

SWCA: A Secure Weighted Clustering Algorithm in Wireless Ad Hoc Networks

Yingpei Zeng^{†‡} Jiannong Cao[‡] Shanqing Guo^{*} Kai Yang[†] Li Xie[†]

[†]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, P.R. China

[‡]Department of Computing, Hong Kong Polytechnic University, Hong Kong

^{*}School of Computer Science and Technology, Shandong University, Jinan, P.R. China

zyp@dislab.nju.edu.cn csjcao@comp.polyu.edu.hk guoshanqing@sdu.edu.cn yang_kai@live.com xieli@nju.edu.cn

Abstract—Clustering has been widely used in wireless ad hoc networks for various purposes such as routing, broadcasting and QoS. Many clustering algorithms have been proposed. However, most of them implicitly assume that nodes behave honestly in the clustering process. In practice, there might be some malicious nodes trying to manipulate the clustering process to make them serve as clusterheads, which can obtain some special power, e.g., eavesdropping more messages.

In this paper we present a Secure Weighted Clustering Algorithm (SWCA). SWCA uses the Weighted Clustering Algorithm (WCA) for clustering and TESLA for efficiently authenticating packets. We propose a novel neighbor verification scheme to check whether the values of election-related features (e.g., *node degree*) are forged by malicious nodes. Also, we theoretically analyze the probability for a malicious node to tamper *node degree* without being detected and derive a lower bound on the probability. Finally, simulation results show that SWCA is secure but still has comparable performance with WCA. To the best of our knowledge, SWCA is the first algorithm considering the security of *1-hop* type clustering (in this type only the *clusterhead* can communicate with ordinary members directly) in ad hoc networks.

I. INTRODUCTION

Clustering is an important technique in ad hoc networks. Originally used to address dynamic topology and transmission conflicts when it was first proposed by Baker et al. [1], clustering now has extended its application into many fields such as routing [2], [3], broadcasting [4], QoS [5], services discovery [6] and security [7], [8], etc. Dozens of clustering algorithms have been proposed in the past more than twenty years [2], [3], [5], [9]–[14].

One problem in previous clustering algorithms is that almost all of them lack security considerations, which may result in severe security consequences. For example, in DCA [12] a malicious node can become a clusterhead by simply broadcasting a “CH” (clusterhead) message. Consequently, if DCA is used to elect clusterheads in a cluster-based routing protocol, a malicious node can easily eavesdrop all the messages routed through it (more messages than simply changing its wireless card into promiscuous model). The cases will be even worse when clustering is used in security mechanisms. For example, attackers will know the keys of all the nodes if there are enough attackers becoming *clusterheads* [8].

Currently several secure clustering algorithms have been proposed [7], [15]–[17]; however they either focus on special cluster type (i.e., clique), or are not applicable in *ad*

hoc networks. In this paper, we propose a secure clustering algorithm called SWCA, which is based on WCA [9]. SWCA uses an efficient symmetric cryptography protocol TESLA [18] to authenticate messages. In order to prevent malicious nodes from increasing their election priorities, the values of all the election-related features are verified by a novel neighbor verification scheme. Also, we theoretically derive a lower bound on the probability for a malicious node to tamper *node degree* without being detected. Finally we evaluate the performance of SWCA by simulation.

The rest of the paper is organized as follows. In Section II we review related work. In Section III, we describe the attacker model and assumptions. In Section IV, we present the design of SWCA. Section V and Section VI give theoretical analysis and simulation results of SWCA respectively. Finally we conclude this paper in Section VII.

II. RELATED WORK

Currently, many clustering algorithms have been proposed in the literature. We can roughly classify them into three types based on the radiuses of clusters they produce: (i) *clique* [3], [7], in which cluster members can communicate with each other directly, (ii) *1-hop* [2], [5], [9], [11], [12], in which only the *clusterhead* can communicate with ordinary members (citizen nodes) directly, (iii) *k-hop* ($k \geq 1$ and k is adjustable) [13], [14], in which the *clusterhead* is within k hop to its ordinary members. Among them the *1-hop* type may be the most natural and also the most widely used one. *1-hop* type cluster can be transferred to another kind of backbone called Connected Dominating Set (CDS) with addition steps, which is shown with concrete example in a recent work [19]. Our algorithm is based on WCA [9], which is a *1-hop* type algorithm and is weight based. Weight based means clusterhead is selected based on the weight of node (weight may be computed from any feature alone or several features combined). Comparing with other single-feature based (e.g., *nodeId* or *degree* based algorithms [2]), weight based clustering algorithms (e.g., WCA [9], DCA [12]) are believed to be more flexible [9].

