A trust enhanced secure clustering framework for wireless ad hoc networks

Article *in* Wireless Networks · October 2014

DOI: 10.1007/s11276-014-0701-6

CITATIONS
20

READS
215

4 authors:

Pushpita Chatterjee
SRM Institute of Science and Technology
41 PUBLICATIONS **319** CITATIONS
SEE PROFILE

Uttam Ghosh
Meharry Medical College
176 PUBLICATIONS **1,541** CITATIONS
SEE PROFILE

Indranil Sengupta
Indian Institute of Technology Kharagpur
142 PUBLICATIONS **1,763** CITATIONS
SEE PROFILE

Soumya K. Ghosh
Indian Institute of Technology Kharagpur
305 PUBLICATIONS **4,782** CITATIONS
SEE PROFILE

Wireless Netw (2014) 20:1669–1684
DOI 10.1007/s11276-014-0701-6

A trust enhanced secure clustering framework for wireless ad hoc networks

Pushpita Chatterjee · Uttam Ghosh · Indranil Sengupta · Soumya K. Ghosh

Published online: 18 February 2014

Abstract Secure clustering in Wireless Ad Hoc Networks is a very important issue. Traditional cryptographic solution is useless against threats from internal compromised nodes. In light of this, we propose a novel distributed secure trust aware clustering protocol that provides secure solution for data delivery. A trust model is proposed that computes the trust of a node using self and recommendation evidences of its one-hop neighbors. Therefore, it is lightweight in terms of computational and communication requirements, yet powerful in terms of flexibility in managing trust. In addition, the proposed clustering protocol organizes the network into one-hop disjoint clusters and elects the most qualified, trustworthy node as a Clusterhead. This election is done by an authenticated voting scheme using parallel multiple signatures. Analysis of the protocol shows that it is more efficient and secure compared to similar existing schemes. Simulation results show that proposed protocol outperforms the popular ECS, CBRP and CBTRP in terms of throughput and packet delivery ratio with a reasonable communication overhead and latency in presence of malicious nodes.

Keywords Ad hoc networks · Cluster · Security · Trust · Routing · Leader

P. Chatterjee (✉) · S. K. Ghosh
School of Information Technology, Indian Institute of Technology, Kharagpur, India
e-mail: pushpita.c@gmail.com

S. K. Ghosh
e-mail: skg@iitkgp.ac.in

U. Ghosh
Department of E&ECE, Indian Institute of Technology, Kharagpur, India
e-mail: uttamg@iitkgp.ac.in

I. Sengupta
Department of CSE, Indian Institute of Technology, Kharagpur, India
e-mail: isg@iitkgp.ac.in

1 Introduction

A Wireless Ad Hoc Network (WANET) is a self-organized, decentralized and distributed collection of wireless nodes that supports dynamic topology, multi-hop routing and node mobility. The possible attacks may range from passive eavesdropping to active interference due to some inherent characteristics of such networks. Cryptographic schemes cannot help in detecting/preventing such attacks as these behaviors are continuously changing. In addition, reliability/reputation of the information received from nodes, quality of information assessment and providing various levels of access control cannot be done effectively. Such security threats can be most effectively handled using trust management systems. Trust management cannot be seen as a complete replacement for cryptography, rather a supplement to it. Due to the unique characteristics and the inherent unreliability of the wireless medium, the concept of trust in WANETs should be carefully defined.

Several trust models have been proposed for self organized networks in distributed paradigm. Among them, Balakrishnnan et al. developed a trust model [1] to strengthen the security of WANETs and to deal with the issues associated with recommendations. A continuous time Markov chain based trust model [2] has been proposed by Chatterjee et al. where trust state changes with

specified events and the authors proved that the model reaches to steady state. Wang et al. [3] proposed a mechanism extension to AODV, to distinguish selfish peers from cooperative ones based solely on local monitoring. A trust decision framework should not assume that all nodes are cooperative. Trust management, including trust establishment, trust update, and trust revocation, is also much more challenging than in traditional centralized environments.

Several algorithms have been proposed for clustering ad hoc networks in recent years. Clustering of nodes has been achieved with different perspective such as highest-id, lowest-id, energy efficient, mobility based clustering and many others. But none of them are able to completely handle the secure clustering. Among them, an on-Demand WCA [4], proposed by Chatterjee et al., does not specify the well defined clustering mechanism, new cluster head selection, and other important issues. Among several secure solutions based on clustering ad hoc networks, Vaidya et al., and Sudarshan et al., have proposed several algorithms [5, 6] for distributed leader election in ad hoc network scenario but the malicious nature of the nodes is not considered.

Therefore, it is evident that a self-organized distributed secure clustering framework is needed to enhance the overall security of the network with minimal overhead. In this light we briefly address the following issues:

- provides trust based secure clustering in fully distributed independent WANET. Instead of assigning single authority to a particular node, trust is evaluated in a collaborative fashion.
- restricts any node to join multiple clusters simultaneously.
- detects malicious behavior of any compromised or attacker (both inside and outside) nodes.
- ensures the graceful leave and detects graceless leave of a node.
- analyzes network capacity to increase lifetime of a node as well as the network.

We analyze our proposed protocol to show that it efficiently forms and maintains clusters with less communication overhead and low latency compared to similar existing protocols. Further, the proposed protocol reduces the security threats present in WANET with a minimal use of cryptographic functions. Simulation results are given to show that the proposed protocol offers higher throughput and packet delivery ratio in presence of malicious nodes compared to the popular CBRP and CBTRP.

The rest of the paper is organized as follows: The preliminaries, useful definitions and the assumptions made in this paper are listed in Sect. 2, which is followed by Sect. 3, wherein our proposed secure distributed trusted clustering protocol is presented. Section 4 presents analysis of the proposed protocol with respect to performance and security consideration, and brief comparison with existing trust based protocols. The simulation results are shown in Sect. 5. Finally, Sect. 6 concludes the paper.

2 Preliminaries

We consider an autonomous wireless ad hoc network working on its own and it has no connection to the external world. The network is formed starting from one node and then the other nodes add up one by one. The nodes are free to move around and can join/leave the network at any point of time. Therefore, the size and the topology of the network are dynamic and unpredictable in nature.

2.1 Node types

In the proposed clustering framework, we have categorized mobile nodes in the following types:

- Guest Members (GM)—are mobile hand held devices ranging from laptops, smart-phones, tablet PCs and others, that can communicate between each other are referred as nodes. Each node can operate in promiscuous mode. When a node becomes member of any cluster (initial) referred to as guest member (GM). When a node joins under a leader by giving vote, it is known as member node (MN).
- Clusterheads (CH)—are MN with the following added responsibility to maintain the proper functionality of the network: detect and isolate the malicious nodes; generate session key for secure communication and *Warning* message to misbehaved MN; handle node joining and registration; collect recommendation trust from members and calculate resultant trust.
- Guard nodes (GN)—are MN with some added responsibilities to monitor the behavior of a CH and to convey the information about misbehaved CH to other MNs. MNs are grouped and scheduled to act as GNs in a round robin fashion such that all the MNs are equally responsible to maintain the cluster.
- Gateway nodes(GW)—are MN which reside at border of the cluster that means they have a connection between two adjacent clusters. For security requirement all border nodes cannot be gateway. Only trusted border nodes can be acted as GW nodes.
- Standalone (SA)—If a node does not have any node in its radio range becomes standalone node(SA). It cannot communicate to others until it enters broadcast range of any node.

2.2 Definition

In order to give a formal description of trust and trustworthiness, we give our definitions of trust/trustworthiness as follows:

- *Direct trust* Direct trust is referred as the relationship or opinion generated from direct observation by a node about another node in its radio range using local monitoring method.
- *Indirect trust* Indirect trust is referred as the indirect observation about a node from the opinion of other nodes.
- *Resultant trust* Resultant trust is the collection of opinions from other nodes to represent resultant belief about a node.
- *Trustworthy route* Trustworthy route is the most trusted route of available routes towards destination.

In order to formalize the clustering framework, we point out some important definitions which make the basis of distributed clustering:

- *Cluster radius* In CH based clustering, nodes are organized within a given maximum number of hops, R, from a node called CH into a cluster. The parameter R is referred as the *cluster radius*.
- *Cluster-recycle distance* The distance between two cluster heads must be less than or equal to a predetermined number of hops, D ($D \leq R$). The parameter D is referred as the *cluster-recycle distance*.
- *Cluster information* For clustering purpose, each node maintains a small amount of information of itself and its neighboring nodes. The information of a node is referred as the *cluster information* of a node. This information is used to identify a node uniquely.

