J Supercomput (2013) 64:717–743
DOI 10.1007/s11227-010-0508-x

Dynamic camouflage event based malicious node detection architecture

Kanthakumar Pongaliur · Li Xiao · Alex X. Liu

Published online: 26 November 2010

Abstract Compromised sensor nodes may collude to segregate a specific region of the sensor network preventing event reporting packets in this region from reaching the basestation. Additionally, they can cause skepticism over all data collected. Identifying and segregating such compromised nodes while identifying the type of attack with a certain confidence level is critical to the smooth functioning of a sensor network. Existing work specializes in preventing or identifying a specific type of attack and lacks a unified architecture to identify multiple attack types. Dynamic Camouflage Event-Based Malicious Node Detection Architecture (D-CENDA) is a proactive architecture that uses camouflage events generated by mobile-nodes to detect malicious nodes while identifying the type of attack. We exploit the spatial and temporal information of camouflage event while analyzing the packets to identify malicious activity. We have simulated D-CENDA to compare its performance with other techniques that provide protection against individual attack types and the results show marked improvement in malicious node detection while having significantly less false positive rate. Moreover, D-CENDA can identify the type of attack and is flexible to be configured to include other attack types in future.

Keywords Sensor · Networks · Security

1 Introduction

Sensor networks find application in different disciplines ranging from research to practical on-field military interests. Provided the vast scope, securing a sensor network is critical given the deployment scenario wherein the nodes may be left unattended over long durations of time. The sensor network can be under attack from

K. Pongaliur (✉) · L. Xiao · A.X. Liu
Department of Computer Science and Engineering, 3115 Engineering Building, East Lansing, MI 48824, USA
e-mail: pongaliu@cse.msu.edu

different types of adversaries, some intentional (malicious user) and some unintentional like environmental conditions.

In this paper we look at collaborative adversaries whose goal is to segregate part of the network in order to prevent event reporting by either dropping or corruption of packets. Also, if the events are sporadic, then the basestation will not be able to differentiate between non-occurrence of an event and non-report of an event due to malicious activity. This is important in military applications such as detecting heavy artillery movement. There have been several studies to protect the sensor network from different kinds of adversaries trying to launch various types of attacks. Protocols like TinySec [4], SPIN [8], TinyPK [14], SERP [7] have been developed to provide security and to maintain integrity of the communication data. Pirzada et al. [9] use a trust model like a reputation scheme to identify the sinkholes and wormholes in the network. In this scheme, they guarantee that the packet reaches the basestation using a trusted path, but do not detect the malicious nodes in network.

The research studies so far have provided solutions which work in an inside-out fashion, i.e. the onus of identifying such mal-intent is left to the sensor nodes. There have been numerous reputation-based systems, which use this methodology attempting to solve the problem of intrusion detection and mal-behavior detection of sensor nodes [2, 3, 11]. Krontiris et al. design an IDS to detect specifically sinkhole attacks [5]. In [6], Ngai et al. provide a scheme to analyze packets to detect the intruder. Many of the existing approaches tend to prevent a specific attack type. Even the primary goal of an intrusion detection system is to detect an intrusion while identifying the attack type is secondary. An extension of the intrusion detection system is the intrusion prevention system, which is a real time reactive system to block malicious activities. In [13], Su et al. present an intrusion detection and prevention system. What is lacking is a proactive architecture that can detect a malicious attack while also being able to identify the type of attack. The threat model considered is a collaborative attack to segregate a region so as to disallow event packets in this region from reaching the basestation. The adversary is a laptop class attacker which can perform four attack types, namely sinkhole attack, selective forwarding, wormhole and sybil attack to achieve the same.

In this paper we look at solving the problem in which the onus of identifying the mal-intent of a sensor node is left to the basestation. The work presented in this paper is an enhancement over Camouflage Event Bases Malicious Node Detection Architecture (CENDA) [10]. CENDA is a multiphase proactive architecture in which the basestation exploits the spatial and temporal information of the camouflage event to detect the malicious node. D-CENDA improves CENDA by using dynamic feedback from the network in the form of sensor node usage for event detection and packet propagation. Also, the overhead of the architecture is reduced by using simple bit-arrays and a new shortened relative addressing scheme. D-CENDA uses a multiphase approach that includes camouflage event generation, planning the mobile-node tour, encoding the address in the packet, packet analysis by the basestation, detection and verification of the malicious node.

A camouflage event is a reputable event generated in response to a basestation request. The camouflage event generator is a mobile-node which can be mounted on robot or an unmanned aerial vehicle. This mobile-node traverses the path decided by

Table 1 List of notation

n	Number of nodes	C_a	Set of nodes sensing over unit area a
f_i	Forwarding number of node i	CoI	Coverage inheritance of node i
f_{mean}	Mean of forwarding numbers	b_i	Boolean value of node i is cut vertex
CI	Critical index	d_i	Feedback on usage of node i
U_i	Unit areas having node i coverage	CA	Cumulative attack

the basestation and generates the camouflage events at regulated intervals of time. These events are called camouflage events since they mimic the real events, but are not real events in the true sense. A simple solution is for the basestation initiating a camouflage event from the node by sending a message, but a powerful adversary can overhear the communication and allow such event packets. The nodes which are in the sensing range of the camouflage event location will detect the event and report back to the basestation. We provide a lightweight address encoding scheme wherein each node encodes the shortened relative address of the node from which it received the packet. Each node also maintains the information about the overheard packet transmissions by the neighbors. We present a bloom filter [1] and a bit-array-based scheme which are used for verification while branding a node malicious.

Performance of D-CENDA is compared with [15] and [6] that provide protection against individual attack types and is seen to show marked improvement in malicious node detection while having significantly less false positives. Moreover, D-CENDA is able to identify the type of malicious activity and is flexible to include other attack types. Additionally, we present results of D-CENDA in relation to the results from CENDA.

The paper is organized as follows. We discuss classification of nodes and definition of metrics in Sect. 2. The threat model is presented in Sect. 3. As part of the architecture, we design the mobile route for the camouflage event generator in Sect. 4. This is followed by attack fingerprinting in Sect. 5. Then, we present the security architecture in Sect. 6. Finally, we present the results in Sect. 7, followed by the conclusions in Sect. 8.

2 Node classification and metrics definition

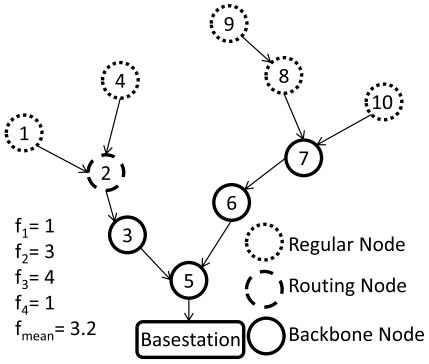
This section presents the node classification and metrics definition.

