DMN2SC: Detecting Malicious Nodes with 2-hop Secure Channel Support in Wireless Sensor Networks

Prathap U #1, Kiran K #2, Deepa Shenoy P #3, Venugopal K R #4

# *Department of Computer Science and Engineering, University Visvesvaraya College of Engineering*
Bangalore University, Bangalore, India, 560001

1 prathap.u@gmail.com
2 kirank.uvce@bub.ernet.in
3 shenoypd1@gmail.com
4 venugopalkr@gmail.com

***Abstract*—Security in wireless sensor networks is critical due to its way of open communication. In this paper we have considered suite of attacks and provided a solution to detect malicious nodes. In literature, many schemes have been proposed to mitigate such attacks but very few detect the malicious nodes effectively and also no single solution detects all attacks. In the proposed approach, each node chooses the parent node for forwarding the packet towards Sink. Each node adds its identity as a routing path marker and encrypts only the bytes added by a node in packet before forwarding to parent. Child node observes the parent, handles acknowledgement from 2-hop distance node and decides the trust on parent based on successful and unsuccessful transactions. Data transmission is divided into multiple rounds of equal time duration. Each node sends a trust value report via multiple paths to Sink at the end of each round. Sink identifies the malicious node based on the number of packets a node participates in forwarding and also based on the trust value report sent from each node for its parent. Each node chooses the parent node at the beginning of a round based on its own observation on parent to recover itself from malicious parent node. With the combination of trust factor, 2-hop acknowledgement and fixed path routing to detect malicious activity, simulation results show that proposed method detect malicious nodes efficiently and early, and also with low percentage of false detection, compared to other recently proposed approaches.**

I. Introduction

A. Wireless Sensor Networks

A wireless sensor networks (WSNs) consists of spatially distributed autonomous devices having sensing, computing and communication capabilities. Sensor nodes cooperatively monitor physical or environmental conditions, such as temperature, pressure, sound, vibration, motion or pollutants. Wireless sensor networks are used in environmental conditions where information is difficult to access. Sensor node, also known as a 'mote', is a node in a wireless sensor network that is capable of performing some processing, gathering sensory information and communicating with other connected nodes in the network. Sensor network transmits the data from one node to another node in an adhoc way and finally to a base station where the data is stored, processed and displayed.

B. Security Attacks on Sensor Nodes

Sensor nodes are vulnerable to a wide range of attacks [1][2]. Attacker can listen to radio transmissions, modify the packet before forwarding, misroute the packet to unintended next hop node, inject false data in the channel, replay previously heard packets to drain the energy of other nodes as battery power is crucial in nodes. Attacker may deploy few malicious nodes with similar or better hardware capabilities or by 'turning' few legitimate nodes by capturing them and physically overwriting their memory. Sybil attack - attacker deployed nodes may also use the identities of the other genuine nodes to frame other genuine node as malicious. Packet dropping, modification, misrouting are basic problems which have large impact on the information gathered by sensor nodes as network loses lot of important sensed data. Cryptography techniques alone are not sufficient to protect the data. Attacks such as colluding collision[3], misrouting, power control, sinkhole, wormhole[4], rushing attacks can be launched without the help of cryptography keys [5].

In packet dropping attack, the received packet is dropped without forwarding to next hop node or Sink node. In figure 1, node *Y* may drop the packet received from node *Z* without forwarding the packet to node *X*. In packet modification attack, a node changes the content of the packet and forwards to next hop node. In figure 1, node *Y* may change the bits in the packet received from node *K* and modified packet is forwarded to node *X*. In misrouting attack, a node forwards the packet to unintended next hop node either to delay the packet transmission time or to reach the packet to a non-sink node. In figure 1, node *Y* may misroute the packet to node *L* instead of routing the packet to intended next hop node *X*.