3 Proposed protocol: secure distributed trusted clustering

In this section, we present secure distributed trusted clustering protocol that utilizes a proposed trust model to compute the trust value of a node and securely forms the cluster in a distributed way. Routing also presented where it provides the trusted route towards destination.

3.1 Trust model

Trust model is presented in Fig. 1 and its components are as follows: *Monitor module*: The main function is to monitor the packet transmissions and collect useful data about network functionalities, such as packet forwarding, dropping, false route injection. Each node monitors each neighbor node in short term monitoring. However, if a node finds any neighbor is malfunctioning, it monitors the particular in long term. *Processing module*: This module compares the collected data with predefined threshold and finds the deviation. *Combine module*: Getting the deviation from processing module, this module calculates direct trust and combines direct evidence with collected recommendation from its neighbors. *Decision module*: After calculating resultant trust, this module is called to categorize nodes according to their trust metric.

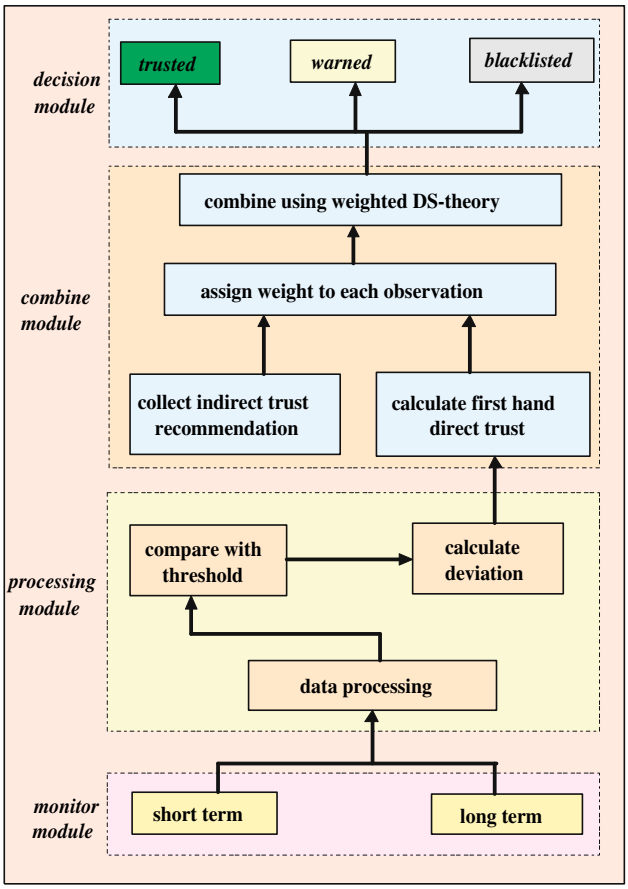

Fig. 1 Trust generation model

3.1.1 Trust generation

Once the trusted cluster is formed, *resultant trust* calculation of member nodes in a cluster is started. Each node including CH determines its *direct trust* opinion about its all neighbor nodes. In order to calculate *resultant trust* CH requests *recommendation trust* about the node under review. Neighbor nodes of the said node send their opinion. CH calculates *resultant trust* by combining all direct and indirect observations using proposed weighted DS method for combining evidences described in Sect. 3.1.3. After trust calculation, CH generates and digitally signs a *Trust_Cert* for each MN depending upon the *resultant trust*. A MN receives the *Trust_Cert* if it is *trusted* or a *warn* message if its trust value in the *warned* range and *blacklist* message otherwise. This is to be noted for security purpose only encrypted and signed *Trust_Cert* is sent. To

prevent the cheating, CH stores the trust value of its members in the *Member Table* and periodically sends the *Available-neighbor list* to each MN. This list consists of node-IDs which are permitted to take part in route; i.e., their *resultant trust* is greater than 0.25. This *Available-neighbor list* is further used to set up route between source and destination.

3.1.2 Direct trust prediction

As trust value is subject to review, after a certain time, each node evaluates the trustworthiness of its one hop neighbor nodes. We carefully choose certain parameters to quantify the trust of a node according to its contribution towards proper functioning of the network and minimizing the number of bad nodes from the network. The behavior of a node which plays the important role in formalizing trust are : packet forward, packet drop, and packet false injection. The direct trust is calculated using the following procedure. The predicted trust evidence associated with respective parameter is denoted as T_{Fwd} for forward ratio, T_{Drp} for drop ratio and T_{Fls} for false injection ratio at time t can be predicted from their previous p state values using the following equations of AR(p) model. The autoregressive model AR(p) is one of a group of linear prediction formulas that attempt to predict an output Y_n of a system based on the previous outputs ($Y_{(n-1)}$, $Y_{(n-2)}$, …, Y_1). This model has been is used to predict network traffic, location of a node, remaining energy and trust [7].

$$\begin{aligned} T_{Fwd}(t) &= K_1 + W_1 * \Sigma_{i=1}^{p} T_{Fwd}(t-i) + E_1(t) \\ T_{Drp}(t) &= K_2 + W_2 * \Sigma_{i=1}^{p} T_{Drp}(t-i) + E_2(t) \\ T_{Fls}(t) &= K_3 + W_3 * \Sigma_{i=1}^{p} T_{Fls}(t-i) + E_3(t) \end{aligned} \quad (1)$$

wherein W_1, W_2 and W_3 are weights, $E_1(t)$, $E_2(t)$ and $E_3(t)$ are noise term with mean 0, and K_1, K_2, K_3 are constants and can be omitted for simplicity. By the autoregression formula specified in Eq. 1, the output at time t can be estimated knowing W_i and p. The main problem in AR modeling is to find proper W_i, so as to represent the model best. To compute AR model coefficients, different algorithms are there in literature. The Least-Square (LS) approach and the Yule Walker (YW) approach are the basic ones [8]. The model order p defines the sample set size. There may be confusion with the size of sample set. Generally, it seems the larger the model order, the more accurate the model as a large model order implies that a large number of samples from the past. In reality, when the samples are too old, there is a chance of interference to the prediction as they may not be able to reflect the current value of the time series. Thus another problem is to select the optimal model order p. Two well-known selection criterions [9] are Final Prediction Error (FPE) and Akaike Information Criterion (AIC). We are restricting ourselves to elaborate the mathematical model [8, 9] here, as both of these are well known mathematical model for predicting W_i and p. To determine the model order, we have used the AIC method. The LS method has been used to estimate parameters in our present work. At time t, each AR(p) model gives a prediction for trust associated with each trust parameter respectively described in Eq. 1.

Now, these trust evidences are combined to generate trust using weighted combination. Weight is assigned to each calculated trust depending upon the environment. in this way each node calculated direct trust evidence about its neighbor. The trust evidences about a node under observation are combined using a proposed weighted DS theory of combining evidences. In the following section we describe this process in detail.

3.1.3 Proposed modification in DS theory for combining evidences

The DS-theory [10] assumes that all evidences are given same priority while combining evidences to get resultant one. But in the real scenario, all evidences can not be equally trusted. In our work, the process of building a generalized trust model working with nodes of different accuracy, it is often difficult to get the mobile nodes to report their observation accuracy along with appropriate ignorance estimation. There could be malicious nodes present in the system generate false report about their recommendation about a good node. Giving same priority to all evidences may result erroneous $T^{M}_{Resultant}$. Alternatively, the malicious report may affect the decision making. To resolve this issue, we propose a new concept for a *Weighted Dempster-Shafer Evidence Combining Rule*. The basic idea is this: We know how a node performs historically, because each node has a direct trust rate about all its neighbor nodes; so we can then use this direct trust rate as the reference to decide how much we trust the node's current estimation from its current observation. The highest weight is assigned to the direct trust value of the query node about the node under review. The weights assigned to the belief of referee nodes i.e., the common neighbor node according to their direct trust. Let m_i and m_j are the direct trust observation w_i, w_j is the weight estimation inferred by the direct trust evidence of node N_i and N_j respectively. It is evident that any node gives the highest weight on its own observation i.e., its direct trust observation. Resultant trust is calculated as follows:

$$T^{M}_{Resultant} = \frac{\sum_{A_k \bigcap A_{k'}=A} [w_i m_i(A_k) \cdot w_j m_j(A_{k'})]}{\sum_{A_k \bigcap A_{k'}=\phi} [w_i m_i(A_k) \cdot w_j m_j(A_{k'})]} \quad (2)$$

It may be noted here that this weight based trust calculation ensures lower impact by malicious node on resultant trust calculation. We validate this claim (Claim 7) in "Appendix".