2.1 Node classification

Let n be the number of nodes and f_i is the count of the number of nodes to which node i forwards the packet. f_{mean} is the mean of the forwarding number of all nodes. Table 1 provides a list of notations. At network initialization, the forwarding number of a node is set to be inversely proportional to its distance from the basestation. An example is presented in Fig. 1. The nodes are classified as follows:

1. Regular nodes: Nodes forward packets for two or less nodes including itself.

Fig. 1 Node classification



2. Routing nodes: Routing nodes forward packets for more than two nodes but less than the f_{mean}.
3. Backbone nodes: Nodes forward packets for greater than or equal to f_{mean}.

$$\text{Regular Nodes} = \{n_i\} \quad \forall i \text{ where } f_i \leq 2,$$

$$\text{Routing Nodes} = \{n_i\} \quad \forall i \text{ where } 2 < f_i < f_{\text{mean}},$$

$$\text{Backbone Nodes} = \{n_i\} \quad \forall i \text{ where } f_i \geq f_{\text{mean}}.$$

2.2 Metrics definition

Critical Index (CI) is a per node metric, which is the availability of sensor coverage over an area. The sensing area is divided into m unit regions. Each node i senses over a set of unit areas called Sensing Area. U_i is a set of unit area's over which node i has sensing coverage. Each unit area will be monitored by a set of nodes called Sensing Nodes represented as C_a where a is the unit area identifier. The Coverage Inheritance (CoI_i) of a node i is defined as the average of C_a where area a belongs to the set U_i.

$$\text{CoI}_i = \frac{\sum_a C_a}{|U_i|} \quad \text{where } a \in U_i.$$

The coverage inheritance is an inverse property i.e. higher the coverage inheritance of a node, lesser is its importance. It is an acquired factor because the value of this parameter is dictated by other peer nodes sensing the region.

We next add an active component called progressive feedback (d_i). The progressive feedback is the dynamic component which changes based on the real time usage of a node. A node can be close to the basestation but can remain redundant until another node fails and the packets need to be routed through it. Also, progressive feedback is different from forwarding number, as it is based on the practical usage of the node in the past time periods and not a calculated theoretical parameter. We

compute the CI of a node based on the Forwarding Number and the CoI of the node. The CI of a node is calculated as follows:

$$\mathrm{CI}_i = \alpha f_i + \frac{\beta}{\mathrm{CoI}_i} + \gamma b_i + \rho d_i \quad \text{where } \alpha + \beta + \gamma + \rho = 1,$$

where α, β, γ and ρ are constant coefficients, b_i is a boolean variable identifying if node i is a cut-vertex. d_i is the usage of node i in the network over the last time period. Based on the node usage as a sensing unit or as a forwarding entity, the critical index gets updated in real time. If a node is a cut-vertex, then the importance of the node increases manifold, as the loss of this node can partition the network resulting in some nodes unable to reach the basestation. The constant coefficients are set based on the application.

2.2.1 Impact of coefficients

We performed an analysis to study the impact of constant coefficients on the number of critical nodes selected. We were interested in seeing the variance of critical nodes selected when compared to the critical nodes selected when each parameter was given the same weight of 0.25. The threshold value of critical index (normalized) is set to 0.25. It should be noted that a lower threshold value will result in higher number of nodes selected as critical nodes. The total number of nodes is 512. It was seen that initially 371 nodes were identified as critical. After five runs of simulation, we performed two cases of analysis with the weights as follows:

1. $\alpha = \beta = \gamma = 0.2$ and $\rho = 0.4$
   The resultant number of critical nodes after five periods of simulation is depicted in Fig. 2. It is seen that of the 371 critical nodes only 325 nodes were reselected as critical nodes, but in addition 37 new nodes were selected as critical nodes. 46 nodes which were selected initially as critical nodes were not selected.

Fig. 2 Critical nodes case 1

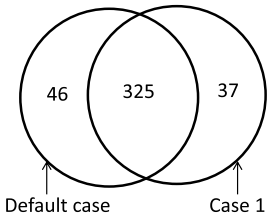
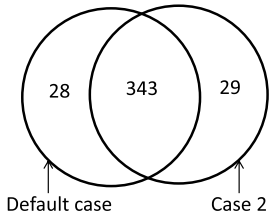


Fig. 3 Critical nodes case 2



2. $\alpha = \beta = \gamma = 0.3$ and $\rho = 0.1$
   Similarly, number of critical nodes and its variance compared to the default is depicted in Fig. 3.

It is seen that the changes in the weights impact the critical nodes selected. Giving a higher weight to progressive feedback has its own pros and cons. As discussed earlier, a node may not be currently used, but can be significant in routing packets if another node which is currently heavily used goes down. Hence, depending on the application, it is important to give the proper weightage to each of the parameters.

3 Threat model

The adversary is a laptop class attacker and on capturing a node, it has access to everything on the node. Also, the malicious entity would want to capture the nodes such that the minimum number of node captures result in the maximum damage. Being a powerful node, it is reasonable to assume that the malicious entity can overhear communication over larger area compared to the sensor node and can make informed decisions about the nodes it wants to attack. The malicious entity captures a node and does one or more of the following attacks: Sinkhole attack, Wormhole attack, Sybil attack or Selective Forwarding attack. The goal of the attacker is to segregate a region such that event packets do not get reported from the region, but may allow other communication. The modus operandi of the threat model is to increase the value of Cumulative Attack (CA), where cumulative attack is defined as follows:

$$\text{CA} = \frac{\sum C_p}{|p|} \quad \text{where } C_p\text{—critical index of node } p.$$

The adversary wants to compromise a node and perform attack from the list above in order to achieve the goal of region segregation. The attacks can be composite in nature i.e. multiple different types of attacks simultaneously.

4 Mobile-node route design

The mobile node is the camouflage event generator. The route followed by this node governs the location and time of the camouflage event which in turn determines the nodes that will detect this event. The design of an optimized route is critical due to the following reasons: a poorly designed mobile node route can cost the network dearly in the form of event detection by unwarranted nodes and having the network flooded by these packets. The second reason is the waste of time and energy of the mobile node to follow a non-prime path. The process includes a node selection preliminary step. Table 2 presents a list of notations.

4.1 Node selection

It is costly to protect all nodes. Under attack, we need to be able to identify and protect the nodes which are critical to the network functionality. Hence, we prioritize

Table 2 List of notation

CN	Set of critical nodes	MCN	Maximum coverage node
C_m	Set of unit areas	L_c	Camouflage event location set
CovAreas	Set of unit areas covered	X_p	Set of privileged critical nodes
X_n	Set of all nodes	T	Sensing range of node
τ	Critical index threshold		

Algorithm 1: Node Selection

Input: $\{\text{CovArea}\} \Leftarrow C_m$, $\{\text{CN}\} \Leftarrow \phi$, $X_n = \{\text{Nodes}\}$
Output: Critical nodes in {CN}
for $x_i \in X_n$ and $\text{CI}_i > \tau$ **do**
  $\{\text{CN}\} \Leftarrow \{\text{CN}\} + x_i$
  $\{\text{CovArea}\} \Leftarrow \{\text{CovArea}\} - U_i$
while $\{\text{CovArea}\} \neq \{\phi\}$ **do**
  $\text{MCN} \Leftarrow x_i$ where $(((x_i \in X_n)$ and $(x_i \notin \text{CN}))$
    and $x_i = (\max(U_n - \{\text{CovArea}\})))$
  $\{\text{CovArea}\} = \{\text{CovArea}\} - U_i$
  $\{\text{CN}\} = \{\text{CN}\} + \text{MCN}$

the nodes based on their role/importance. This importance is dictated by different factors like location, power etc. Based on the classifications and definitions in Sect. 2, we want to select a subset of nodes such that they satisfy the requirements and is depicted in Algorithm 1.

4.2 Route design

M is the mobile route, $L_1, L_2, \ldots, L_c$ are the locations on route M where the event generation occur and c is the number of stops in a mobile route. If we know the routing paths of the packets, we find the subset of nodes in X_n which should detect the event such that all nodes in CN either detect the event or are on the routing path of node detecting the event. Let this subset of nodes which detect the event be the set of privileged node represented by X_p. If we do not know the routing paths of the packets, then the set CN is the set of privileged node represented as X_p. The goal is to find set of locations L_a in the field such that all nodes in X_p are at a distance of sensing range or less from at least one location in L_n. This is presented in Algorithm 2. In the event of having multiple mobile vehicles, the set X_p is geographically partitioned into the number of mobile vehicles available and the mobile route can be planned for each of the partitions independently. The next step is to calculate the shortest path to cover the locations in set L_c. This is formulated similar to the traditional 'Traveling Salesman Problem' but with an optimization exception to allow the mobile node to revisit any prior visited location.