There are several research papers on the security of clustering. In [15], Vasudevan et al. proposed two secure leader election algorithms, but both of them require nodes to honestly perform priority evaluation function. In [7], Huang et al.

proposed a clusterhead computation protocol to ensure *fair election* in cliques. Our algorithm is different from theirs because we don't emphasize on the *fairness*; we consider the natural difference among electors (such as the node degree). Sun et al. also proposed a secure distributed clique formation protocol in [16]. It can make normal nodes form mutually disjoint cliques when there are attackers, and the view of cliques is consistent to all normal members. Another related work targeting sensor networks is presented by Liu et al. [17], in which three techniques are proposed for resilient cluster formation. However these techniques cannot be applied directly to ad hoc networks, because they need a central control, e.g., tagging partial nodes as pre-determined clusterheads and routing packets to the *base station*.

III. ATTACKER MODEL AND ASSUMPTIONS

A. Attacker Model

An attacker may alter the contents of packets, or inject fabricated packets into the network. Because in ad hoc networks, packets may pass through multi-hop before arriving target node, attackers in the middle of the route may change the values of critical fields to manipulate the election. Also, a malicious node may masquerade as another node to send fabricated packets.

An attacker may fabricate the values of election-related features to increase its election priority. An example is the fabrication of the node degree: an attacker forges neighbors which don't really exist to increase its priority. The attacker can success because in clustering algorithms, node degree is usually an important feature [2], [9], and each node usually computes node degree and other factors by itself [2], [9], [12], then the attacker can easily forge a neighbor to increase its node degree, resulting a more favorable combined weight and higher election priority.

There may be single-node attacks and collusion attacks in the network. In the former case, nodes carry out attacks independently. In the later case, information may be exchanged between colluding attackers, so they are able to claim fabricated but consistent-alike election-related information.

B. Assumptions

We assume that nodes are equipped with omnidirectional antennas. Also we assume R and r are the maximal and minimal transmission ranges in the network respectively; one node can hear another node if they are within distance r and cannot hear another node if they are outside distance R . This is a reasonable assumption because in [20], Guha et al. found their radios $r:R=121\text{cm}:183\text{cm}\approx 0.66$ in most cases through experiments. In Section VI-B we turn to the quasi unit disk graph model (Quasi-UDG) [21] in simulation.

We also assume that each node has been assigned a unique identity with corresponding public/private key pairs, which can be completed by many proposed methods (e.g., [22], [23]). All nodes can move randomly in the network; we assume that nodes in the network have an approach to know their

locations (e.g., using GPS) and a location verification scheme is employed, such as [24], [25].

In order to use TESLA, we inherit all the assumptions of TESLA [18]. For example, all nodes have loosely synchronized clocks (maximal error synchronization is Δ), and each node in the network should be aware of the time interval of TESLA. TESLA is also used in other protocols of ad hoc networks, e.g., [26].

IV. SECURE WEIGHTED CLUSTERING ALGORITHM (SWCA)

A. Overview

If we can prevent attackers from altering the contents of clustering related packets, and constrain all the nodes to send their election-related information honestly, then the result of clustering will be correct. This is also the way SWCA follows. Specifically, SWCA uses efficient symmetric cryptography TESLA [18] to prevent attackers from modifying packets, and uses neighbor verification to make sure that the values of election-related features are correct. These election-related features include *node degree*, *distances with neighbors*, *speed* (which is representing the *mobility* of a node [9]) and *cumulative clusterhead time* (which is representing the inverse of the *battery power* of a node [9]).

B. SWCA Cluster Formation

In SWCA, Nodes periodically broadcast *hello packets* to neighbors. *Hello packets* contain *nodeIds* and *locations* of nodes and employ TESLA to authenticate the sources. Location can be acquired from equipped GPS or localization algorithms [24].

Before clustering, nodes need to verify neighbors' locations. According to our assumption, the verification is carried out by a location verification scheme e.g., VM [25]. If a node want to be a candidate for clusterhead, its location must have been verified. The correctness of nodes' locations is important; it is used in the verification of *distances with neighbors*.