Assignment of weights to belief Here, we determine weight w of belief of each neighbor is directly proportional to its direct trust. Another important factor in wireless medium is signal fading. As distance increases, resulting in message garbling and message loss. So weight can be assigned based on the distance D of the recommender node from the query node. We are considering weight is proportional to inverse square of distance. So for predicting the weight, it can be easily seen that w is directly proportional to T^2 and inversely proportional to D^2. The distance D is the Euclidian distance and trust T. Thus we can write,

$$\begin{aligned} w_1 &= \omega_1 * T^2 \\ w_2 &= \frac{\omega_2}{D^2} \\ w &= \frac{\omega * T^2}{D^2} + p \end{aligned} \tag{3}$$

In distributed environment the proportionality constant ω depends on some environment variables that can be represented as a function of those variables. The variable p is the priority factor which depends upon type of nodes, where x_i is environment elements that can be chosen depending upon the application. Environment variable may be ranging from Signal-to-Noise Ratio (SNR), jitter (for multimedia application), delay, distance etc., are taken into consideration depending upon the application. As we consider a trusted environment, two parameters *Direct trust* of the query node about the node giving the belief and the *distance* between the query node and the belief giving node. Considering the variables are independent the function $f(x)$ can be defined as

$$f(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} (x_i.x_j), \quad \forall (x_i \neq x_j) \tag{4}$$

In our proposal we are dealing with three variables: $x_i = T$ and $x_j = D$ and n=2.

3.2 Clustering framework

The proposed clustering technique uses both hop distance and number of member nodes in a cluster to control the cluster structure. The most trustworthy node in each cluster takes the role of the cluster master known as clusterhead (CH). The distance between a CH and any member of the cluster is equal to a predetermined *cluster radius*, we consider $R = 1$. Thus the proposed clustering is *1-hop clustering* and it may be extended for k hops ($R = k$). When a node moves away from its CH and the distance between them becomes larger than the *cluster radius*, it searches for an available CHs in its radio range. If the search is successful, the node joins the cluster to the CH, otherwise it becomes CH to form a new cluster. When the distance between two clusterheads is detected to be less than or equal to *cluster-recycle distance* ($R = 1$ for this proposal), the cluster with less number of member nodes is recycled. Each of the nodes of the recycled cluster finds a new cluster to join. Let the *cluster information* of a node i be presented by the following 7-tuple:

- *Public key (Pub_key)* This key of size 1024-bit RSA is generated by the node itself when it boots.
- *Node-ID* The node-ID (of size 64-bit) of a node is generated using a secure one-way hash function on its self generated public key. The first bit of this node-ID is set to 0 to indicate it as member node.
- *Cluster-ID* This ID (of size 64-bit) is generated by the CH using the secure one-way hash function on its node ID (i.e. CH-id) and a random number.
- *CH-ID* This CH-ID (of size 64-bit) is its node-ID except the first bit is set to 1 to indicate it as CH.
- *Trust* This field indicates the status of a node may be 1, 0 or −1. Here, 1 indicates *trusted*, 0 indicates *warned*, −1 indicates *no-definite* status (initial state).
- *SOC*-State of charge (*SOC*) is the ratio of residual capacity(*RC*) and full charge capacity (*FCC*) of the battery.
- *Neigh*-number of neighbors whose *cluster information* the node has already received.

The primary contribution of this proposed distributed trusted clustering can be encapsulated into the following tasks:

- *Cluster set up* This phase is a combination of the following:
  - *Initial clustering* Initially nodes join to the network one by one and forms the clusters.
  - *Trusted clustering* After the formation of initial clusters nodes start to take part in intra cluster communication. A secret voting mechanism is invoked for CH election and forms the stable trusted clusters.
- *Cluster maintenance* A node can join or depart from the cluster due to mobility or switch-off itself. Therefore, cluster may split at any time and merge again. Cluster maintenance addresses these situations.
- *Secure routing* Secure intra cluster routing can start after the initial clustering, whereas inter cluster routing can only possible after the trusted cluster setup.

3.2.1 Initial clustering

When a node, say i, is powered up, it sets its *cluster information* to be (*Pub_key, node-ID, 0, 0*, −1, *SOC, 0*) indicating that it is in *initialization* phase (no Cluster-ID and no CH-ID). Subsequently it searches for neighbor nodes. If found, it sends its

cluster information to all of its neighboring nodes. Upon receiving the *cluster information* of node *i*, each neighbor node adds it as neighbor and sends *cluster information* to it. When node *i* receives the *cluster information* from all of its neighbor nodes, it checks whether there is at least one such node which is not in the *initialization* phase (having CH-ID and Cluster-ID). If it finds any CH, node *i* joins as *Guest member (GM)* and updates *cluster information*. Subsequently it broadcasts *cluster information*. Upon receiving the *cluster information* of node *i*, each neighbor node updates neighbor table. This phase is referred as *initial clustering*. CHs in this initialization phase is referred as *initial CH* or simply *ICH*. If node *i* fails to find any CH, it becomes *standalone* node and waits for any existing CH for a certain time, otherwise it declares itself as CH. After certain round of communication, CH election is revoked. The most trustworthy node is elected as CH by a secret voting scheme and voters become cluster *member nodes*. This phase is known as *trusted clustering*. An elected CH periodically broadcasts *Join-Cluster* message attracting new comer or *standalone* nodes to join under it.

3.2.2 Trusted clustering

After the cluster formation by initial clustering, *Guest member* nodes start to communicate. At this phase, inter cluster communication is not allowed due to security primitives. Each node starts to monitor its neighbors including CH to decide the direct trust opinion. The overall direct trust mechanism is discussed in Sect. 3.1.2. After certain round of communication CH election is revoked. ICH or any other GM wants to become CH, signs and broadcasts *Vote_me* message with current battery power *SOC*, *node-ID* and neighbor ratio N_R. Following are the metrics depending on which a node is voted as CH: direct trust T_{DIR}, state of charge (*SOC*), neighbor ratio (N_R) and stability factor (*SF*). Each node periodically calculates its direct trust T_{DIR} about its 1-hop neighbors including *ICH*, after joining under the *ICH*. Each node broadcasts 1-hop *Hello* message in an interval known as *Hello-interval* (t_H). Stability factor (*SF*) of a neighbor node is the ratio of the number of received *Hello* message and the expected number of *Hello* message in a particular interval τ_H. Therefore, the expected number of *Hello* message is calculated by taking ratio of t_H and τ_H. Each node calculates N_R by taking the ratio of the degree of a node and allowable load per node. Each neighbor of the candidate CH keeps the *SF* and T_{DIR}. Upon receiving the *Vote_me* message they execute the following Algorithm1. If the candidate node meets the CH election criteria, neighbor nodes secretly vote for *M* by signing *Vote_me* message. The secret voting mechanism is achieved by the use of parallel multiple signatures [11], which is described below: Let p be a large prime, g be a primitive element in $GF(p)$, SK_{M_i} is the secret key and $PK_{M_i} \equiv_p g^{SK_{M_i}}$ is the corresponding public key of node M_i. A candidate node M_i for CH, performs the following steps to sign the *Vote_me* message and broadcasts his signature to its 1-hop neighbors. Following are the steps for generating parallel multiple signature:

- *Step 1* computes $S \equiv_p PK_{M_i}^{Vote\_me}$
- *Step 2* generates a random number $r_i \in [1, p-1]$ and compute $k_i \equiv_p g^{-r_i}$;
- *Step 3* solves the congruence $(S + t_i)SK_{M_i} \equiv_{p-1} r_i - k_i$ and derive t_i
- *Step 4* broadcasts (t_i, k_i, S), M_i 1-hop.

A node M_j, wherein $2 \leq j \leq n$ and $i \neq j$, receives (t_i, k_i, S), M_i, performs the following steps:

- *Step 1* computes $Vote\_me \equiv_p PK_{M_i}^{S+t_i}.k_i g^{k_i}$ and checks $S \equiv_p PK_{M_i}^{Vote\_me}$.
- *Step 2* if M_j wants to join under the candidate M_i, it signs the *Vote_me* message:
  - computes $k_j \equiv_p Vote_m e.g^{-r_j}$ and solves t_j that satisfies $(S + t_j)SK_{M_j} \equiv_{p-1} r_j - k_j$;
- *Step 3* M_j sends $\{(t_j, k_j, S), M_j\}$ to M_i.

Receiving all individual signatures $\{(t_j, k_j, S), M_j\}$, where $2 \leq j \leq n$, M_i performs the following steps.

