $p$=$\{N_1,\cdots,N_k\}$; This means that the longer the size of a recommendation path, the less trust value should be computed from individual recommenders along the path;
- if $rtv^S(T,p)$ is a positive and $dtv^{N_k}(N_{k+1})$ is positive, then $rtv^S(T,p')$ is positive, where $p$=$\{N_1,\cdots,N_k\}$ and $p'$=$p \cup \{N_{k+1}\}$; The means that the definition of the trust value of recommendation is transitive.

## 3.3 Minmax Principle for Trust Metrics

We will show that the principle for computing of trust proposed above satisfies Yao's Minimax theorem[11]. As a result, the expected running time of the optimal deterministic algorithm for an arbitrary chosen input distribution is a lower bound on the expected running time of the optimal randomized algorithm for trust evaluation. This is the most significant feature of our metrics.

Let $\Pi$ be a problem with a finite set $\Theta$ of input instances of fixed size $(k_A, k_B, aux^A(B)\ \mathcal{A})$, and a finite set of deterministic algorithms $\Gamma$=$(\mathcal{PT},\mathcal{SG})$. For an input $inp \in \Theta$, and algorithm $alg \in \Gamma$, let $\mathcal{T}(\Theta, alg)$ be the running time of an algorithm $alg$ on an input $inp$. For probability distribution $\iota$ over $\Theta$, and $\tau$ over $\Gamma$. Let $inp_\iota$ denote a random input chosen according to $\iota$ and $alg_\tau$ denote a random algorithm chosen according to $\tau$. Then by Yao's Minimax theorem[11], we have the following statement

$$min_{alg \in \Gamma} E[T(inp_\iota, alg)] \leq max_{inp \in \Theta} E[T(inp, alg_\tau)]$$

In other words, the expected running time of the optimal deterministic algorithm for an arbitrary chosen input distribution $\iota$ is a lower bound on the expected running time of the optimal randomized algorithm for $\tau$.

**Remarks** We remark that in case of two paths with the same trust value, say $0.9 \diamond 0.9 \diamond 0.3 = 0.4 \diamond 0.3 \diamond 0.3 = 0.3$, we will simply compute the mean of direct trust values in the path and then choose the path with the highest value (if the values are still same for different paths, then we can choose path according to the history record of nodes in the path). We stress that an alternative to avoid this problem is to use the product operator that is restricted to the interval [0,1] (see [1] and [6] for more details). Although the product operator has all required properties claimed above, we do not know whether the product operator satisfies Yao's Minimax theorem[11] or not. This leaves an interesting research problem.

## 4 Computing of Trust in the Small World

The concept of small world in the context of wireless networks first studied by Helmy [5] enables a path-finder to search paths originated from a source node to a designated target node in wireless networks efficiently. Based on this observation, we provide a practical approach to compute trust in wireless networks by viewing individual mobile device as a node of a delegation graph $G$ and mapping a delegation path from the source node $S$ to the target node $T$ into an edge in the correspondent transitive closure of the graph $G$, from which a trust value is computed.

### 4.1 Path-Finder

Since wireless networks typically can be formalized as a small world [5], we thus use the technique presented in [15] for our path-finder. That is, we run an initiator of a route discovery process to generate a route request, which contains the identifiers of the initiator and the target, and a randomly generated query identifier. Each intermediate node that receives the request for the first time appends its identifier to the route accumulated so far, and re-broadcasts the request. When the request arrives to the target, it generates a route reply. The route reply contains the identifiers of the initiator and the target, the accumulated route obtained from the request, and a digital signature of the target on these elements. The reply is sent back to the initiator on the reverse route found in the request. Each intermediate node that receives the reply verifies that its identifier is in the route carried by the reply, and that the preceding and following identifiers on the route belong to neighboring nodes. If these verifications fail, then the reply is dropped. Otherwise, it is signed by the intermediate node, and passed to the next node on the route (towards the initiator). When the initiator receives the route reply, it verifies if the first identifier in the route carried by the reply belongs to a neighbor. If so, then it verifies all the signatures in the reply. If all these verifications are successful, then the initiator accepts the route.