Step 1: When the clustering starts, each node broadcasts its election-related information in a packet to neighbors within 2 hops (we call them within-2-hop neighbors). The packet (we call it *info packet*) contains information used to compute the *combined weight*, and these information will be verified later by neighbors. Format of this packet ¹ is shown in Fig.1. The *pkId* is an identifier that has not been used recently in transmission. The *location* is the verified location of node. The *neighborList* is a list of neighbor information; each entry of the list includes *nodeId* and *location* of a neighbor. *speed* and *CCHtime* are the speed and cumulative clusterhead time of the node, respectively. *Info packet* only propagates 2 hops, which can be realized by setting the TTL (time-to-live) value of IP packet to 2.

¹The whole packet is usually in $\langle M_j, i, K_{i-d}, MAC(K'_i, M_j) \rangle$ form, where M_j is the payload, the rest parts compose a TESLA block: i is the current TESLA time interval, d is the chosen key disclosure delay, the key K_{i-d} is the disclosed key to be used to authenticate foregoing packets, and K'_i is derived from the current key. MAC is message authenticating code.

Node:	$M_{info}=(\text{INFOPKT}, \text{pktId}, \text{nodeId}, \text{location}, \text{neighborList}, \text{speed}, \text{CCHtime})$
Node→	$info \text{ packet}=(M_{info}, \text{tesla})$ to 2 hops
Node:	$M_{weight}=(\text{WEIGHTPKT}, \text{pktId}, W_v, \text{neighborIdList})$
	$W_v = w_1\Delta_v + w_2D_v + w_3M_v + w_4P_v$
Node→	$weight \text{ packet}=(M_{weight}, \text{tesla})$ to whole network
Node:	$M_{authen}=(\text{AUTHPKT}, \text{pktId})$
Node→	(optional) $authen \text{ packet}=(M_{authen}, \text{tesla})$ to whole network

Fig. 1. Packets Propagation in SWCA

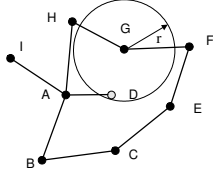


Fig. 2. Verification of neighbors' node degrees.

Step 2: Nodes calculate their *combined weights*, and broadcast weights to the whole network through *weight packets*. This step is similar with WCA [9]. As shown in Fig.1, the Δ_v, D_v, M_v, P_v are the corresponding weights of the four features mentioned before (we assume that the value of a node of Δ_v, D_v, M_v, P_v will be smaller if it has corresponding more *node degree*, less *distances with neighbors*, less *speed* and less *cumulative clusterhead time*). The w_1, w_2, w_3 and w_4 are corresponding *weighing factors*. The *neighborIdList* is a list of neighbors' *nodeIds*, which is used to gain the topology to compute global minima [9]. The broadcast can be done by broadcasting a single packet to network, or by exchanging weight information with neighbors as in [9]. In our simulation we use the first method. *Weight packets* can be sent immediately after *info packet*, and the TESLA key used to authenticate its *info packet* may also be used to generate MAC for this packet.

Step 3: When the key disclosure time arrives, each node broadcasts a key to authenticate the key used in the last interval. This key is needed by other nodes to verify its previous packets (i.e., *info packet*, *weight packet*); if key K_i was used to authenticate the last sent packet, we have to send a *disclosure key* K_x which is successive or equal to key K_i ($x \geq i$) in the key chain. We call the packet containing the key *authen packet*. However, if the TESLA key chain is shared by other applications of the same node, then this packet *may be canceled* if another application has broadcasted packets in this time interval.

Step 4: Nodes verify their neighbors' election-related information and their *combined weights*. Since nodes have received and authenticated the *info packets* and *weight packets* of neighbors within 2 hops, now they will verify the values of the four features and the *combined weights* of their neighbors. If a node find some misbehaving node, it will report that node to others by broadcasting the error to network. The processing of error reports will be described in Section IV-C. Next we will discuss how to verify node degree and other features.

Algorithm 1 Verification of neighbors' node degrees.