- *Step 1* Verifies each signature $\{(t_j, k_j, S), M_j\}$ by computing $Vote\_me_r \equiv_p PK_{M_j}^{S+t_j}.k_j g^{k_j}$ and checks whether $Vote\_me_r$ and *Vote_me* are same.
- *Step 2* If the verification in step 1 succeeds, M_i computes $k_j.Vote\_me^{-1} \equiv_p g - r_j$.
- *Step 3* generate a random number $r \in [1, p-1]$ such that $r \neq r_i$ and computes R and solves T as follows:

$$R \equiv_p Vote\_me.g^{r_2} \ldots g^{r_n}.g^{-r}.g^{k_2} \ldots g^{k_n}.$$
$$(PK_{M_2})^{t_2} \ldots (PK_{M_n})^{t_n}$$
$$(S + T) \equiv_{p-1} SK_i^{-1}.(r - R)$$

The resulted multi-signature on *Vote_me* message is $\{T, R, S\}$. A node may receive multiple *Vote_me* messages form different candidate nodes, in that case it would vote the best candidate. It is restriction imposed in this model that once a node is vote for some candidate, it is not allowed vote to others. We use the parallel signature scheme in *fork-all* and *join some* method. In a parallel multiple signature scheme, the signature of each signer is on the content of the form, and not on the signature of other signers. If the form can be duplicated, a copy of the form will be distributed to each of the parties. This mechanism is called fork. In fork-all method, the approval and the signature will be performed simultaneously for different parties. The mechanism to collect the signatures is referred as join. In Join-some, there is no need to wait for all the signatures, but only those who are obligatory and those that

satisfy the additional conditions. In the parallel multiple signature scheme, all the signatures have to be validated, one by one, and for each form. Here, the signature manager (SM) is the candidate itself. *Vote_me* message contains a Boolean field *Vote*, if a node chooses the candidate in polling, it sets this *Vote* field as 1 or 0 otherwise. After voting it signs the message and sends it to candidate. The *ForkAmount* is equal to *n*, where *n* is number of neighbors. In *join some* policy, the minimum number of *JoinAmount* is *n*/2 would be sufficient to elected as CH. *SM* validates signature one by one. The voting exception is raised that the voting procedure is revoked if the number of valid signatures is less than *JoinAmount*. Thus validating and counting the vote a CH is elected and voters become member of the cluster. It can be easily seen that the parameters used to elect CH, have a great impact on proper network functionality.

Algorithm 1: Secure Distributed Clusterhead Election (CH-ELECT)

Input: Different trust evidences about a mobile node
Output: The CH and Cluster with Member Nodes

foreach *1-Hop neighbor N* **do**
  if *M is* Trusted **then**
    Calculate the Global_weight (G_w^M) of M;
    $G_w^M = \sum_{j=1}^{4} w_j * Lead_j$;
    $Lead_1$, $Lead_2$, $Lead_3$, $Lead_4$ are T_V, SF, SOC, N_R;
    if $G_w^M \geq G_w^{TH}$ **then**
      N sends digitally signed $Vote\_me$ message to M ;
    else
      N drops reqest;
    end
  else
    N drops *Start-election* ;
  end
end
M waits for τ and then calculates $\mathbf{T}_{Vote\_me}$;
if $\mathbf{T}_{Vote\_me} > Threshold - \mathbf{T}_{Vote\_me}$ **then**
  M broadcasts *Leader* message with voter list and$\mathbf{T}_{Vote_me}$;
else
  M waits forτ;
end
if *any node B finds its*node_{id} *is wrongly included in the voter list* **then**
  B generates *Spurious-voter* message and broadcast;
else
  B broadcasts *OK-voter* message;
end
nodes wait for interval ξ and counts *Spurious-voter* & *OK-voter* ;
if Spurious-voter=*0 &* OK-voter=$\mathbf{T}_{Vote\_me}$ **then**
  each node signs *Leader-certificate* & sends to M;
else
  node generates *Warn* message to M;
end
M declares itself as CH and voter nodes are cluster members;

3.2.3 Cluster maintenance

The efficacy of cluster maintenance phase depends on how efficiently any clustering scheme can handle the following issues:

Cluster merge Consider, allowable load or number of nodes per cluster is *k*, if any cluster size (no. of member nodes) is less than *k*/2, referred as *shortfall cluster*. The initialization phase may end with a large number of shortfall clusters or blacklisting of nodes may lead to shortfall clusters. CH of a shortfall clusters, broadcasts *Merge-Cluster* message. Any other shortfall CH sends *Ack-Merge-cluster* message as an acknowledgement of clusters merging. CH requests its cluster member to join under *Join-New-cluster* with replying CH-ID. Upon joining, member nodes send *Confirm-New-Cluster* message. If all member node can join and send *Confirm-New-Cluster*, the merging ends, else new search starts. Any CH upon finding itself as leader of merged cluster broadcasts a member list, receiving this message node updates cluster information. Thus completes merging.

Cluster split Each node periodically beacons Hello messages (consists of node-ID, CH-ID and cluster-ID) to its neighbors. Each node along with CH keeps a neighbor table with corresponding node-ID, cluster-ID, CH-ID, and stability factor (SF). When CH finds that SF falls below a predefined threshold (application specific) for a certain time to a member node, it detects cluster splitting occurs and CH broadcasts broken link message with node-ID of that member to other members of it. Cluster splitting occurs due to mobility or switch-off of a node. It may happen that a CH departs from the network or due to mobility of CH, member nodes are not getting Hello messages from it. If SF is less than the threshold, member nodes search another cluster to join or in the worst case calls the CH election procedure.

3.3 Routing

We consider that only *Trusted* node are allowed to communicate outside the cluster. AODV is used as a basic routing protocol for all intra and inter cluster communications. The *trustworthy route* is selected for communication. Route is established by avoiding malicious nodes. Routing is accomplished in three steps:

Neighbor management Each node keeps track of its neighbors by sending periodic *Hello* messages.

Route discovery When a source (S) node has to send a data packet to a destination (D) node: there are two possible cases; either the D may be in the same cluster under same CH or the D is outside the cluster. In our proposal, any route discovery is initiated by CH only for security reason. Source first sends route request (RREQ) to CH with <Source-ID, Destination-ID, is-neighbor>. Here,

$is - neighbor$ is a boolean variable and it is 1 when S and D are neighbor; and 0 otherwise. Then CH checks the state of the S and D. If it is intra cluster, CH checks the trust state. If either S or D is *blacklisted*, RREQ is dropped, if not, $is - neighbor$ is checked. If $is - neighbor = 1$, CH generates session key and generates route permit. If $is - neighbor$ is 0, RREQ is broadcasted. On receiving RREQ message, MN checks whether it teself the destination or the D is its neighbor, it then sends RREP otherwise rebroadcasts RREQ. This process continues until D is reached. If D is not the member of same cluster, CH checks state of the node. If source is *trusted*, CH broadcasts the RREQ. Then with the help of gateways nodes and other CHs route is established.

Route maintenance When a node on the active route finds any link is broken towards destination, it generates route error (RERR) and sends it to CH. CH unicasts RREQ to the reporting node and route discovery starts from that node only. This process significantly reduces routing overhead.

For assuring security, CH initiates the route discovery by checking the status of node and once the route established, CH generates session key *SK* and sends to source and destination by encrypting it with the public key of respective nodes. At the data delivery phase, packet are encrypted with this session key. So using some light weight cryptography, active attacks like message forgery and passive eavesdropping can be prevented.

4 Analysis of the protocol

In this section, we first analyze the performance and security of the proposed clustering protocol and then compare with other similar existing protocols.

4.1 Performance analysis

Claim 1 *There may be at most* k $(k > 0)$ *number of malicious nodes* $n \geq 2k + 1$ *(where the total number of nodes in a cluster is n); such that the protocol runs successfully.*

Proof There may be two possible cases: **Case 1:** Either number of malicious nodes is equal to k. In this case, the candidate node for CH is elected by the $(n - 1)/2$ number of non-malicious nodes. The number of malicious nodes k is among $(n - 1)$ nodes then there is $(n - 1 - k)$ number of non-malicious nodes. Therefore, $(n - 1 - k) = (n - 1)/2$, or it can be written as $n = 2k + 1$. **Case 2:** or number of malicious nodes is less than k. Let j (where $j < k$) be the number of malicious nodes. Then the candidate node is elected as CH by the $(n - 1 - j)$ number of non-malicious nodes. Here also the candidate node gets at least $(n - 1)/2$ number of votes from the non-malicious nodes. Therefore, $(n - 1)/2 < (n - 1 - j)$ Or $n > 2j + 1$.

Thus considering two possible cases our claim is proved.