Algorithm 2: Camouflage Event Location Selection

Output: Set of event locations $\{L_c\}$
$\{L_c\} \Leftarrow \{\phi\}$
while $\{X_p\} \neq \{\phi\}$ **do**
  for $C_m \notin \{L_c\}$ **do**
    $\text{SenseCov}_m \Leftarrow \text{Count}(\text{Distance}(X_p, C_m) \leq T)$
  $\text{MaxCovArea} = \text{Max}(\text{SenseCov}_m)$
  $\{L_c\} = \{L_c\} + \text{MaxCovArea}$
  $\{X_p\} = \{X_p\} - \{\text{Node } X_i \text{ Sensing MaxCovArea}\}$

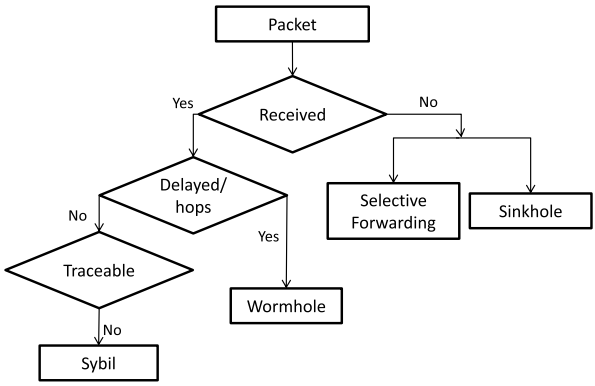


Fig. 4 Attack taxonomy

5 Attack fingerprinting

In this section we model the attacks based on the characteristics possessed by each attack type. A high level classification based on some of the parameters used in D-CENDA is shown in Fig. 4. A detailed information about the attacks is presented in [16] and [12].

5.1 Sybil attack

The basis of sybil attack depends on two factors, first the ease of acquiring different identities, and second, the amount of damage a node can inflict by acquiring the identity. In our system which is based on the node's response to the real-world-like camouflage events, a node will not be able to tarnish the reputation of another node since the neighborhood of the nodes is set and the peers are not required to maintain reputation. The control of managing the reputation lies with the basestation. In D-CENDA, the sybil attack possesses characteristics similar to wormhole attack in which case, the non-verification of the traceback path to the originating node indicates the presence of malicious activity.

5.2 Wormhole attack

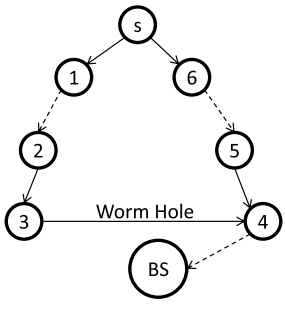
Lemma: It is not possible for the colluding nodes to do significant damage and thwart being identified in case of wormhole attacks in D-CENDA.

Successfully tracing back to the source under wormhole attack: Consider the scenario in Fig. 5 wherein the source node s sends packet to the basestation. The nodes 3 and 4 collaborate to launch a wormhole attack. In this case, node 4 will fake a relative address of a neighbor before forwarding the packet since it cannot specify that it received the packet from node 3. For the basestation to be able to traceback to the source, it needs to satisfy the following. There needs to exist a valid path from node 4 to source with $h + 1$ number of hops where h is the number of hops from node 3 to the source. Let there exist a path P satisfying this condition. It needs to satisfy the per-hop condition such that relative addresses encoded in the packet from source to node 3 match that of a new path from node 4. Also, the effectiveness of a wormhole attack depends on how farther away the traffic is re-routed in the network, thereby increasing the resource consumption in the network. If it needs to satisfy the first condition i.e. to maintain the hop count, then the collaborating attacking nodes within the network cannot reroute the packet across large distances.

5.3 Selective forwarding attack

Selective forwarding attack is a form which can be easily confused with sinkhole attack. Also, a bad channel leading to packet losses can imitate a selective forwarding attack. To differentiate this attack from packet losses due to bad channel or interference, some knowledge of the channel condition is required which is not in the scope of this work. But to differentiate it from a sinkhole attack, we will need to gather packet loss information over a larger period of time. For this purpose, the dynamics of analysis for sinkhole attack and selective forwarding attacks are different. For selective forwarding attacks, the information collection period has to be longer compared to selective forwarding attacks. Hence, in D-CENDA while a sinkhole attack can be identified within a few rounds, identifying the selective forwarding attack requires to gather information over multiple rounds. In our simulation, to detect a selective forwarding attack we analyze the data collected over five rounds.

Fig. 5 Wormhole



5.4 Sinkhole attack

In sinkhole attacks, the node advertises a low cost path to the basestation and may or may not evince itself to be within the neighborhood of a larger number of nodes than it actually is. The basestation knows the location and neighborhood information of all the nodes and can track the loss of packets to within a few hops of this malicious node. This attack type can be identified by analyzing the packets lost and then verified by querying the bloom filters/ bit-arrays of neighborhood nodes in the identified region.

6 Secure architecture

The secure architecture is a sequence of processes for proactively identifying the malicious nodes in the network. It begins with the initiator in the form of camouflage event generator and ends with a verifier using which the basestation verifies the mal-intent of the node before qualifying it as a malicious node. It is a standalone module which consists of the camouflage events, embedding path information, packet meta-analysis, and verification.

6.1 Camouflage events

The events generated by the mobile nodes are called camouflage events. To the basestation, the camouflage event possesses some distinct properties which are not characteristic of real world events. These properties are as follows:

1. The basestation is aware of the location of the event occurrence (L).
2. The basestation knows the precise time of the occurrence of the event (T).
3. With the knowledge of the location and time of event occurrence, the base station knows the set of sensor nodes that detect the event (SN).

The mobile node starts the route designed by the base station. While traversing the path, it stops at the designated location and generates the camouflage event. The nodes sensing the event respond back to the base station. The sensor nodes cannot differentiate this event from an actual event and hence it is handled like a real world event. We need to emphasize this because event type anonymity is crucial to this methodology. If a malicious node can differentiate this event from a real world event, then it can treat just these events like a well-behaved node to escape detection. In D-CENDA, the onus of detecting malicious behavior is on the basestation. The sensor nodes only need to record overheard packet transmissions by the neighbors.

The base station collects all packets that were generated and have traversed through the network. For each packet received by the base station in response to a camouflage event, we have two types of nodes involved. The event notifier node which reports the occurrence of the event and the intermediate nodes through which the packet traversed to reach the base station. At a higher level every packet received by the base station provides one bit information about each of the nodes involved in the delivery of the packet.

Depending on routing, there are two distinct cases that we consider. One in which the packet follows the same path for a set period of time. In this case the basestation

is aware of the path the packet traverses. Hence, when the packet reaches the destination, the basestation knows the nodes which propagated the packet to it. In the second case, instead of following a particular route, the packet follows geographical routing. For the basestation to know the path traversed by the packet, the intermediate nodes need to append additional information. In the absence of this additional information the basestation can only make an educated guess about the intermediate nodes using the knowledge of the location of the nodes and their sleep cycles.

6.2 Embedding route information

In the absence of fixed route, we devise a scheme in which the nodes append the route information to the packet before transmitting over the next hop. This address information being attached to the packet can be of three types.

1. Absolute address: In this case the node appends the absolute address of the neighbor from which it received the packet. The drawback of this scheme is the amount of space it takes to represent the absolute node address.
2. Relative address: We take advantage of the knowledge of the node distribution by the basestation and present a scheme in which the nodes appends the relative address of the neighbor node from which it received the packet.
3. Shortened relative address: The direction of the flow of the event packets is toward the basestation. Using this, we identify the neighbor nodes that should forward the packet to the node and take into consideration only these nodes while encoding the neighbor address in the forwarded packet.