Input: *locations* and *neighborLists* of within-2-hop neighbors and itself.

```

1:  $Vset \leftarrow$  1-hop neighbor nodes, itself
2: for all each 1-hop neighbor node  $N_i$  do
3:   if itself not in  $N_i$ 's neighborList then
4:     report error, continue
5:   for all each  $N_i$ 's neighbor node  $N_{ij}$  do
6:     if  $N_i$  is not in  $N_{ij}$ 's neighborList then
7:       report error, continue
8:      $ND = \text{dist}(N_i, N_{ij})$ 
9:     if  $ND > R$  then
10:      report error, continue
11:    if  $N_{ij} \notin Vset$  then
12:       $NS = \text{dist}(N_{ij}, \text{self})$ 
13:      if  $NS < r$  then
14:        report error, continue
15:      add  $N_{ij}$  to  $Vset$ 
16: for all each 2-hop neighbor node  $N_i$  do
17:   for all each  $N_i$ 's neighbor node  $N_{ij}$  do
18:     if  $N_{ij} \notin Vset$  then
19:        $NS = \text{dist}(N_{ij}, \text{self})$ 
20:       if  $NS < r$  then
21:         report error, continue
22:       add  $N_{ij}$  to  $Vset$ 

```

(i) **verification of node degree.** This procedure is shown in Algorithm 1. The $Vset$ represents nodes that have been verified as non-fabricated nodes (fabricated node means node not really existing). The function $\text{dist}(A, B)$ computes the Euclidean distance between any node pair A and B . Each node only verifies neighbors within 2 hops: (1) *when verifying a 1-hop neighbor node* N_i , first, node checks the existence of itself in node N_i 's *neighborList*. Then it checks whether node N_{ij} also claims node N_i as its neighbor if node N_i claims N_{ij} as a neighbor. At last, node will try to detect whether node N_{ij} is fabricated or is colluding with node N_i through this fact: if a node is within r distance to the checking node, then follow our assumption, it should be the checking node's neighbor. Take node F in Fig.2 for example, it will verify node G and E . When verifying node G , it checks whether node G 's *neighborList* contains node F itself, then it checks whether node G 's neighbors F and H also claim G as their neighbor, at last it checks whether node F and H are not fabricated. Node E is verified in the similar way. (2) *when verifying a 2-hop neighbor node* N_i , we only need to verify that all N_i 's neighbors are not fabricated. This verification doesn't add any communication overload (because sending information to the 2-hop neighbors is already needed in (1)) but increases the detection ability. For example, as shown in Fig.2, node G is a 2-hop neighbor of node A , and node A fabricates a node D and "places" it at a location nearby. Then we can see that only node G can detect the fabrication of D , because node G is supposed to have node D as its neighbor for D is less than r distance to it.

(ii) **verification of distances with neighbors, speed, cumulative clusterhead time and combined weight.** Here nodes only verify these values of 1-hop neighbors. We will discuss

the verification method of different items one by one. (1) all nodes' degrees have been verified in (i) and we assume that each node's location has been verified before clustering. So we can first check whether the locations of neighbor nodes equal with their locations in the *neighborList* fields of received packets, then we calculate and verify the *distances with neighbors* of 1-hop neighbor nodes. (2) the *speed* of a node can be verified by using its location in the former cluster election round and its current location. (3) in SWCA each node knows the elected clusterheads, they certainly can verify the *cumulative clusterhead times* of neighbors. (4) since all the four weights corresponding to four system parameters are known and verified, each node can calculate the *combined weights* of neighbors now, and then checks whether these values are equal with the values in the weight packets of neighbors.

Step 5: Now that the combined weights of all nodes are known and verified, nodes can compute the clusterheads distributedly as in WCA [9], which is an iterative process: (1) choose the current minimum weight node as clusterhead, (2) assign the head's unassigned neighbors as the head's citizen nodes, (3) mark the clusterhead and its citizens as assigned, and repeat (1) until no unassigned node existing.

C. Dealing With Error Reports

In order to avoid defamation by malicious nodes, only *acceptable* reports are processed in SWCA. Specifically, we limit the number of report a node can send by a threshold δ , e.g., $\delta = 1$. Then a report is *acceptable* if and only if the reporting node hasn't reported before (i.e., $repNum < \delta$) and the reported node hasn't not been precluded. This strategy can efficiently limit defamation by malicious nodes. For example, when node *A* discovers misbehavior of a node *B* during the verification, it will broadcast an error report to the network. Node *B* may also send an error report to defame node *A*. Then with our strategy, the worst case is that both a normal (well-behaved) node and a malicious node are precluded from being elected as head. We will show in Section V that the loss of small number normal nodes doesn't have much effect on detection ratio of node fabrication, given that the density of normal nodes is enough.