Claim 2 *The message overhead of the proposed scheme is* $O(1)$ *packet per time step per node in the cluster.*

Proof In the proposed scheme, additional messages are incurred due to cluster formation and maintenance. The consequence of these additional messages is called message overhead. In the following, we show that The message overhead of the proposed scheme is $O(1)$ packet per time step per node in the cluster. Here, the continuous runtime is divided into discrete time steps, which are the duration between the time when a message is sent and the time when the message is received and processed by a receiver [12].

Cluster formation In the proposed scheme, clusters are formed in two phases, namely *initial clustering* and *trusted clustering*. *Initial Clustering*: In order to discover its neighbor and inform its cluster information, each node periodically broadcasts HELLO messages per time step. Thus, this introduces an overhead per time step for all nodes in the cluster is

$$Overhead_{cluster\_information} = f_{HELLO} * n \tag{5}$$

The initial CH or ICH periodically broadcasts a Join-cluster message per time step. This message can be treated as periodic HELLO message. Therefore, the overhead introduces by Join-cluster message per time step for all nodes in the cluster is

$$Overhead_{Join-cluster} = f_{HELLO} * n \tag{6}$$

Thus, total overhead of initial clustering per time step for all nodes in the cluster is

$$\begin{aligned} Overhead_{initial\_clustering} &= f_{HELLO} * n + f_{HELLO} * n \\ &= 2f_{HELLO} * n \end{aligned} \tag{7}$$

Trusted Clustering: In order to become CH, the candidate node broadcasts a *Vote_me* message to the neighbors in one time step. After receiving *Vote_me* message, each neighbor unicasts a *Vote* message to the candidate node in one time step. In response of *Vote* message, the candidate node broadcasts a *Leader* message in one time step. The neighbor node upon receiving *Leader* message unicasts a *Leader-certificate* message to the candidate node in one time step. Thus, for all nodes in the cluster, they send $4n$ messages in four time steps. In other words, total overhead of trusted clustering per time step for all nodes in the cluster is

$$Overhead_{trusted\_clustering} = n \tag{8}$$

Cluster Maintenance: In order to maintain the cluster that is to detect the cluster split or merge, each node

periodically broadcasts a HELLO message per time step. Also, CH broadcasts a Merge-cluster message in one time step for clusters merging. Therefore, total overhead of cluster maintenance per time step for all nodes in the cluster is

$$Overhead_{cluster\_maintenance} = f_{HELLO} * n + n \tag{9}$$

Thus, total message overhead of the proposed scheme per time step for all nodes in the cluster is

$$\begin{aligned} Overhead &= 2f_{HELLO} * n + n + f_{HELLO} * n + n \\ &= 3f_{HELLO} * n + 3n \end{aligned} \tag{10}$$

As $f_{HELLO} = \Theta(1)$ [13], let $f_{HELLO} \leq C$, where C is a constant. Therefore,

$$\begin{aligned} Overhead &\leq 3Cn + 3n \\ &\leq (3C + 3)n \\ &= O(n) \end{aligned} \tag{11}$$

Total overhead of DSTCRP per time step for all n nodes in the cluster is $O(n)$. Therefore, total overhead of the proposed scheme per time step per node in the cluster is $O(1)$ and hence the Claim 1 is proved.

Claim 3: *The time complexity for CH election in trusted clustering of the proposed scheme is O(t), where t is 1-hop transmission delay*. *Proof:* In trusted clustering, the candidate node broadcasts a *Vote_me* message to its neighbors. On receiving *Vote_me* message, each neighbor unicasts a Vote message to the candidate node. The candidate node broadcasts a Leader message to its neighbors. In response to Leader message, the neighbors unicast a Leader-certificate. Therefore, total time complexity for CH election in trusted clustering of the proposed scheme is

$$T = O(t) + O(t) + O(t) + O(t) = O(t) \tag{12}$$

Hence the *Claim 3* is proved. Here it may be noted that other delays (such as queuing, processing, signature generation and verification) are neglected.

4.2 Security analysis

In this section, we show that the proposed model based on the trust evaluation scheme along with other security scheme/architecture can handle the following attacks. While a MN notices that another MN is misbehaving (i.e., drop or false inject packets), corresponding behavior count is decreased as well as the trust value. If the MN continues to behave maliciously, gradually its trust value reaches a lower limit and it is marked as *blacklisted* and isolated from the network. Further the weight-based trust calculation ensures lower impact by malicious node on resultant trust calculation (refer to Claim 7 in "Appendix"). Here we are categorically examining the protocol with different kind of attacks:

An attacker cannot spoof an address (that is IP and id) of a node without knowing the one-way secure hash function. In addition, it cannot generate victim node's signature as because an attacker may know the public key, but it is difficult to know the private key of the victim's node.

Routing table overflow and resource consumption attacks can be detected by our proposed trust model. As MN promiscuously overhears the traffic for each neighbor MN, it easily finds false injection of RREQ to the network. Thus any compromised or malicious attacker generating these attacks can be identified.

If a node finds another neighbor MN sends packet to a particular MN repeatedly or any packet coming to a node is not forwarded properly within a certain interval (say α_t), it informs the corresponding CH. CH generates a *Warning* message to the particular misbehaved node and temporarily restrict other good nodes to communicate with the suspicious node until the next review comes. It also asks other MN to monitor the suspicious node. A second chance is given to the suspicious node to revoke its status by behaving properly. If the node fails in the next review CH will blacklist the node and isolate it from the network. Thus the trust model can detect Byzantine attack and blackhole attack as well.

If a node suspects that any neighbor MN unnecessarily holds any packet for a longer time, it calculates the packet delivery time and reports to the CH. For a given network traffic load a predefined allowable delay is also added to predict the packet delivery time from each node (say δ). If the CH finds packet delivery of the node is longer than δ, CH generates an *Alert* to that node and asks MN to monitor its further behavior. All MN is 1-hop from the CH and intra and inter-cluster communication are initiated by the CH only. Any node tries to communicate with any distant node repeatedly in a certain interval of time; a chance of wormhole may arise. In such a case, CH informs neighbor trusted MNs to monitor the suspected node. CH sends *Alert* beacon to the particular misbehaved node. If the said node continues to behave this way, the CH detects the node as malicious and isolates from the network. So using our model wormhole attack can be detected.

If the CH finds an MN asks for some information repeatedly in a small interval of time, the CH checks whether it is a fraud request, it generates an *Alert* message and blacklists the MN in the next review, i.e., any request from that node would not be entertained. Thus sleep deprivation attack may be identified and mitigated.

CH generates, stores and propagates unique node-ID to its legitimate MN and initiates all intra and inter-cluster communications. Any legitimate MN request for any communication includes the unique MN-node-ID to CH. Therefore, for any outsider malicious node, it is difficult to get the

legitimate node-ID and initiate any intra or inter-cluster communication. This prevents the impersonation attacks.

In the proposed scheme, nodes need some time for bootstrapping mechanism (including start up and initial clustering) to become a valid member of a cluster. This bootstrapping time is vulnerable time for our protocol. As trust is calculated depending upon real time data over some parameters, a certain round of communications is needed (number of rounds depends upon the application and deployment scenario) before the trust calculation and generation procedure is started. Thus the proposed scheme requires vulnerable time to detect the above malicious activities.

4.3 Comparison with existing protocols

A survey on existing trust management schemes along with our proposed scheme for WANET are given in Table 2. Description of different trust management schemes are adapted from [14] to compare our proposed scheme.

5 Simulation

We have compared our proposed protocol with ECS [15], CBRP [16] and CBTRP [17] using NS-2 (version-2.34) simulator [18] on Fedora Core 9 platform. *Clustering latency*, *communication overhead*, *packet delivery ratio* and *throughput* are chosen as the performance metrics and results are presented in this section. *Clustering latency* refers to the total time needed to form a cluster, *communication overhead* refers to the average number of message exchange per node to form and maintain a cluster (including Hello messages of AODV).

Efficient clustering scheme (ECS) [15] is suitable for cluster formation and cluster maintenance in a large and dense mobile ad hoc network. It eliminates the frozen period requirement for cluster formation, reduces cluster overlapping and prolongs the cluster lifetime without producing excessive clustering overheads.

In cluster-based routing protocol (CBRP) [16], the nodes are divided into several disjoint or overlapping clusters. Each cluster elects one node as the so-called clusterhead for operating the routing process. A gateway is a node that has two or more clusterheads as its neighbors or when the clusters are disjoint, at least one clusterhead and another gateway node. The routing process itself is performed as source routing by flooding the network with a route request message. The CBRP uses a variation of the lowest-ID algorithm which is an identifier-based algorithm. If the network size is small (about 150 nodes) CBRP can be a good routing solution. In cluster-based trust-aware routing protocol (CBTRP) [17], the network is organized into one-hop disjoint clusters, whereby every node elects the most qualified and trustworthy node of its 1-hop neighbors to be its clusterhead. Each node monitors the behavior of other nodes to ensure the passage of packets through trusted routes only. Once a malicious node is detected by CBTRP, the node is isolated from the network so that no packet is forwarded through or from it.