Absolute address is suitable if the number of nodes is small, resulting in shorter address. But the number of nodes in a sensor network can be large, making the absolute address longer. The length of the relative address is dictated by the number of neighboring nodes. To analyze the overhead disparity between the absolute address and relative addresses, we ran simulations to study the distribution of the nodes based on the following parameters: number of nodes, area of field, type of distribution. The type of distribution can be uniform or non-uniform. In uniform distribution, the nodes are homogeneously spread in the sensing area, which leads to the nodes having a very even neighborhood resulting in relative address length close to the mean of all relative address lengths. If the distribution is non-uniform, for densely distributed areas, the nodes are programmed such that only a set of nodes are active at a point of time. This set of nodes can be mutually exclusive to non active nodes.

Using the availability of the topology at the basestation, we further enhance the relative address by using a shortened relative address. This is achieved by classifying the neighboring nodes as upstream and downstream neighbors. An upstream neighbor node is a valid node from which this node receives a packet to forward to the basestation. The other nodes which should not send the packet to this node to forward to the basestation are classified as downstream neighbor nodes. Since, in validity, a node should be only receiving packets from its upstream neighbor nodes, the relative address can be shortened to only encode these neighbors and is called shortened relative address. Table 3 displays the comparison between the absolute address length and the relative address length for different area sizes and different node densities. The absolute address length is calculated as the average number of hops multiplied by the

Table 3 Node distribution

Node	Area (m^2)	Average hops	AAL[a]	RAL[b]	SRAL[c]
512	1000×1000	8	72	32	24
1000	1000×1000	8	80	34	26
2000	1000×1000	7	77	38	31
3000	1000×1000	7	84	38	31

[a]Absolute Address Length (bits)

[b]Relative Address Length (bits)

[c]Shortened Relative Address Length (bits)

number of bits needed to represent the absolute address. Let us consider the case in which the area is 1000×1000, with 512 nodes (this gives us a coverage of four times unit density since we need 128 nodes for unit density coverage while assuming sensing range of 50 m). We see that the average path length is eight hops and the average neighborhood is 14.12. Also, on an average we need four bits to represent the relative address of the neighbor (average neighborhood $\leq$16). Along with the average path length of eight hops we require 32 bits to represent the path from the source to the destination.

6.2.1 Shortened relative neighbor address

The address and representation of the neighbor A for node B is based on two factors. The absolute node identifier of A and the number of neighbors of node B. The relative addresses are assigned in the increasing order of neighbors node identifier. For node 23 in Fig. 6, the relative address and shortened relative address of the neighbor nodes are listed in Table 4. The relative address can be represented using four bits while the absolute address can require up to seven bits. The neighbors which forward packet to a node are the upstream neighbor of the node and all other neighbor nodes are downstream nodes. The downstream neighbors all have shortened relative address of 0, while the upstream neighbors have an incremental address starting from 1 based on their absolute address. Using this classification we can further reduce the address encoding length. We have address length shortening (up to 25%) when using the shortened relative address.

6.2.2 Encoding the path address

When a node receives a packet it appends the relative address of the node from which it received the packet and encrypts it after appending the relative address before forwarding it over the next hop. If node o received the packet from node m, the en-

Fig. 6 Packet route

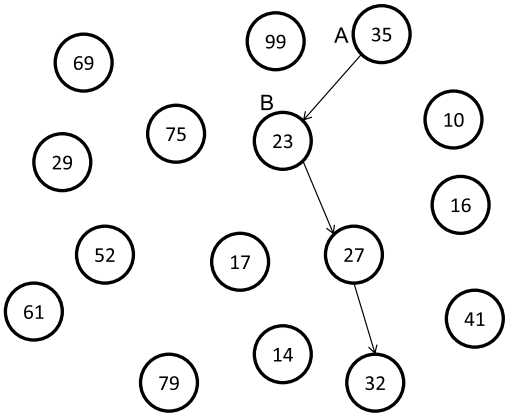


Table 4 Node 23 neighbor address

Aid	Rid	SRid
10	0	1
16	1	0
17	2	0
27	3	0
35	4	2
75	5	0
99	6	3

Aid—Absolute Node identifier
Rid—Relative Address
SRid—Shortened Relative Address

cryption is shown below, where $[\text{data}]_{\text{PK}_o}$ is the encryption of data with the key of node o.

$$[\text{Address} \parallel \text{RA}_{m|o}]_{\text{PK}_o} \quad \text{where } A \parallel B\text{—append } B \text{ to } A,$$

$$\text{RA}_{m|o}\text{—relative address of } m \text{ w.r.t. } o,$$

$$[\text{Addr}]_{\text{PK35}}$$

$$\downarrow$$

$$\left[[\text{Addr}]_{\text{PK35}} \parallel 10\right]_{\text{PK}_{23}}$$

$$\downarrow$$

$$\left[\left[[\text{Addr}]_{\text{PK35}} \parallel 10\right]_{\text{PK}_{23}} \parallel 11\right]_{\text{PK}_{27}}$$

$$\downarrow$$

$$\left[\left[\left[[\text{Addr}]_{\text{PK35}} \parallel 10\right]_{\text{PK}_{23}} \parallel 11\right]_{\text{PK}_{27}} \parallel 10\right]_{\text{PK}_{32}}$$

In Fig. 6, the event is detected by node 35 and it is transmitted to the basestation via nodes 23, 27, 32. Table 5 lists the shortened relative address for the three nodes

Table 5 Address encoding

Node 23		Node 27		Node 32	
Aid	SRid	Aid	SRid	Aid	Rid
10	1	14	0	14	1
16	0	16	1	27	2
17	0	17	2	41	3
27	0	23	3		
35	2	32	0		
75	0	41	0		
99	3				

Aid—Absolute Node id
SRid—Shortened Relative Node id

receiving the packet. Each node before forwarding the packet over the next hop, encodes the relative address information in the packet as follows. Node 23 will encode the relative address of node 35 which is 2 before transmitting to node 27. Node 27 will encode the relative address of node 23 which is 3 before forwarding the packet to node 32 and so forth.

6.2.3 Decoding the path address

When the basestation receives the packet, it knows the sender of the last hop of the packet. In Fig. 6, the last hop forwarder is node 32. The basestation also knows the neighbors of node 32. On decoding the packet using the key specific to node 32, the basestation knows the relative address of the node which forwarded the packet to node 32. Looking up the table, the basestation identifies that the packet was received from node 27. Recursively, the basestation can trace the complete path back to the sender. This is represented as follows:

$$
\begin{aligned}
[\text{EncryptedAddress}]_{\text{PK}_{32}} &\Longrightarrow [\text{EncryptedAddress}_1 \parallel 10] \\
&\downarrow \\
[\text{EncryptedAddress}_1]_{\text{PK}_{27}} &\Longrightarrow [\text{EncryptedAddress}_2 \parallel 11] \\
&\downarrow \\
[\text{EncryptedAddress}_2]_{\text{PK}_{23}} &\Longrightarrow [\text{EncryptedAddress}_3 \parallel 10] \\
&\downarrow \\
&\text{Node 35}
\end{aligned}
$$

6.3 Packet meta-analysis

Let X_i be the set of packets that are generated as a result of a camouflage event. Let R_i be the set of packets that were received by the basestation and L_i is the set of

packets that were not received.

$$R_i \subseteq X_i, \qquad L_i \subseteq X_i, \qquad R_i + L_i = X_i.$$

The basestation maintains records of characteristics for the different attack types for each node. This record consists of a set of parameters which are updated by the basestation whenever a packet is generated for a camouflage event. As a packet is received or not-received, the record for the nodes gets updated accordingly.