V. ANALYSIS

In this section, we discuss the security properties of SWCA. From Step 4 (ii) of Section IV-B, we can see that the correctness of *cumulative clusterhead time* can be easily achieved because each node knows all the clusterheads in each round (computed in Step 5), also the *speed* can be easily verified when the locations of neighbor node in the former round and in current round are known. However, the correctness of *distances with neighbors* depends on the truth of *node degree*. Because if there is any fabricated node, the values of distances to neighbors can be easily changed. Also *node degree* itself is an importance election-related feature. So it's critical to ensure that the values of *node degrees* are correct. Next we will consider how SWCA resists to *node degrees* tampering

(i.e., adding fabricated node). We consider the problem in 2-Dimensions here and start the analysis with a definition.

Definition 1: The *possible coverage area* of a node is the circular area around node with radius= R , and the *secure coverage area* of node is the circular area around node with radius= r . R and r are the maximal and minimal transmission ranges of node respectively.

SWCA has the desirable property as shown in Theorem 1 below:

Theorem 1: If each node's *possible coverage area* is covered by within-2-hop normal neighbor nodes' *secure coverage areas*, that's to say all locations of its *possible coverage area* are in at least one within-2-hop normal neighbor node's *secure coverage area*, then no fabricated node can exist without being detected.

Proof: We prove it by contradiction. Suppose that a malicious node fabricates a node *F*, and claims it at a location successfully without being detected, in the *secure coverage area* of a normal neighbor node *B*. *B* is within 2 hops to the malicious node. Then according to our assumption, node *B* and node *F* should be neighbors, so if node *F* really exists, node *B* and *F* will be in each other's neighbor list during verifying node's degree in Step 4. But in fact, *F* is faked and doesn't lie there, so *B* will fail to find such a neighbor. Then because *B* is a normal node, it will detect and report the error. So there is a contradiction. ■

If the premise of Theorem 1 holds, then node degree tampering will be thwarted *absolutely*. However, the theorem only gives the ideal case; when we assume the fabricated node is uniformly distributed, the real detection ratio (i.e., $\frac{\text{number of detected fabricated nodes}}{\text{number of total fabricated nodes}}$) of node fabrication is equal to the *coverage ratio* of its *possible coverage area* by within-2-hop normal neighbors' *secure coverage areas*. So we will give a lower bound of the coverage ratio in the next subsection.

A. A Lower Bound of The Coverage Ratio

As stated previously, the detection ratio of node fabrication depends on the coverage ratio of node; we will compute a lower bound of the coverage ratio (P_{cov}) in this subsection. We assume that there are N nodes deployed in an area of S . Then the node density (nodes per unit area) $D = \frac{N}{S}$. Given a region of area S_a , the probability of i nodes lying in it is Poisson distributed, i.e.,

$$p_i = \frac{(S_a D)^i e^{-S_a D}}{i!} \quad (i = 0, 1, 2, \dots) \quad (1)$$

The probability for a node *A* to be covered by at least one within-2-hop neighbor is the complement of the probability for no neighbor to cover the node *A*, so,

$$P_{cov} = 1 - P_{ncov} \quad (2)$$

and

$$P_{ncov} = \frac{\iint_{\Sigma} p(s) ds}{\text{Area of } \Sigma} \quad (3)$$

where Σ is the circle of radius R and $p(s)$ is the probability for no within-2-hop neighbor of node *A* to cover the small differential s area. Next we will compute the P_{ncov} .

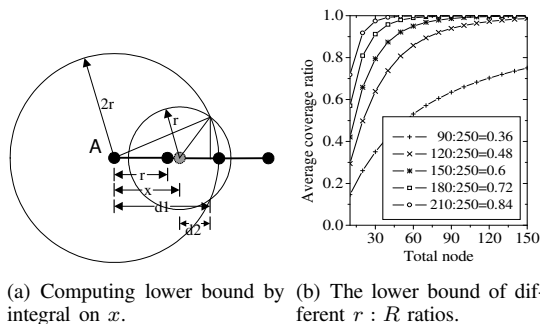


Fig. 3. Lower bound illustration.