## 4.2 Transitive Graph and Transitive Signature in PKI Setting

**Notion** Given an undirected graph $G$, two vertices $u$ and $v$ are called connected if there exists a path from $u$ to $v$; Otherwise they are called disconnected. The graph $G$ is called connected graph if every pair of vertices in the graph is connected. A vertex cut for two vertices $u$ and $v$ is a set of vertices whose removal from the graph disconnects $u$ and $v$. A vertex cut for the whole graph is a set of vertices whose removal renders the graph disconnected. The vertex connectivity $k(G)$ for a graph $G$ is the size of minimum vertex cut. A graph is called $k$ vertex connected if its vertex connectivity is $k$ or greater.

**Syntax of Transitive Signatures** A probabilistic polynomial time undirected transitive signature scheme $TS$ is specified by four polynomial-time algorithms $TKG$, $TSig$, $TVer$ and $Comp$ [13]:

– The randomized key generation algorithm $TKG$ takes input $1^k$, where $k \in N$ is the security parameter, and returns a pair $(tpk, tsk)$ consisting of public key and security key of a transitive signature scheme.
– The signing algorithm $TSig$ consists of a pair of separate algorithms: a vertex/node signing algorithm $VSig$ and a edge signing algorithm $ESig$. $VSig$ is a stateful or randomized algorithm that takes input of the security key $tsk$ and a node $v_i$ and returns a value called certificate of node $v_i$ which is denoted by $Cert_{v_i}$. $ESig$ is a deterministic algorithm that takes input of the security key $tsk$ and two different nodes $v_i, v_j \in V$, and returns a value called certificate of edge $\{v_i, v_j\}$ relative to $tsk$. $TSig$ maintains states which it updates upon each invocation.
– The deterministic verification algorithm $TVf$ consists of a pair of separate algorithms $(VVer, EVer)$. $VVer$ is the deterministic vertex/node certificate verification algorithm that takes input of $tpk$ and a certificate $Cert_{v_i}$ of vertex $v_i$, returns either 1 or 0. $EVer$ is the deterministic algorithm that takes input of $tpk$ and two nodes $v_i, v_j \in V$, and a certificate $\sigma$ of edge $\{v_i, v_j\}$, returns either 1 or 0 (in the former case we say that $\sigma$ is a valid signature of edge $\{v_i, v_j\}$ relative to $tpk$).
– The deterministic composition algorithm $Comp$ takes input of $tpk$ and nodes $v_i, v_j, v_k \in V$ and values $\sigma_1$, $\sigma_2$ to return either a value of $\sigma$ or a symbol *null* indicate failure.

**The Definition of Security** Associated to transitive signature scheme ($TKG$, $TSig$, $TVer$, $Comp$), adversary $Adv$ and security parameter $k \in N$, is an experiment which is denoted by $Exp^{tu-cma}_{TS,Adv}(k)$ that returns 1 if and only if $Adv$ is successful in its attack. The experiment begins by running $TKG$ on input

$1^k$ to get keys $(tpk, tsk)$. It then runs $Adv$, and providing this adversary with input $tpk$ and oracles access to the functions $ESig(tsk, \cdot)$ and $VSig(tsk, \cdot)$. The oracles are assumed to maintain state or toss coins as needed. Eventually, $Adv$ will output $(v_{i'}, v_{j'}) \in V \times V$ and some value $\tau'$. Let $E$ be the set of all edges $\{v_a, v_b\}$ such that $Adv$ made oracle queries $v_a, v_b$, and let $V$ be the set of all nodes $v_a$ such that $v_a$ is adjacent to some edge in $E$. We say that $Adv$ wins if $\tau'$ is a valid signature of $\{v_{i'}, v_{j'}\}$ relative to $tpk$ but the edge is not $\{v_{i'}, v_{j'}\}$ in the transitive closure $G$ of a graph $G = (V, E)$. The experiment returns 1 if $Adv$ wins and 0 otherwise. The advantage of adversary in its attack on $TS$ is the function $Adv^{tu-cma}_{TS,Adv}(\cdot)$ defined for $k$ by

$$Adv^{tu-cma}_{TS,Adv}(k) = \Pr[Exp^{tu-cma}_{TS,Adv}(k) = 1]$$

We say that a transitive signature scheme is transitively unforgeable under adaptive chosen-message if $Adv^{tu-cma}_{TS,Adv}(k)$ is negligible for any adversary $Adv$ whose running time is polynomial in the security parameters $k$.