The neighbors of each node are classified as either downstream neighbor or upstream neighbor. A neighbor node which helps propagate the packet toward the basestation is a downstream node and the nodes in the neighborhood which are not downstream nodes are upstream nodes. It must be noted that a node which is farther away from the basestation as compared to the neighbor node can be a downstream node.

We maintain a set of four parameters for each node; packets generated, packets received, packets lost, packets garbled. Packets generated is a parameter which keeps track of the packets generated by the node in response to the camouflage event. Packet received is a parameter of the node which represents the number of packets successfully received when the node was either the source of the packet or an intermediate node forwarding the packet. Packet lost parameter of a node tracks the packets that were lost and is statistically determined to see what path the packet should have taken to reach the basestation. Using this information, the intermediate nodes are also marked for the packet lost. The packet garbled parameter tracks the number of packets which were successfully received, but cannot be traced back to the source.

6.3.1 Packets received

These are the packets which are received by the basestation in response to a camouflage event. We calculate the hop count and the total time elapsed between the camouflage event and the time of reception of the packet. This is possible because we have the temporal information about the occurrence of the event. The basestation checks if the number of hops and the time of delivery is within the deviance limits for the particular node. If either of the two parameters lie beyond the threshold limits, we do a path analysis.

If the number of hops is within the limits (called Hop-Validity) and if the time elapsed is also within deviance (Time-Validity), we mark the nodes packet received parameter. If either of hop-validity or time-validity fails, we do a path analysis for the packet. The first step in path analysis is to do a traceback to the source. Under certain circumstances like a wormhole attack, we will not successfully traceback to the source. If the traceback is successful, we perform a per-hop analysis for the packet received. In a per-hop analysis, we consider each pair of consecutive intermediate nodes and see if they adhere to the upstream-downstream requirement.

Let P_i be the packet sent by node N_s to report an event E. m is the number of hops and t is the time taken by the packet to reach the basestation. h_e and t_e are normalized

expected number of hops and time.

$$\text{Hop Validity}_i = \frac{m - h_e}{h_e}, \qquad \text{Time Validity}_i = \frac{t - t_e}{t_e}.$$

If either of the two validity fails, we trace the packet back to the source based on the encoded relative addresses. If relative address is used and trace back to the source is successful, for each hop of the packet, let node x_t be the transmitter and node x_r be the receiver of the packet. Check if node x_r is a downstream neighbor of node x_t. If it is not x_r then we mark node x_t. If the traceback fails, mark the packet garbled parameter similar to the packet lost parameter. This is presented in Algorithm 3. If shortened relative address is used, and if any of the neighborhood address is 0, then query the node which received the packet from the downstream neighbor and mark that node. We populate the table over a period of time and perform a short term analysis and a long term analysis. Each of these analysis caters for a particular type of attack. The short term analysis is done to identify wormhole, sybil and sinkhole attacks while a long term analysis is performed to identify a selective forwarding attack. The long term analysis has sufficient information that helps us to differentiate between the sinkhole and selective forwarding attacks. Also, it helps us differentiate between a dead node and a malicious node performing a selective forwarding attack. The reasoning behind this is the fact that to identify a selective forwarding attack, we need to gather information over longer durations as compared to other attack types.

6.3.2 Packets lost

With the knowledge of the occurrence of the events, the basestation expects a set of packets from a group of nodes. If there is malicious activity, there may be loss of packets. For each packet lost the basestation is able to traceback the packet loss to within a subset of nodes. This is depicted in Algorithm 3.

6.3.3 Verification

Verification is the process performed by the basestation before finalizing the node as malicious. To pinpoint the node which was responsible for the packet loss from within the subset of nodes, we have each node maintaining a simple but efficient data structure which works on the principle of bloom filter and stores overheard packet transmissions. We present two methods to maintain the overheard packet transmission.

1. Bloom filter: Each node maintains multiple counting bloom filters. These are used to mark the overheard packet information during different intervals of time. The nodes overhear the packet transmissions of their neighbors and mark information into the bloom filter using the relative address of the neighbor. The number of bloom filters maintained by each node will decide the granularity of the information stored. For example, information maintained in a single bloom filter over a period of time t will give us less accurate description of events compared to 10 bloom filters maintained by the node for each of $t/10$ time intervals.

Algorithm 3: Node Marking

```
Input: N → all nodes, R_i → received packets
L_i → lost packets, x_n → nodes on path of packet P_n
for P_i ∈ R_i do
    if ((HopValidity_i) || (TimeValidity_i)) fails then
        Initiate Source Traceback
        while (n_i ≠ N_s) || (TracebackAdd! = φ) do
            p = NumNeighbors(n_i)
            RelAddLen = sizeofbitlength(binary(p))
            RelAdd = RelAddLength LSB of TracebackAdd
            n_i = Lookup → RelAdd(n_i)
            TracebackAdd = TracebackAdd − RelAdd
        if Source Traceback == SUCCESS then
            for (Node n ∈ x_n) do
                if n ∈ path of P_i then
                    if x_r is downstream neighbor of x_t then
                        x_t → SuspectNode
        else
            Intermediate Node Identification (Node n ∈ x_n) → Path of P_i
            Mark PacketGarbled(n)
for P_i ∈ L_i do
    Intermediate Node Identification
    for (Node n ∈ x_n) → Path of P_i do
        Mark PacketLost(n)
    Marked Node Analysis
    for (Node n ∈ N) do
        if PktLost(n) ≥ 1 then
            u ∈ UpstreamNeighbor(n)
            if PktLost(n) > ∑PktLost(u) then n ⇒ suspectnode
        if PktGarbled(n) ≥ 1 then
            u ∈ UpstreamNeighbor(n)
            if PktGarbled(n) > ∑PktGarbled(u) then
                n ⇒ suspectnode
```

2. Bit-array: It is seen in Sect. 6.2, each node will have a maximum of 32 active neighbors when the node itself is active. If a node has a higher neighborhood density, then the sleep cycles are programmed such that a node has a maximum of 32 active neighbors. Since the total number of neighbors are few, the second method is to use bit-arrays to store overheard packet transmission. Use of bit-array simplifies the amount of computation required at a sensor node to gather and store the data. Additionally, when the node is queried by the basestation, it is a simple lookup operation by the sensor node to respond to the query. Also, converting

a bit-array into a counting bit-array will require replacing the individual bits by multiple bit cells. Although it is not the most optimal method, but it is the most simplistic method, particularly considering a energy constrained sensor node.

The basestation can query the neighboring downstream nodes to check if they overheard the packet transmission by the node over a particular interval of time. As for the response to the query, the architecture provides two options. One, the queried node responds back to the basestation with its bloom filter or bit-array representation and the basestation processes it to decipher the information. In the second case, the node processes the data structure to extract the data and replies back with only the required information. The two methods have their own advantages and disadvantages. In the first method where the sensor node returns the entire structure as a response, the basestation might be saved from making repeated queries to the same node about its different neighbors. Another significant advantage of this method is the discreteness with regards to the node under observation. The drawback is the increased packet size. In the second method, the communication cost is less, but the basestation will need to query the node about a specific neighbor and cannot be discrete like in the first method. Hence there is loss of anonymity and it can be qualifying factor in certain military applications.