Table 1 Simulation parameters and environment

Simulation parameter		Assigned value
Application agent		CBR
Packet size		512 bytes
Transport agent		UDP
Routing protocol		AODV
Addressing scheme		IDDIP
Mobility		0–5 m/s
Pause time		5 s
Simulation time		300 s
Mobility model		Random way-point
Total no. of nodes	Network area	No. of malicious nodes
11	330 × 330	3
16	370 × 370	4
21	440 × 440	5
25	470 × 470	6

5.1 Simulation parameters

In our simulation, the radio transmission range of nodes is 250 m and the routing protocol used is AODV. The *Hello* interval of AODV is set to 1s. UDP is used as the transport layer protocol with Constant Bit Rate (CBR) traffic generation of packets of size 512 bytes. A node joins the WANET every 5 s using IDDIP scheme [19]. The random way point mobility model is adopted, wherein node speeds are randomly distributed between 0 and 5 m/s and on reaching the destination the pause time of a node is set to 5 s. The propagation model used is two ray ground and total simulation time is set to 300 s for each set of simulations. Malicious nodes are selected at random and falsely inject, drop, or modify all messages that they are to forward. The simulation is done for different number of nodes present in a cluster. The network area, total number of nodes, and the number of malicious nodes in a cluster are chosen according to following Table 1 (Table 2).

5.2 Simulation results and observations

5.2.1 Communication overhead and clustering latency

Figure 2a shows the effect on communication overhead per node with number of nodes increase in a cluster for the

Table 2 Survey on existing trust management schemes along with proposed scheme

Author/Refs. no	Objectives	Methods	Attacks handled	Performance evaluation	Trust property
Yan and Prehofer [20]	Trust prediction, evaluation	Direct observation reputation recommendation	N/A	The system may not give good result when correlation co-eff is less between observed samples	Kalman filter is used to predict trust. Trust is a continuous value
Pirzada and McDonald [21]	Authentication	Direct observation	Packet modification, fabrication impersonation	N/A	Extension of DSR and Marsh's model dynamic
Pisinou et al. [22]	Secure routing	Direct observation	Black hole, route injection, selfish nodes	Route overhead no.of routes route error	Extension of AODV trust embedded, dynamic
Ghosh et al. [23]	Secure routing	Direct observation reputation	Black hole, gray hole false accusation	Route overhead route error no.of routes	Extension of AODV trust embedded, incentive mechanism
Virendra et al. [24]	Key management	Direct indirect observation	False key generation, compromise of others' keys	N/A	Physical–logical trust domain, dynamic subjectivity
Theodorakopoulos and Baras [25]	Trust evaluation	Direct/ observation recommendation	Impersonation false accusation	Confidence level opinion about others	Trust model using Semiring theory
Balakrishnan et al. [1]	Secure routing	Direct/ observation reputation	Packet dropping DoS(flooding) routing information modification false accusation false recommendation	Packet delivery ratio latency	Fellowship model using DSR
Boukerche and Ren [26]	Reputation evaluation	Direct/ observation reputation	General misbehaving nodes	Overhead for security, query % of packets, % malicious nodes	Generalized reputation evaluation prototype group-based trust model
Cho et al. [27]	Battlefield trust evaluation under no-prior interaction	Direct/indirect observation recommendation	General misbehaving nodes	Trust level	Quantitative model using hierarchical stochastic petri-nets
Wang et al. [3]	Trust prediction, evaluation	Direct observation reputation	Black hole DoS	The system may not give good result when correlation co-eff is less between observed samples	Personal-trusted-bubble, subjective
Proposed protocol	Secure clustering routing	Trust evaluation in a collaborative manner	Address spoofing On-Off, DoS blackhole, gray hole Byzantine,rushing Wormhole impersonation Resource consumption Man-in-the-middle location disclosure	Clustering latency, energy, communication overhead for each MN	Trust, is measured in [0,1], dynamic, distributed fashion, is predicted by self and recommendation evidences

both schemes under consideration. It can be seen that as the number of nodes increase in the cluster, communication overhead per node also increases for both the approaches. This is because when the number of nodes in the cluster is less, number of message exchanges during cluster formation and maintenance is also less for both schemes. Effect of increased node number in a cluster on clustering latency for ECS and our proposed scheme is shown in Fig. 2b. As the number of nodes increase in the cluster, clustering latency is also increased for both approaches. This is because clustering latency depends on round trip delay and number of message exchange between nodes during cluster formation. Number of messages exchange and round trip delay will be more as the number of nodes increases in the

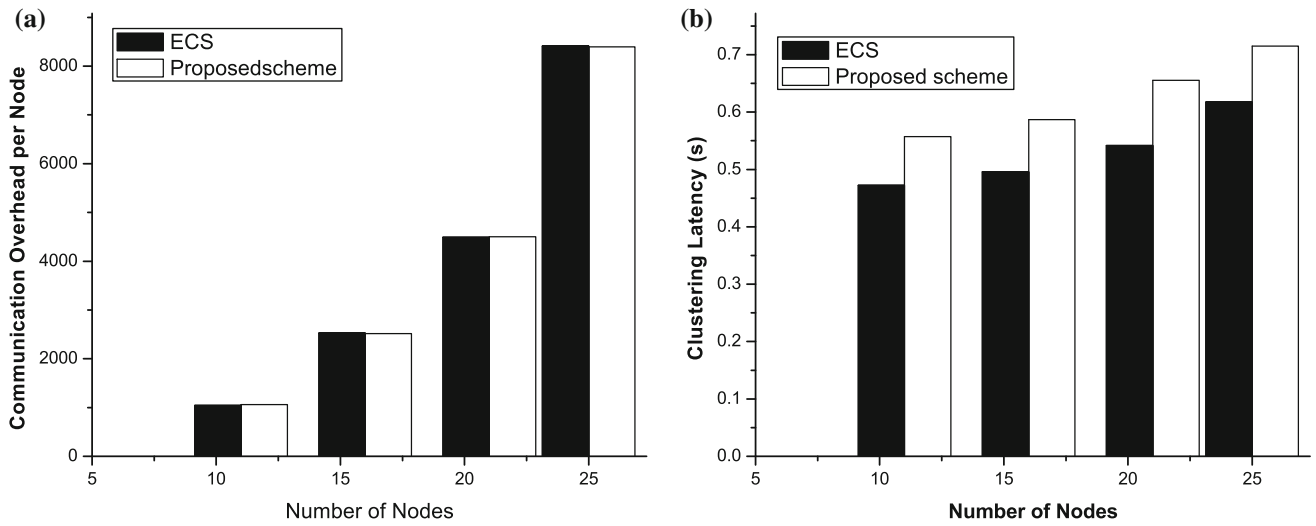

Fig. 2 Communication overhead (**a**) and clustering latency (**b**) versus number of nodes

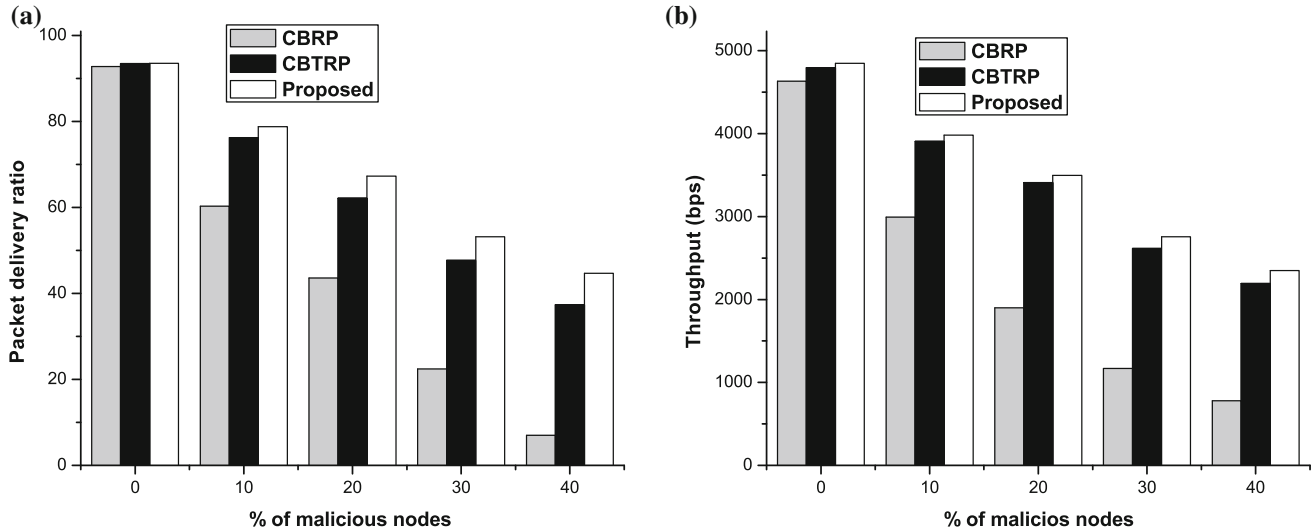

Fig. 3 Packet delivery ratio (**a**) and throughput (**b**) versus number of malicious nodes

cluster. It may be noted that even with added security mechanisms our proposed scheme has little higher clustering latency and almost similar clustering overhead with ECS.