**Known Implementation in the PKI Setting** There is an efficient implementation of transitive signature presented in [13], which is sketched below:

- System parameters: Let $p, q$ be two large safe primes such that $p - 1 = 2p'$ and $q - 1 = 2q'$, where $p', q'$ are two primes with length $l'$-bit. Let $n = pq$ and $QR_n$ be the quadratic residue of $Z_n^*$. Let $h$ be two generators of $QR_n$. Also chosen are a group $G'$ of prime order $s$ with length $l$ and two random generators $g_1, g_2$ of the group $G'$. We also assume that the discrete logarithm problem is hard in $G'$.
- Representation of vertex: a vertex $v_i = {g_1}^{x_i} {g_2}^{y_i}$ in an undirect graph $G$, is an element of group $G'$.
- Representation of edge: Signature of an edge $\{i, j\}$ is a pair: $\alpha_i = x_i - x_j \mod s$ and $\beta_i = y_i - y_j \mod s$ in an undirect graph $G$.
- Certificate of vertex: The certificate of each vertex $v_i$ in authenticated graph is defined by $Cert_i = (e_i, y_i, t_i)$ derived from the signature equation: ${y_i}^{e_i} = Xh^{H({g_1}^{t_i} {g_2}^{H(v_i)})} \mathrm{mod} n$.
- A transitive signature: We now can describe our transitive signature scheme: on input $1^k$ ($k$ stands for the system security parameter), the key generation scheme algorithm creates a pair of signing keys $(spk, ssk)$ for the signature scheme defined above. The signing algorithm $TSign = (VSign, ESign)$ maintains the state of $VSign(i), ESign(i, j)$, where the node $v_i = {g_1}^{x_i} {g_2}^{y_i}$ and a signature of the vertex is defined by $Cert_i = (e_i, y_i, t_i)$ which is derived from the signing equation ${y_i}^{e_i} = Xh^{H({g_1}^{t_i} {g_2}^{H(v_i)})} \mathrm{mod} n$. The signature of an edge $\{i, j\}$ is $\delta_{i,j} = (\alpha_{i,j}, \beta_{i,j})$, where $\alpha_{i,j} = x_i - x_j \mod s$ and $\beta_{i,j} = y_i - y_j \mod s$.
- The composition algorithm $Comp$: Given nodes $v_i, v_j$ and $v_k$ and the signatures of edge $\{i, j\}$ and edge $\{j, k\}$, it checks the validity of certificate of each node $Cert_i$, $Cert_j$ and $Cert_k$ and it checks the validity of signature of each edge $\delta_{i,j}$ and $\delta_{j,k}$. If all are valid then it outputs $\delta_{i,k} = (\alpha_{i,k}, \beta_{i,k})$.

The undirected transitive signature scheme described above is provably secure under the hardness assumption of strong RSA problem, the hardness assumption of the discrete logarithm problem as well as the $H$ is a collision free hash function in [13].

## 4.3 Removal of PKI Assumption

We stress that a collection of claimed public keys of a path must be certified. Thus, either a trusted third party (a certificate authority) or a public key infrastructure is required. To remove the concept of certified identity graph $G_x$ from the transitive signatures, we will make use of the following assumption: each node in an undirected graph $G$ has a unique identity that cannot be forged and it knows the identities of its neighbors in $G$. We remark that if our keyed-identity graph $G_x$ assumption is not met, an adversary can use different identities to different neighbors. With the help of fixed-identity assumption, an algorithm determining genuine keyed-identity can be proposed. That is, assuming that the underlying graph $G$ is $2k+1$ vertex connected with $k$ adversaries, then between every pair of good nodes, there exists at least $(k+1)$ vertex disjoint paths that traverse only good nodes (the fact that adversaries can at most prove $k$ disjoint paths to a fake node is critical for the solvability of this problem, see [3] and [10] for more details). Based on the above assumptions, we can describe our undirected transitive signature scheme below:

- system parameters: The system chooses a group $G'$ of prime order $s$ with length $l$ and two random generators $g_1, g_2$ of the group $G'$. We also assume that the discrete logarithm problem is hard in $G'$.
- individual system parameters: For each user in the network, it chooses two large safe primes $p_i$ and $q_i$ such that $p_i - 1 = 2p'_i$ and $q_i - 1 = 2q'_i$, where $p'_i, q'_i$ are two primes with length $l'$-bit. Let $n_i = p_i q_i$ and $QR_{n_i}$ be the quadratic residue of $Z^*_{n_i}$. Let $X_i$ and $h_i$ be two random generators of $QR_{n_i}$.
- representation of vertex: a vertex $v_i = {g_1}^{x_i} {g_2}^{y_i}$ in an undirect graph $G$, is an element of group $G'$.
- representation of edge: Signature of an edge $\{i, j\}$ is a pair: $\alpha_{i,j} = x_i - x_j \mod s$ and $\beta_{i,j} = y_i - y_j \mod s$ in an undirect graph $G$.
- certificate of vertex: The certificate of each vertex $v_i$ in an authenticated graph is defined by $Cert_i = (e_i, z_i, t_i, x_i, y_i)$ which is derived from the equation: ${z_i}^{e_i} = X_i {h_i}^{H({g_1}^{t_i} {g_2}^{H(v_i)})} \mathrm{mod} n_i$.

Given a path-finder program, a source node $S$ searches a collection of paths from $S$ to $T$. Since each node in a graph shares the global system parameters, it follows that each node can be viewed as a self-signed certificate in the transitive graph. Consequently, by applying the technique presented in Section 3, we can calculate the trust value immediately.

## 5 Conclusion

In this paper, we have introduced and formalized the notion of action in terms of DNF and we have formalized the notion of trust in terms of action, predict function and strategy function. We have already proposed a concise structure for computing of trust value in ad-hoc networks by mapping a path in the underlying network graph $G$ to the corresponding edge of its transitive closure graph $G^*$. Finally, we have outlined a generic model for computing of trust in ad-hoc networks.

**Acknowledgment** The first author is grateful to Professor Herbert Leitold for his invaluable comments on the publication and kind help.

## References

1. I.Agudo, J.Lopez, J.A. Montenegro. A Representation Model of Trust Relationships with Delegation Extensions, 3th International Conference on Trust Management (iTRUST'05). LNCS 3477, Springer, 2005.
2. T.Beth, M.Borcherding and B.Klein: Valuation of Trust in Open Networks. ESORICS 1994: 3-18.
3. D.Dolev: The Byzantine Generals Strike Again. J. Algorithms 3(1): 14-30 (1982)
4. J.A. Golbeck. Computing and applying trust in web-based social networks, University of Maryland, College Park, 2005.
5. A.Helmy. Small worlds in wireless networks. IEEE communication letters, Vol.7, No 10, October 2003.
6. A. Jøsang, D. Gollmann, R. Au: A Method for Access Authorisation Through Delegation Networks. Australasian Information Security Workshop 2006.
7. T.Li, H.Zhu, K.Lam: A Novel Two-Level Trust Model for Grid. ICICS 2003: 214-225, Springer Verlag.
8. S.Micali and R.Rivest: Transitive Signature Schemes. CT-RSA 2002: 236-243.
9. M.Reiter and S.Stubblebine. Resilient authentication using path in- dependence. IEEE Transactions on computers, Vol.47, No.12, December 1998.
10. L.Subramanian, R.H.Katz, V.Roth, S.Shenker and I.Stoica: Reliable broadcast in unknown fixed-identity networks. PODC2005: 342- 351.
11. A.C.Yao: Probabilistic Computations: Toward a Unified Measure of Complexity (Extended Abstract) FOCS 1977: 222 -227.
12. R.Yahalom, B.Klein and T.Beth: Trust-Based Navigation in Distribution Systems. Computing Systems 7(1): 45-73, 1994.
13. H.Zhu. New model on undirected transitive signatures. IEE Proceedings of Communication, 2004.
14. H.Zhu, B.Feng and Robert H.Deng. Computing of Trust in Distributed Networks. http:// www.iacr.org, eprint, 2003.
15. H.Zhu, F.Bao and T.Li: Compact Stimulation Mechanism for Routing Discovery Protocols in Civilian Ad-Hoc Networks. Communications and Multimedia Security 2005: 200-209, Springer Verlag.
16. H.Zhu, F.Bao and Robert H.Deng. Computing of Trust in wireless Networks. IEEE Vehicular Technology Conference, 2004.