For un-received packets over a period of time, the basestation can track the packets loss to a subset of nodes from the information it gathers for every un-received packet. The basestation queries multiple neighbors (this number can be configured) of these nodes to check if they overheard a packet being transmitted by their neighbor at a particular time interval. The responses are analyzed to check the presence of suspect nodes. We aggressively mark the nodes as suspect, hence this verification process plays a important role in keeping the false positives low while detecting malicious node. We will see in the results in Sect. 7, that the percentage of the false positives considering critical nodes is low (6.5% in sinkhole attacks, 8% in selective forwarding attacks, 9.75% in sybil attacks, and 10.5% in wormhole attacks). The low false positive rates indicate that the exoneration of the non-malicious suspect nodes is high, displaying the higher probability of successful verification by the basestation.

7 Simulation results

The simulations were performed over an area of 500×500 m, having 125 nodes so as to have a unit coverage over the entire field with each node having sensing range of 25 m. The critical index threshold is 0.25 and the four parameters $(\alpha, \beta, \gamma, \rho)$ were each set to 0.25. The simulation is performed in Matlab. There were two varying factors in the attacks. First, the number of malicious (compromised) nodes was varied as 5%, 10%, 15%, 20%. The compromised nodes were randomly selected but the critical nodes were given a higher priority to be under attack considering a worst-case scenario with a sophisticated attacker. Secondly, the attack type of the compromised nodes were again randomly assigned. The base station gathered data over multiple runs. We present the results of D-CENDA while keeping the results from CENDA for a quick comparison and also include the results from [6] and [15] where relevant.

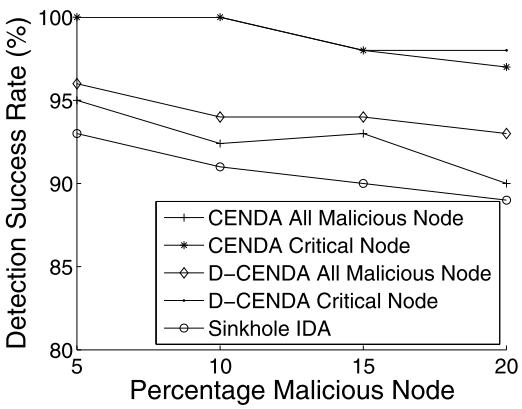


Fig. 7 Sinkhole detection

Figure 7 depicts the detection success, when all the malicious nodes are performing sinkhole attack. Even though when the success rate of identifying the malicious node was not 100%, the subset of malicious node which are critical were identified accurately. There are two reasons for this, the critical nodes have connectivity to the basestation and the camouflage events are generated to cover the critical nodes. It is also seen that the success rate of detecting all malicious nodes is higher for D-CENDA compared to CENDA. This is a result of having dynamic selection of critical nodes which results in selection of some new non-critical malicious nodes for detection purpose that will not occur in CENDA. The results from [6] (sinkhole IDA) have been included for comparison, and it is seen that D-CENDA out performs.

The false positive rate when only considering the critical nodes is slightly higher as compared to the false positive rate when the entire set of malicious nodes was considered. Again, as the number of malicious nodes increased we saw an increase in the false positive rate. This is because we aggressively mark a node as a suspect node to get lesser false negatives, with assurance that the verification discards the false positives as depicted in Fig. 8. Even in false positives accuracy we see an improvement in D-CENDA. The false negative rate when considering the critical nodes was much lower compared to the false negative rates of the whole malicious nodes set (Fig. 9).

The next set of results comprise the malicious nodes all performing selective forwarding attack. We present the results when the nodes randomly dropped 30% of the packets. It was seen that this drop percentage affected the number of rounds the camouflage event generator needed to make, to get accurate results. Lower the drop percentage, higher was the number of rounds needed by the mobile-node, but again lower drop percentage means a less severe attack. The overall performance of D-CENDA was better than CENDA for all the cases except when the percentage of malicious nodes was 15%. This was happening because of dynamic selection of critical nodes changes the critical nodes which were under observation. As mentioned earlier, identifying a selective forwarding attack required us to gather data over multiple runs and the change in nodes under observation within those runs resulted in some undetected malicious nodes. This was seen to happen for critical nodes in the 5% and 20% cases. The results are presented in Figs. 10, 11, 12 and are similar to

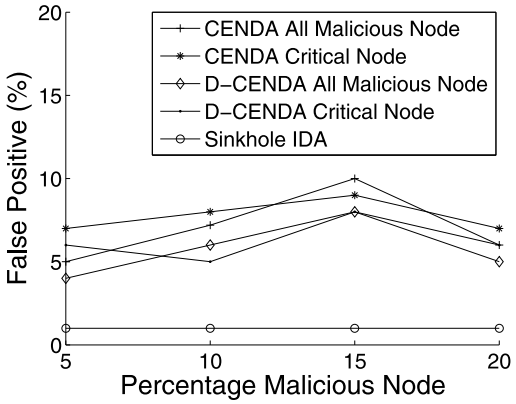


Fig. 8 Sinkhole false positive

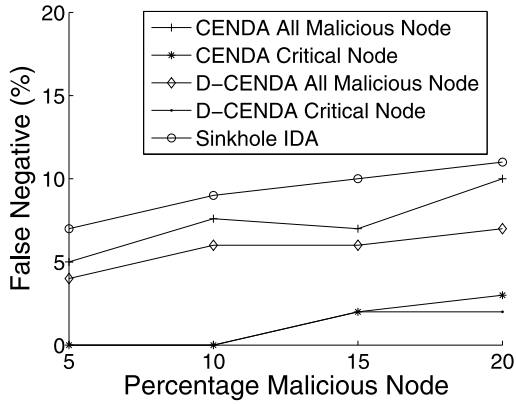


Fig. 9 Sinkhole false negative

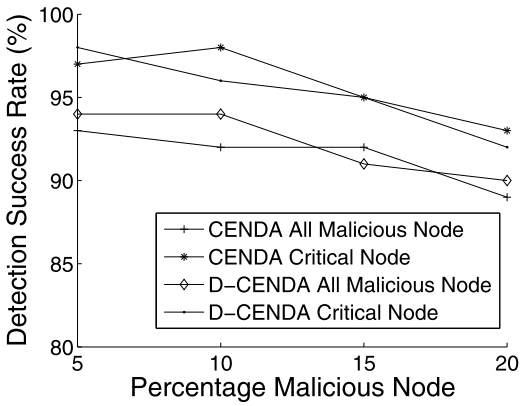


Fig. 10 Selective forwarding detection

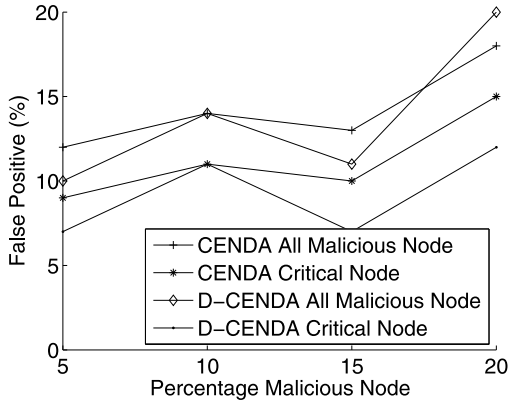


Fig. 11 Selective forwarding false positive

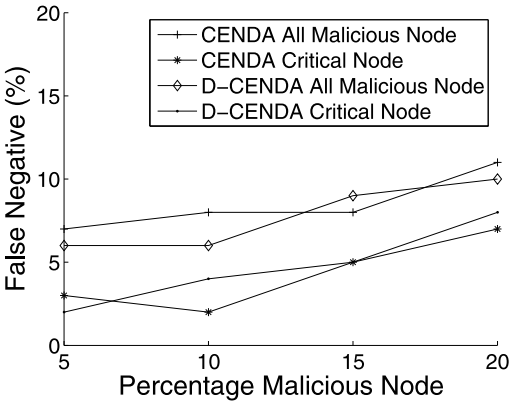


Fig. 12 Selective forwarding false negative

sinkhole. Unlike in sinkhole, the analysis to detect selective forwarding attack is performed over five rounds by the mobile-node. In other words, it was slower to catch a malicious node performing selective forwarding attack.