We assume that a node is a within-2-hop neighbor of the node A only when it is within $2r$ distance to node A , to get a lower bound. We classify the calculation into two different cases for a easy integral, as shown in 3(a):

(i) when $R \geq r$ and $R < 3r$, the integral interval on x is from 0 to R , which can be separated to one part of “the whole r -radius small circle” and another part of “the intersection of two circles”:

$$P_{ncov} = \frac{\int_0^r 2\pi x e^{-\pi r^2 D} dx + \int_r^R 2\pi x e^{-S(x)D} dx}{\pi R^2} \quad (4)$$

$$= \frac{r^2 e^{-\pi r^2 D} + \int_r^R 2x e^{-S(x)D} dx}{R^2} \quad (5)$$

where $S(x)$ is the area of the intersection region of two circles, which can be computed by geometric property. There are two conditions:

- when $x \geq \sqrt{(4r^2 - r^2)} = \sqrt{3}r$, $S(x) = 4r^2 \arccos \frac{d_1}{(2r)} - d_1 \sqrt{4r^2 - d_1^2} + r^2 \arccos \frac{d_2}{r} - d_2 \sqrt{r^2 - d_2^2}$, where $d_1 = \frac{x^2 + 3r^2}{2x}$, $d_2 = \frac{x^2 - 3r^2}{2x}$,
- when $x < \sqrt{(4r^2 - r^2)} = \sqrt{3}r$, $S(x) = (2r)^2 \arccos \frac{d_1}{(2r)} - d_1 \sqrt{4r^2 - d_1^2} + r^2(\pi - \arccos \frac{d_2}{r}) + d_2 \sqrt{r^2 - d_2^2}$, where $d_1 = \frac{x^2 + 3r^2}{2x}$, $d_2 = \frac{3r^2 - x^2}{2x}$.

(ii) when $R \geq 3r$, the calculation is the same with above. But the second integral will be only to $3r$, because if a location is beyond $3r$, no within-2-hop neighbor will cover it.

$$P_{ncov} = \frac{r^2 e^{-\pi r^2 D} + \int_r^{3r} 2x e^{-S(x)D} dx}{R^2} \quad (6)$$

Now, following equation (2), (5), and (6), we are able to compute a lower bound of P_{cov} . We demonstrate the values of different input parameters to give an intuitive impression. We deploy various number of nodes randomly in an area of $1000m \times 1000m$ and different $r : R$ ratios are used. The results are shown in Fig.3(b). We can see that when $r : R = 150m : 250m = 0.6$, the coverage ratio is more than 90% with only 50 total node number. Also a greater $r : R$ ratio can gain higher detection ability.

B. Security Analysis

For the page limitation we selectively list some attacks and briefly discuss how SWCA resists these attacks, according to our attacker model.

Attacker may alter the packet contents or masquerade another node to send packets. For example, attacker changes the others packet fields corresponding to the values of election-related features to decrease their election priority. In SWCA, the usage of TESLA prevents such attacks: receiver accepts a packet only if it satisfies TESLA constraints [18].

Node may fabricate neighbor nodes and add them to its neighbor list to increase its degree, thus it gains a higher election priority. SWCA is resilient to such attack. According to the Theorem 1, it will be impossible to fabricate nodes without being detected, provided that the *possible coverage area* of attacker is covered by its normal within-2-hop neighbor nodes. We showed by analysis that not many nodes are needed in reality in Section V-A, also we will show that by simulations in Section VI-B. Additionally, it's easy to see that the reduction of node degree can be detected by neighbors.

Nodes may collude to increase their node degrees. For example, node A and node B are not neighborhood, but they may construct *info packets* claiming each other in their neighbor lists and tunnel packets to each other. However these attacks will fail because such attacks have the same effect of node A (so does B) creating a fabricated node in its neighborhood, which will be detected by normal neighbors.

VI. SIMULATION EVALUATION

A. SWCA Performance Evaluation

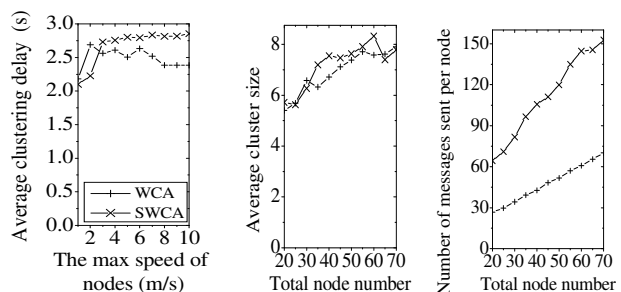


Fig. 4. Simulation Results of SWCA Performance.

We implement SWCA in GloMoSim to evaluate its performance. The TESLA parameters we use in SWCA are: interval $I=0.6(s)$, maximal synchronization error $\Delta = 0.1(s)$, delay $d = 2$. We make a comparison to the implementation of WCA by Jorge Nuevo [6]. In all the simulations nodes are randomly placed within a $1000 \times 1000m^2$ area, and transmission range is 250m, using IEEE 802.11 as MAC protocol, AODV as routing protocol, and Random Waypoint as mobility model.