5.2.2 *Packet delivery ratio and throughput*

Figure 3 shows the impact of malicious node behavior on average packet delivery ratio for CBRP, CBTRP and our proposed protocol. All the protocols give around 93 % packet delivery ratio when none of the nodes in the network are malicious. However, as the number of malicious nodes increase in the network, packet delivery ratio decreases for them. In case of CBRP, as number of malicious node increases, packet delivery ratio decreases significantly around 8 % when 40 % of the nodes are malicious. This is because no security mechanism is incorporated with CBRP. As a result, malicious nodes can potentially manipulate CBRP to make routes pass through themselves and drops the data packet instead of forwarding. On the other hand, the proposed protocol gives near about 45 % packet delivery ratio in the presence of 40 % malicious nodes in the cluster. This is because the proposed protocol verifies the authenticity of the node while setting up the route. It can also be seen from the figure that the proposed protocol performs better than CBTRP. This due to the fact that the proposed protocol predicts the trust (i.e., resultant trust) of a node in a collaboration of all other nodes in a cluster, whereas CBTRP calculates only the direct trust of the node. Hence our protocol detects the

malicious node more accurate and faster than CBTRP. Similarly throughput is reduced with increment of malicious nodes in the network for all the protocols under consideration. From Fig. 3, we can observe that all the protocols give good throughput when none of the nodes are malicious. However, as number of malicious nodes increase, the performance of CBRP degrades significantly. On the other hand, the proposed protocol and CBTRP give steady throughput in presence of malicious node. In presence of 40 % malicious nodes our proposed protocol gives throughput around 2,400 bps whereas in CBRP and CBTRP the throughput becomes around 785 bps (almost one-third of our protocol) and 2,194 bps respectively.

6 Conclusions

In the paper, we have proposed a novel distributed secure trust aware clustering protocol that provides reliable end-to-end delivery of data for WANETs. The proposed trust model uses a localized trust management strategy to compute trust and hence eliminates the network-wide flooding. In addition, the proposed clustering protocol divides the network into several one-hop disjoint clusters and elects the most qualified and trustworthy node as a CH for each cluster. Based on the trust model a secure routing protocol is devised to ensure the secure end-to-end data delivery. Analysis of the protocol shows that the present protocol efficiently forms and maintains clusters with less communication overhead and low latency compared to similar existing protocols. Moreover, it reduces the security threats present in WANET with a minimal use of cryptographic functions. Simulation results show that proposed protocol offers higher throughput and packet delivery ratio compared to the popular CBRP and CBTRP in presence of malicious nodes with low communication overhead.

Appendix: Proof of accuracy of the proposed trust model

In this section we present some important features as well as necessary and sufficient properties that a distributed trust model must possess. In addition to that we show our proposed trust model holds and satisfies all these conditions and performs well in different networking scenarios.

Claim 1 *Each legitimate CH must eventually hold the replica of latest trust value of its MNs.*

Proof In our proposed scheme, each MN sends its public key to CH to join in the network at the time of initialization. Trust calculation and propagation is taken care by the CH itself and this trust information is sent periodically. Thus, the trust value of each node is periodically updated which in turn determines the reputation of the node. But due to node failure or mobility or reorganization of clusters, MNs may not get trust information of the CH. All living and legitimate CHs of the network hold the most recently updated trust information about its MNs to continue uninterrupted service. Within each cluster, CH maintains the trust table for all its MN and records all the updates accordingly. This eventually ensures propagation of correct trust information between the CHs. If any good node fails to communicate with its CH, and after some time it revives the connection, the trust value would be automatically assigned to that node from previous record.

Claim 2 *Trust status of a node is uniform/same throughout the cluster and network at any particular time.*

Proof In our proposed scheme, at the time of trust generation phase each MN, GN and CH monitors the traffic of its neighbor nodes choosing some packets in a random interval of time and calculates the direct trust. CH accumulates all trust information from the neighbors of review node. Then CH combines all recommendation and finally determines the resultant trust. Therefore, CH of a cluster aggregates the trust information and finally decides and propagates the trust status throughout the network. Trust of a node is uniform throughout the network, i.e., all nodes have to trust a node by the degree decided by the CH.

Claim 3 *The reputation of node can be directly acquired by the any other node without network wide flooding.*

Proof The direct trust of a node M is calculated using the local information and the local behavior monitoring of any node in the radio range. The periodic trust calculation and propagation procedure (by the CHs) ensure that the reputation of any member node can be easily detected by checking the trust table (stored in the CH). Any other member nodes while trying to communicate or get information about its neighbor (in the same cluster) or any other nodes in the network, it has to query to the associated CH only. This makes the distributed trust computation procedure flexible and efficient and thus completes the proof.

Claim 4 *A node's locally stored trust value is readable to the CH but confidential to any node including the node itself.*

Proof Our scheme holds this property. This is one of the important security measures. Any node cannot have access on its own trust value. After the final calculation of resultant trust CH securely propagates the *Trust_Cert* and *Available-Neighbor list* to its MNs digitally signing the message. The trust value is stored in CH only. So there is no way for any malicious node to advertise itself as *trusted* and disrupt the network functionality.

Claim 5 *A node's unauthorized modification to its local trust and reputation information can be identified and prevented.*

Proof This claim can be corroborated using *Claim 4*. A malicious node cannot join to two different clusters simultaneously. A node is permitted to join not more than one CH at a time. Suppose, a node tries to join in two clusters simultaneously, then it would get two different IP and ID from two different CHs. If this type of maliciousness cannot be detected, secret information may easily be leaked between two clusters. Our protocol guarantees detection of such malicious activity. If any GW finds a node initiates different communications with different ID or IP address, GW reports to the CH about the malicious node and subsequently the node is removed. If such node exists, GW detects, and block this node; restores previous local trust value. Thus the unauthorized modification to local trust and reputation are identified and can be prevented.

Claim 6 *The model ensures better accuracy in trust calculation and its secure propagation.*

Proof The proposed model does not rely only on the local monitoring method or observation of any individual node. Firstly, direct trust of a node N, is calculated by all neighboring nodes depending on the behavior of N. Here, the behaviors (both good and bad) of N is parameterized by some metrics and corresponding weight. Once, direct trust are calculated by each neighboring node, these trust values are combined (based on appropriate weights) to formulate the resultant trust values using the weighted DS-theory presented in the Eq. 2. The uniqueness/major strength of this weighted trust calculation scheme is in combining the evidences from different nodes with uneven (unequal) priority distribution. Thus neighboring nodes of N modify the trust values depending on weights, which are again computed depending on environment parameters. This ensures more accuracy in the proposed trust calculation scheme. On the other hand, after final trust calculation for the member nodes of any cluster, only a digitally signed trust certificate is propagated. Thus, the proof completes.

Claim 7 *Weight based trust calculation scheme ensures lower impact by malicious node on trust calculation.*

Proof In the proposed protocol, Clusterhead (CH) calculates direct trust of a node by directly monitoring the node. Subsequently it collects the trust evidences (i.e., Recommendation Trusts) from different neighbor nodes of the node under review. Finally, in order to calculate the resultant trust of the node under review, CH combines its direct trust and all the recommendation trusts using proposed DS-theory of combining evidences. In traditional DS-theory of combination, as all beliefs are given equal priority, the impact of recommendation from a malicious node which reports its belief about another node under review may be higher. As a result of this, a good node may be treated as bad one. However in our proposed proposal, we have eliminated such ambiguity. Here may be two cases: in the first case, a node is already detected as malicious (blacklisted) and in the second case, the node behaves maliciously but not marked as blacklisted. In the first case, the proposed protocol does not consider the recommendation from malicious nodes while CH combines the trust evidences. In the second case, the proposed weight-based DS-theory assigns weight to the evidences from different nodes before combining them. This weight is determined by the past behavior of the recommender node depending with environment parameters. The highest weight (say 50 %) is assigned to the belief generated by the CH itself to quarantine the impact of the malicious node. If the recommender node has higher trust value, it is less likely to report with wrong recommendation. So, our trust calculation scheme guarantees lower impact of evidence by any malicious node on the final trust of a node under review.