Figures 13, 14, 15 represents the detection success, false positives and false negatives detected for the network under sybil attack. We assumed that at a point in time, a node can acquire the identity of another node, but does not have access to cryptographic information of the node. When comparing the performance of D-CENDA with CENDA, D-CENDA performed better in most cases and no worse than CENDA in the rest.

In the next simulation, all randomly selected malicious nodes launch a wormhole attack. A wormhole attack is different from other attacks due to the fact that a single attack will have at least two colluding nodes. In the attack, randomly selected nodes collude with each other and launch the attack by directing the traffic to the other side of the network. The results for the wormhole attack is presented in Figs. 16, 17, 18. When considering all malicious nodes, CENDA outperformed D-CENDA, and while considering only the critical nodes the performance was comparable. Under the cir-

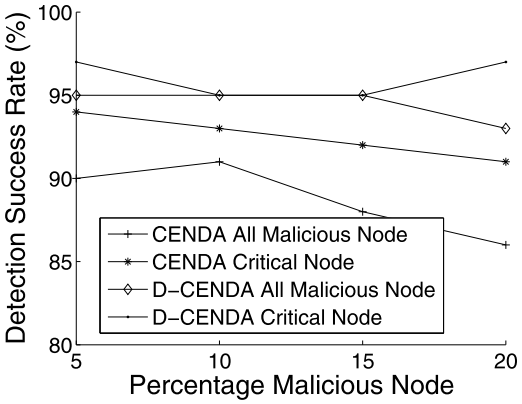


Fig. 13 Sybil detection

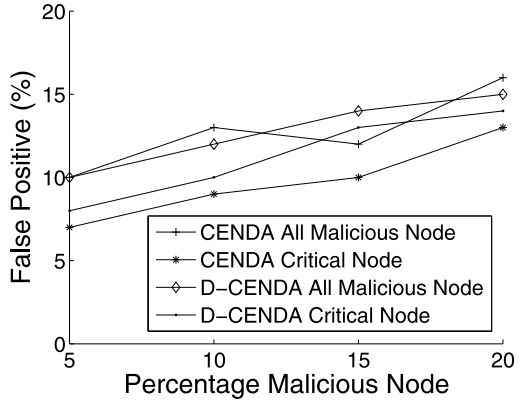


Fig. 14 Sybil false positive

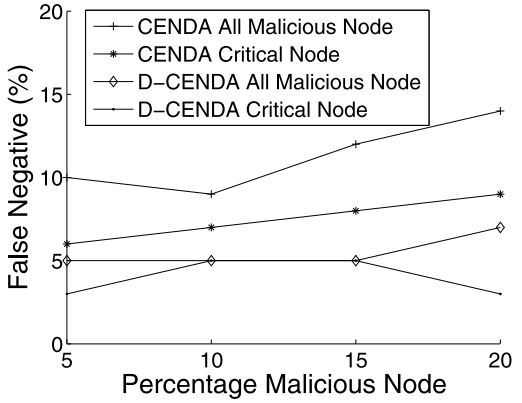


Fig. 15 Sybil false negative

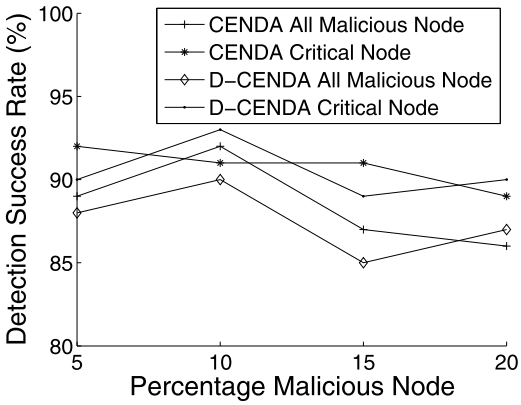


Fig. 16 Wormhole detection

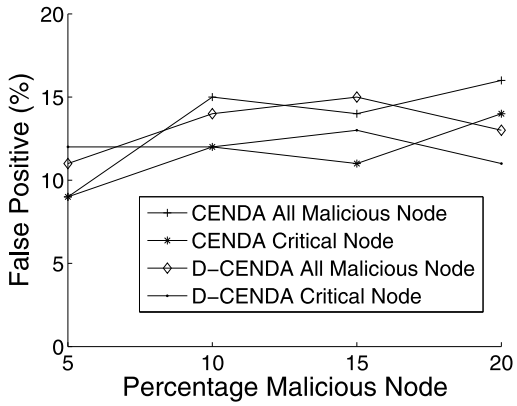


Fig. 17 Wormhole false positive

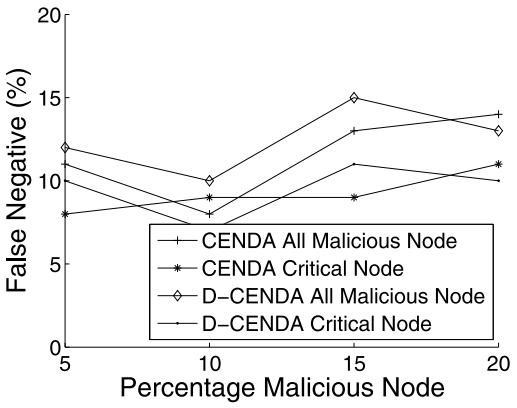


Fig. 18 Wormhole false negative

cumstance when just one of the node is directing traffic to another part of the network, the model had difficulty in identifying the other colluding node since it was only directing traffic to the basestation and was not doing anything malicious per se. This

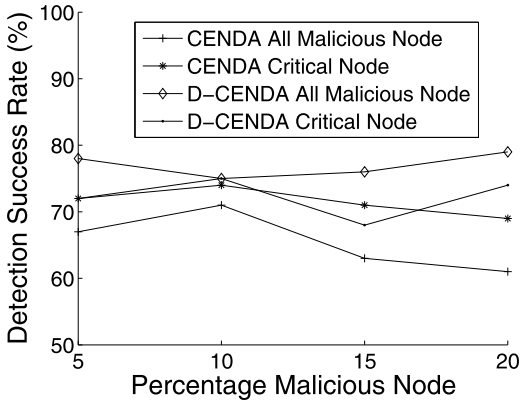


Fig. 19 Wormhole detection 1 of 2 nodes

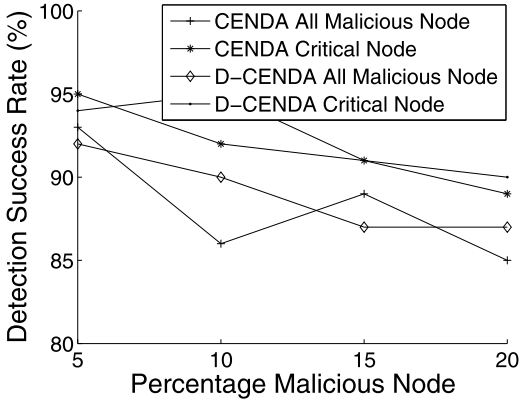


Fig. 20 All attacks detection

is depicted in Fig. 19. When only one of the nodes is launching a wormhole attack, D-CENDA outperforms CENDA. This can be accounted to the dynamic selection of critical nodes which results in looking out for the more active malicious nodes.

The next set of simulations involved malicious nodes having equal distribution of all the four attack types. The results for the same are shown in Figs. 20, 21, 22. In short term analysis, i.e. when analyzing each round of camouflage event packets, it was difficult to differentiate the sinkhole attack from selective forwarding. Analyzing the camouflage packets over multiple rounds produced sufficient information to demarcate the same. The results provided in Figs. 20, 21, 22 are for five rounds of camouflage events.