Three performance metrics are considered here: *clustering delay*, *cluster size* and *the number of messages sent per node* in each clustering process. The result of clustering delay is shown in Fig.4(a), which indicates that the average delay of SWCA is a little longer but close to WCA. The average cluster sizes of both algorithms is shown in Fig.4(b), we can see that the cluster size of SWCA is similar to WCA, since they follow the same weight parameters. The number of packets sent to MAC layer is captured as *the number of messages sent per node*.

The result is shown in Fig.4(c). We can see that the number of messages sent by SWCA is more than WCA, because we need more authenticated information on topology. However we believe the overload is tolerable and can be amortized since the authenticated information can also be used by other applications like routing.

B. Detection of Node Degree Tamping Evaluation

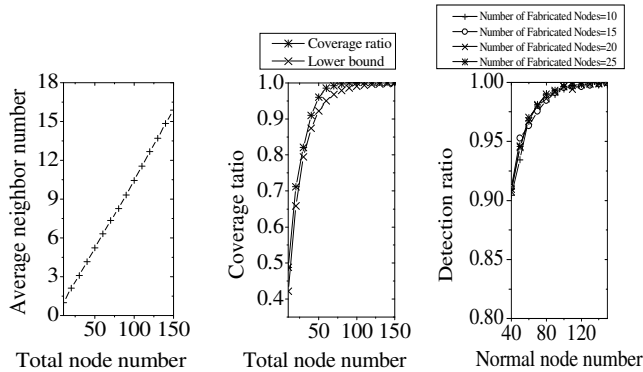


Fig. 5. Simulation Results of the Detection of Node Degree Tamping.

We evaluate the detection ability of node degree tamping here to confirm the analysis in Section V-A. Quasi unit disk graph model (Quasi-UDG) [21] is used here; it is more realistic than unit disk graph model [21]. In this model, there are two parameters: the transmission range R , and a additional factor α . Then there is a link between a node pair when the distance of them $d < \alpha R$, no link if $d > R$; when d is within $[\alpha R, R]$, we assume presence of a link with probability $\frac{R-d}{R-\alpha R}$. In our simulation, nodes are randomly placed within a $1000m \times 1000m$ area, $R = 250m$ and $\alpha = 0.6$ (i.e., $r = 150m$).

Fig.5(a) shows the average neighbor node number under this model. In Fig.5(b), we show the average coverage ratio (i.e., P_{cov}) of simulation, and the corresponding theoretical lower bound values computed following the analysis in Section V-A. The figure indicates that, with 40 and 50 nodes, the coverage ratios exceed 90% and 95% respectively. We give another simulation on detection ratio: we randomly choose some nodes as attackers, and each selected node will fabricate a node and randomly place it in its possible coverage area. Fig.5(c) shows the detection ratios with different number of attackers. We can see that the detection ratio is more than 95% in all the cases when there are 60 normal nodes.

VII. CONCLUSION

In this paper we present the design and evaluation of SWCA, a secure clustering algorithm based on WCA. SWCA uses TESLA to authenticate the communication, and a novel neighbor verification scheme to verify the values of election-related features (this scheme only concerns nodes within 2 hops). Our analysis and simulation results show that SWCA is resilient to attacks but still has comparable clustering performance to WCA. To the best of our knowledge, SWCA is

the first algorithm in ad hoc networks considering the security of l -hop clustering.

ACKNOWLEDGMENT

The authors are grateful to Zhuo Li, Lei Xie, Shigeng Zhang, and Daqiang Zhang for their comments, which helped improve the quality of this paper. This work is supported in part by Hong Kong Research Grant Council under CERG grant PolyU 5102/07E, the Hong Kong Polytechnic University under the ICRG grant G-YF61, China 973 Project under Grant No. 2006CB303000, and the Natural Science Foundation of Jiangsu Province under Grant No.BK2007136.