In Sybil attack[6], a malicious node uses the identities of the other genuine nodes to frame other genuine nodes as malicious. In figure 1, node *Y* may use the identity of node *K*,

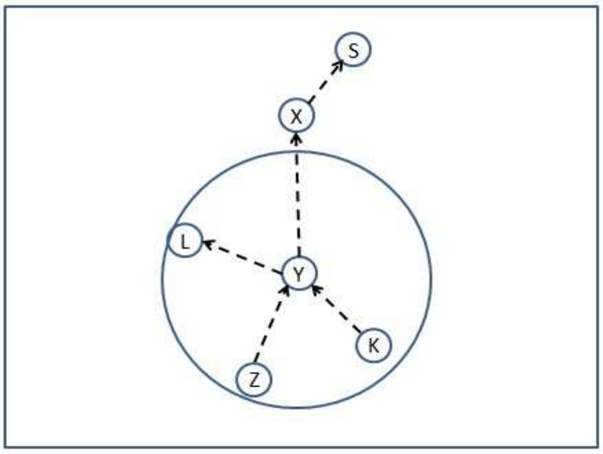

Fig. 1. Power Control Attack Description

modify the packet and send to next hop. When Sink tries to process the packet, Sink detects K as the node which modified the packet. In observation based or trust based or voting based mechanisms, a node tend to give false information about the other to frame the genuine node as a malicious node. Such attack is known as bad mouthing attack[7]. Since the proposed approach takes the support of the trust from several nodes and averages the trust to detect the malicious node, so proposed approach mitigates the effect of bad mouthing attack. In figure 1, node Y may give low trust value on its parent node X to confuse the Sink node while deciding the malicious nodes in the network.

If a node has the ability to control its power to transmit the data, then it can vary the radio range of data transmission[8]. In Figure 1, if node Y has ability to control the power to vary its data transmission distance, then node Y can use less power such that only Z and other neighbor nodes hear the packet forwarding from Y but X does not hear the packet forwarding from Y as X is farther than any other node. Sender and other neighbor nodes feel that Y has actually transmitted the packet to X but intended recipient X missed to receive the packet. If node Y is successful in achieving the power control attack for all packets to be forwarded then the node Y resembles the sinkhole attack without being detected from others. In sinkhole attack[9] malicious node attracts the routing data by publishing the shortest path to Sink and drops all the packets without forwarding further to Sink. Power control attack is a smart way of achieving the sinkhole attack.

C. *Introduction to Proposed Approach: DMN2SC*

In this paper, we propose an effective scheme, named 'Detecting Malicious Nodes with 2-hop Secure Channel Support (DMN2SC)' to identify malicious nodes which performs any or all of packet modification, packet dropping, misrouting, using wrong identity, power control attack, sinkhole attack and bad mouthing attacks. After deployment, each node selects a list of parent nodes which have equal and shortest distance to sink node. Each node selects a parent node among the identified parent nodes and sends parent selection information to Sink. Sink establishes a routing tree rooted at Sink node based on the information received from each node. Data transmission is divided into multiple rounds of equal time duration. Each node chooses a different parent node in the beginning of a round or phase among the selected parents based on the trust they have on the parent. In proposed approach every node tries to make sure that the packet is successfully delivered to 2-hop distance node and detects the malicious activity of the 1-hop distance node, thus secures the channel upto 2-hop distance node.

Intermediate node, say node Z in figure 1 prepares marker data containing node identity, encrypts the marker data and adds to the packet before forwarding the packet to parent node Y. Node Z expects acknowledgement from 2-hop distance node X. Each node builds trust value on parent by observing the parent node for malicious activities such as packet modification, dropping, sinkhole, power control and misrouting. In figure 1, node Z observes the next hop node Y to detect sinkhole, packet modification, packet dropping, misrouting attacks, and expects an acknowledgement from 2-hop distance node X to detect power control attack by 1-hop distance node Y. X does not provide acknowledgement unless it receives packet from Y. Node Z maintain the count of successful and unsuccessful packet transmission from Y based on observation and also based on 2-hop acknowledgement from X. All the nodes send the report to Sink S at the end of each round of operation using multiple paths through all selected parents.

Marker data added by each node in packet and trust report received from each node helps Sink to trace the nodes in the routing path and helps to detect malicious activities. Sink find the nodes which are responsible for malicious activity while processing the packet. If Sink fails to process the packet then uses the trust value and packet count information in report to filter the malicious node among the suspicious pair of malicious nodes. Sink aggregates the trust value collected from each child node on parent and uses the aggregate trust value to avoid bad mouthing attack on parent from a particular child node.