On comparing the results of D-CENDA with CENDA, we find that the results are improved while giving significant overhead savings by using a shortened relative address. With dynamic adaptation on the selection of critical nodes, the nodes on high usage paths are better protected. Hence, those nodes which are not selected as critical initially are also covered for malicious node detection. Additionally, usage of progressive feedback in D-CENDA impacts the critical nodes selected while incorporating real time information as seen in Sect. 2.2.1. This further improves the ‘all malicious

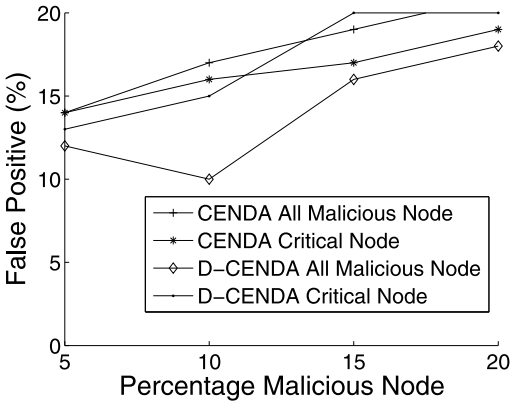


Fig. 21 All attacks false positive

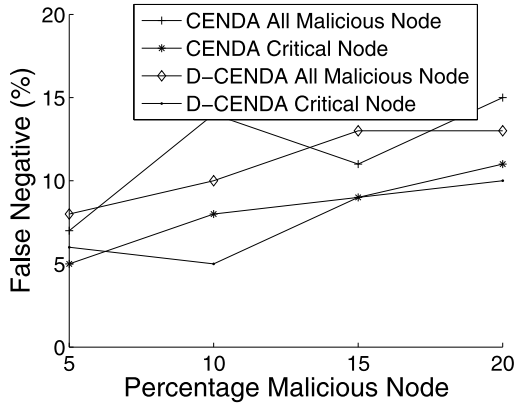


Fig. 22 All attacks false negative

node' detection in D-CENDA. We also see the speed and computational benefit of using a bit-array over bloom filter, although it consumes a little more memory.

The performance of D-CENDA is compared with the results from CHEMAS by Xiao et al. [15] and with intruder detection algorithm by Ngai et al. [6]. The simulation results in Figs. 7, 8, 9 indicate that the performance is at par with the two methods while having the advantage of being able to detect and differentiate multiple attack types at the same time. Also, with CHEMAS [15], the success of detecting the malicious nodes drops off drastically with the increase in the number of malicious nodes. For [15], in a 400 node network, the detection success percentage (in parenthesis) for the number of malicious nodes 1 (99%), 2 (97%), 3 (95%), 4 (94%), 5 (92%). It should be also noted that D-CENDA is a proactive architecture, hence the probability of catching the malicious node without any legitimate event packet loss is higher. The verification of the packet transmissions from the neighbor node helps in drastically reducing false positives in the case of sinkhole and selective forwarding attacks. Additionally, we can prevent region segregation malicious attacks. With regards to conserving energy, based on the requirement or attack type anticipated, the verification part of the system can be turned on/off. As a by product, the system

is able to detect any dead nodes due to energy exhaustion or due to environmental conditions.

8 Conclusions

D-CENDA is a proactive architecture to detect malicious nodes in the sensor network. It can be used as a compliment to an existing system and is controlled by the basestation. We use a mobile-node-based camouflage event generator scheme to overcome the problem of region segregation by malicious nodes trying to prevent event reporting. This is important in case of sporadic events wherein the basestation cannot differentiate between non-occurrence and non-report of events. The camouflage-event-based malicious node detection system uses a bloom-filter-based verification procedure before labeling a node malicious. We provide a node-classification scheme based on the role/importance of the node in the network and present a light weight path marking system using relative-addresses to indicate the path traversed by the packet to reach the basestation. The effectiveness of the system is examined using simulations and the results demonstrate this. D-CENDA cannot only detect an intrusion but also identify the malicious nodes and the type of attack. Additionally, compared to existing systems, which identify attacks of single type, we can differentiate attacks of four types and in future this can be further enhanced to identify other attack types as well.

Acknowledgements The authors would like to thank the anonymous reviewers for their helpful comments. This work was supported in part by the US National Science Foundation under grants CCF-0514078, CNS-0551464, and CNS-0721441.

References

1. Bloom BH (1970) Space/time trade-offs in hash coding with allowable errors. Commun ACM 13(7):422–426
2. Chen H, Wu H, Zhou X, Gao C (2007) Reputation-based trust in wireless sensor networks. In: MUE '07: proceedings of the 2007 international conference on multimedia and ubiquitous engineering. IEEE Computer Society, Washington, pp 603–607
3. Ganeriwal S, Srivastava MB (2004) Reputation-based framework for high integrity sensor networks. In: Proceedings of the 2nd ACM workshop on security of ad hoc and sensor networks, NY, USA
4. Karlof C, Sastry N, Wagner D (2004) Tinysec a link layer security architecture for wireless sensor networks. In: SenSys '04: proceedings of the 2nd international conference on embedded networked sensor systems. ACM, New York, pp 162–175
5. Krontiris I, Dimitriou T, Giannetsos T, Mpasoukos M (2008) Intrusion detection of sinkhole attacks in wireless sensor networks, algorithmic aspects of wireless sensor networks. Springer, Berlin
6. Ngai ECH, Liu J, Lyu MR (2007) An efficient intruder detection algorithm against sinkhole attacks in wireless sensor networks. Comput Commun 30(11–12):2353–2364
7. Pathan ASK, Hong CS (2008) Serp: secure energy-efficient routing protocol for densely deployed wireless sensor networks. Ann Télécommun 63(9–10):529–541
8. Perrig A, Szewczyk R, Tygar JD, Wen V, Culler DE (2002) Spins: security protocols for sensor networks. Wirel Netw 8(5):521–534
9. Pirzada A, Mcdonald C (2005) Circumventing sinkholes and wormholes in ad-hoc wireless networks. In: International workshop on wireless ad-hoc networks, London

10. Pongaliur K, Xiao L, Liu A (2009) CENDA: Camouflage event based malicious node detection architecture. In: Proceedings of the 2nd IEEE international symposium on trust security and privacy for pervasive apageslications (TSP 2009), Macau, China
11. Roman R, Fernandez-Gago MC, Lopez J (2007) Featuring trust and reputation management systems for constrained hardware devices. In: Proceedings of the 1st international conference on autonomic computing and communication systems, Brussels, Belgium
12. Roosta T, Shieh S, Sastry S (2006) Taxonomy of security attacks in sensor networks and countermeasures. In: Proceedings of the first IEEE international conference on system integration and reliability improvements, Hanoi
13. Su CC, Chang KM, Kuo YH, Horng MF (2005) The new intrusion prevention and detection apagesroaches for clustering-based sensor networks. In: Wireless communications and networking conference, vol 4. IEEE Press, New York, pp 1927–1932
14. Watro R, Kong D, Cuti Sf, Gardiner C, Lynn C, Kruus P (2004) Tinypk: securing sensor networks with public key technology. In: SASN '04: proceedings of the 2nd ACM workshop on security of ad hoc and sensor networks. ACM, New York, pp 59–64
15. Xiao B, Yu B, Gao C (2007) Chemas: Identify suspect nodes in selective forwarding attacks. J Parallel Distrib Comput 67(11):1218–1230
16. Xiao Y (2006) Security in sensor networks. Auerbach Publications, Boca Raton