References

1. Balakrishnnan, V., Varadharajan, V., Tupakula, U. K., & Lucs, P. (2007). Trust and recommendations in mobile ad hoc networks. in *Proceedings of 10th IEEE international conference on networking and services* (pp. 64–69).
2. Chatterjee, P., Sengupta, I., & Ghosh, S. (2012). Stacrp: A secure trust based auction oriented clustering routing protocol for securing mobile ad hoc networks. *Cluster Computing, 15*, 303–320.
3. Wang, X., Liu, L., & Su, J. (2012). RLM: A general model for trust representation and aggregation. *IEEE Transactions on Services Computing, 5*(1), 131–143.
4. Chatterjee, M., Das, S. K., & Turgut, D. (2000, November). An on-demand weighted clustering algorithm WCA for ad hoc networks. in *Proceedings of IEEE GLOBECOM 2000* (pp. 1697–1701).
5. Malpani, N., Welch, J., & Vaidya N. (2000, August). Leader election algorithms for mobile ad hoc networks. in *Proceedings of 4th international workshop on discrete algorithms and methods for mobile computing and communications* (pp. 96–103).
6. Vasudevan, S., Decleene, B., Immerman, N., Kurose, J., & Towsley, D. (2003). Leader election algorithms for wireless ad hoc networks. in *Proceedings of DARPA information survivability conference and exposition* (pp. 261–272).
7. Yu, Y., Wang, J., Song, M., & Song, J. (2010). Network traffic prediction and result analysis based on seasonal ARIMA and correlation coefficient. in *Proceedings of the 2010 international conference on intelligent system design and engineering application, ISDEA'10* (Vol. 01, pp. 980–983).
8. Stoica, P., Friedlander, B., & Soderstrom, T. (1986). Least-squares, yule-walker, and overdetermined yule-walker estimation of ar parameters: A monte carlo analys is of finite-sample properties. *International Journal of Control, 43*, 13–27.

9. Akaike, H. (1974). A new look at statistical model identification. *IEEE Transactions on Automatic Control, 19*, 716–723.
10. Dempster, A. P. (1968). A generalization of bayesian interface. *Journal of Royal Statistical Society, 30*, 205–447.
11. Shieh, S.-P., Lin, C.-T. Yang, W.-B., & Sun, H.-M. (2000). Digital multisignature schemes for authenticating delegates in mobile code systems. *IEEE Transactions on Vehicular Technology, 49*(4), 1464–1473.
12. Bettstetter, C., & Konig, S. (2002). On the message and time complexity of a distributed mobility-adaptive clustering algorithm in wireless ad hoc networks. in *Proceeding of the 4th European wireless* (pp. 128–134).
13. Sucec, J., & Marsic, I. (2001). Hierarchical routing overhead in mobile ad hoc netorks. *IEEE Transactions on Mobile Computing, 3*, 45–56.
14. Cho, J.-H., Swami, A., & Chen, I.-R. (2012). Modeling and analysis of trust management with trust chain optimization in mobile ad hoc networks. *Journal of Network and Computer Applications*, *35*(3), 1001–1012.
15. Yu, J. Y., & Chong, P. H. J. (2006). An efficient clustering scheme for large and dense mobile ad hoc networks (manets). *Computer Communications, 30*, 5–16.
16. Chiang, C. -C., Wu, H. -K., Liu, W., & Gerla, M. (1997, April). Routing in clustered multihop, mobile wireless networks with fading channel. in *IEEE Singapore international conference on networks*, SICON'97, (pp. 197–211).
17. Safa, H., Artail, H., Tabet, D. (2010). A cluster-based trust-aware routing protocol for mobile ad hoc networks. *Wireless Networks, 16*, 969–984.
18. Fall, K., & Varadhan, K. ns Manual. isi.edu/nsnam/ns/doc.
19. Ghosh, U., & Datta, R. (2011). A secure dynamic IP configuration scheme for mobile ad hoc networks. *Ad Hoc Networks, 9*(7), 1327–1342.
20. Yan, Z., & Prehofer, C. (2011). Autonomic trust management for a component-based software system. *IEEE Transactions on Dependable and Secure Computing, 8*(6), 810–823.
21. Pirzada, A. A., & McDonald, C. (2004). Establishing trust in pure ad-hoc networks. In *Proceedings of the 27th Australasian Conference on Computer Science (ACSC '04)* (Vol. 26, pp. 47–54), Dunedin, New Zealand.
22. Ghosh, T., Pissinou, N., & Makki, K. (2004, May). Collaborative trust-based routing in multi-hop ad hoc networks. in *Proceedings of 3rd international IFIP-TC06 networking conference*, Lecture Notes in Computer Science (pp. 1446–1451).
23. Ghosh, T., Pissinou, N., & Makki, K. (2005). Towards designing a trust routing solution in mobile ad hoc networks. *Mobile Networks and Applications, 10*, 985–995.
24. Virendra, M., Jadliwala, M., Chandrasekaran, M., & Upadhyaya, S. (2005). Quantifying trust in mobile ad hoc networks. In *Proceedings of IEEE KIMAS 2005* (pp. 65–71).
25. Theodorakopoulos, G., & Baras, J. S. (2006). On trust models and trust evaluation metrics for ad hoc networks. *IEEE Journal on Selected Areas in Communications, 24*(2), 318–328.
26. Boukerche, A., & Ren, Y. (2008). A security management scheme using a novel computational reputation model for wireless and mobile ad hoc networks. In *Proceedings of the 5th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN '08)* (pp. 88–95). New York, NY: ACM.
27. Cho, J.-H., Swami, A., & Chen, I. R. (2012). Modeling and analysis of trust management with trust chain optimization in mobile ad hoc networks. *Journal of Network and Computer Applications, 35*(3), 1001–1012.

Pushpita Chatterjee has received her M.Sc. and M.Tech. degrees in Computer Science and Engineering from the University of Calcutta in 2002 and 2004 respectively. She completed her Ph.D. in School of Information Technology from Indian Institute of Technology Kharagpur, India in 2012. Dr. Chatterjee is currently working as a research lead at SRM-RI, Bangalore. She has over 15 publications to her credit in international journals and conferences. Her research interests are mobile computing, distributed and trust computing, wireless ad hoc and sensor networks, information-centric networking and software-defined networking.

Uttam Ghosh received his B.Tech. in Information Technology from Govt. College of Engineering and Textile Technology, Serampore, WB, India in 2005. He completed his M.S. (by Research) and Ph.D. both from the Department of Electronics and Electrical Engineering, Indian Institute of Technology Kharagpur, India in 2009 and 2013 respectively. Dr. Ghosh has a good number of publications in international journals and Conferences. His main research interests include Computer Networks, Security and Cryptography, Distributed and Trust Computing, Wireless Networks, Mobile Ad-hoc Networks, Information-Centric Networking and Software-Defined Networking.

Indranil Sengupta has obtained his B.Tech, M.Tech. and Ph.D. degrees in Computer Science and Engineering from the University of Calcutta in 1983, 1985, and 1990, respectively. He joined Indian Institute of Technology, Kharagpur, as a faculty member in 1988, in the Department of Computer Science and Engineering, where he is presently a Professor at the Department. Dr. Sengupta is former head of this department and also was heading the School of IT of the Institute. He has over 24 years of teaching and research experience, and over 100 publications to his credit in international journals and conferences. A Centre of Excellence in Information Assurance has been setup at IIT Kharagpur under his leadership, where a number of security related projects are presently being executed. His research interests include cryptography and network security, side-channel attacks on cryptosystems, VLSI design and testing, and mobile computing. He is a member of IEEE.

Soumya K. Ghosh did his M.Tech. and Ph.D. in Computer Science and Engineering from the Indian Institute of Technology (IIT) Kharagpur, India. He is currently an Associate Professor at the School of Information Technology, IIT Kharagpur. Before joining IIT Kharagpur, Dr. Ghosh worked for Indian Space Research Organization in the area of Satellite Remote Sensing and GIS. His research interests include Network Security and Spatial Web Services. He has over 70 research papers in reputed journals and conference proceedings. He is a member of IEEE.