REFERENCES

- [1] D. Baker and A. Ephremides, "The architectural organization of a mobile radio network via a distributed algorithm," *IEEE Trans. on Communications*, vol. COM-29, no. 11, pp. 1694–1701, 1981.
- [2] M. Gerla and J. T.-C. Tsai, "Multicluseter, mobile, multimedia radio network," *ACM Journal on Wireless Networks*, vol. 1, no. 3, pp. 255–265, 1995.
- [3] P. Krishna, N. Vaidya, M. Chatterjee, and D. Pradhan, "A cluster-based approach for routing in dynamic networks," *ACM CCR*, vol. 27, no. 2, pp. 49–65, Apr. 1997.
- [4] Y. chee Tseng, S. yao Ni, Y. shyan Chen, and J. ping Sheu, "The broadcast storm problem in a mobile ad hoc network," *Wireless Networks*, vol. 8, pp. 153–167, 2002.
- [5] C.R.Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE JSAC*, vol. 15, no. 7, pp. 1265–1275, Sept. 1997.
- [6] J. Nuevo and J.-C. Grégoire, "A hierarchical service distribution architecture for ad hoc networks based on the weighted clustering algorithm," in *Proceedings of the 5th European Wireless Conference (EW 2004)*, Barcelona, Spain, 2004.
- [7] Y. Huang and W. Lee, "A cooperative intrusion detection system for ad hoc networks," in *Proceedings of SASN*, Fairfax VA, October 2003.
- [8] M.Bechler, H.J.Hof, D.Kraft, F.Paehlke, and L.Wolf, "A cluster-based security architecture for ad hoc networks," in *Proceedings of INFOCOM*, Hong Kong, China, March 2004.
- [9] M. Chatterjee, S. Das, and D. Turgut, "WCA: A weighted clustering algorithm for mobile ad hoc networks," *Journal of Cluster Computing*, vol. 5, no. 2, pp. 193–204, 2002.
- [10] A. B. McDonald and T. Znati, "A mobility-based framework for adaptive clustering in wireless ad-hoc networks," *IEEE JSAC*, vol. 17, no. 8, August 1999.
- [11] L. Ramachandran, M. Kapoor, A. Sarkar, and A. Aggarwal, "Clustering algorithms for wireless ad hoc networks," in *Proceedings of DIAL-M*, 2000, pp. 54–63.
- [12] S. Basagni, "Distributed clustering for ad hoc networks," in *Proceedings of I-SPAN*, Jun. 1999, pp. 310–315.
- [13] A. Amis, R. Prakash, T. Vuong, and D. Huynh, "Max-min d-cluster formation in wireless ad hoc networks," in *Proceedings of INFOCOM*, March 2000.
- [14] F. G. Nocetti, J. Solano-González, and I. Stojmenovic, "Connectivity based k-hop clustering in wireless networks," *Telecommunication Systems*, vol. 22, pp. 105–220, 2003.
- [15] S. Vasudevan, B. DeCleene, N. Immerman, J. Kurose, and D. Towsley, "Leader election algorithms for wireless ad hoc networks," in *Proceedings of DISCEX III*, April 2003.
- [16] K. Sun, P. Peng, and P. Ning, "Secure distributed cluster formation in wireless sensor networks," in *Proceedings of ACSAC*, 2006.
- [17] D. Liu, "Resilient cluster formation for sensor networks," in *Proceedings of ICDCS*, 2007.
- [18] A. Perrig, R. Canetti, D. Tygar, and D. Song, "Efficient authentication and signature of multicast streams over lossy channels," in *Proceedings of IEEE S&P*, May 2000, pp. 56–73.
- [19] S. Basagni, M. Mastrogianni, A. Panconesi, and C. Petrioli, "Localized protocols for ad hoc clustering and backbone formation: A performance comparison," *IEEE TPDS*, vol. 17, no. 4, pp. 292–306, 2006.
- [20] S. Guha, R. N. Murty, and E. G. Sirer, "Sextant: A unified node and event localization framework using nonconvex constraints," in *Proceedings of MobiHoc*, 2005.
- [21] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Adhoc networks beyond unit disk graphs," in *Proceedings of DIAL M-POMC*, 2003.
- [22] L. Zhou and Z. J. Haas, "Securing ad hoc networks," *IEEE Network Magazine*, vol. 13, no. 6, pp. 24–30, 1999.
- [23] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing robust and ubiquitous security support for mobile ad-hoc networks," in *Proceedings of ICNP*, 2001.
- [24] L. Lazos and R. Poovendran, "SeRLoc: Secure range-independent localization for wireless sensor networks," in *Proceedings of ACM WiSe*, 2004.
- [25] S. Čapkun and J. P. Hubaux, "Secure positioning in wireless networks," *IEEE JSAC*, 2006.
- [26] Y. Hu, A. Perrig, and D. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," in *Proceedings of MobiCom*, 2002.