In order to find the packet modifiers and droppers, Catching Packet Droppers and Modifiers(CPDM) [10] has been proposed recently in literature. But CPDM frames the source node even if the intermediate node drops or modifies the packet and the percentage of false isolation is high. We proposed a solution Catching Packet Modifiers with Trust Support (CPMTS)[11] to over come the issues with CPMD. CPMTS does not consider the sybil attack and packet misrouting attack, which impacts the basic packet modifier detection mechanism of CPMTS. We proposed an enhanced solution Catching Malicious Nodes with Trust Support (CMNTS)[12] for detecting attacks not considered in CPMTS. In this paper, we have provided an integrated solution which detects suite of attacks including attacks not considered either in CPMTS or CMNTS such as power control attack and sinkhole attack. We provide a simulated performance analysis showing the comparison among CPDM, CMNTS and proposed approach with various parameters. In order to detect the sinkhole attack 'Detection of Sinkhole Attack(DSHA)[13]' has been proposed recently in the literature. DSHA first identifies the area of network where malicious node exists and then tries to find the malicious node in the identified area. we provide a simulated analysis comparing the DSHA approach and our proposed approach to evaluate the solution for sinkhole attack. The rest of the paper is organized as follows, section II discusses about the related work, section III describes the problem statement, section IV presents the solution and algorithm, section V provides the performance analysis and results, and section VI concludes the work and discusses the future challenges.

II. Related Work

To mitigate the security attacks and improve the reliability, multipath routing [14], [15], [16], [17], [18] approach has been proposed, where multiple copies of the packet are forwarded to Sink node along the different available paths. Neighbor node observation or monitoring is another approach [19], [20], [21], [22], [23], [24], [25] used to find the malicious activities such as packet modification, dropping and misrouting of the current forwarding node. In monitoring approach, observer nodes monitor the current sender and current receiver for the packet being transmitted. Observers observe for various malicious activities such as packet dropping, modification, power control, sinkhole attacks. Monitoring methods require observer nodes to buffer the packets which are forwarded to next hop node and compare the packet forwarded by next hop node with their buffered packet to find out packet modification. In CMNTS[12] both observation based and trust based techniques are used to detect the malicious nodes performing various attacks, but the approach becomes inefficient with the introduction of power control attack and sinkhole attack. In [3], [8] power control attack has been considered but the approach needs to have observer nodes in the common radio range of the current sender and receiver. In [5], an approach to detect the sinkhole attack has been proposed based on the CPU usage, but the false positive increases by detecting less utilized node as malicious. Paper[13], proposes a sinkhole detection method based on network flow information and routing pattern in the network but has high false isolation of the genuine nodes.

Energy consumption in both multipath routing and neighborhood monitoring is not affordable for sensor networks. In multipath routing, energy is consumed from nodes along multiple path to Sink, to transmit same copy of data. In monitoring approach, many nodes observe each hop while a packet being forwarded and energy of all the observer nodes consumed. In [5], energy efficient sleep-wake approach along with local monitoring method is used to detect malicious nodes but cannot control the bad mouthing attack from observers and also need enough number of observers to make this approach feasible. CPDM [10] proposed a scheme to detect packet modifiers and droppers without using multipath forwarding or monitoring approach, but the method identify the malicious nodes after long time operation of network and also has high false positive detection. CPMTS [11] proposed a scheme to overcome the issues with CPMD by making the child node to observe the parent for successful or unsuccessful transactions, but suffers from packet misrouting attack. CMNTS [12] proposed an enhanced solution of CPMTS to handle misrouting attack as well.

III. Problem Definition

Goal of the proposed DMN2SC method is to identify the nodes performing malicious activities such as packet modification, packet dropping, using wrong identity, misrouting the packet, controlling power to vary radio range(power control attack), dropping all packets in attractive routing path(sinkhole attack) and framing other nodes as bad in the wireless sensor networks. In the proposed approach child node observes the parent node for successful and unsuccessful transactions, expects acknowledgement from 2-hop distance node on the packet forwarding path, builds a trust value on parent and shares the trust value report with Sink. Sink node starts with decryption of the packet with pair wise keys shared with nodes which are in the packet forwarding path. The decryption happens with the shared keys in the reverse order of nodes in the forwarding path from Sink node to source node. Since each node adds the marker information and encrypts the added information, there are two possibilities for packet modification. case i) received packet is first modified before adding and encrypting the marker data, case ii) packet is modified after adding marker data and before forwarding the packet.

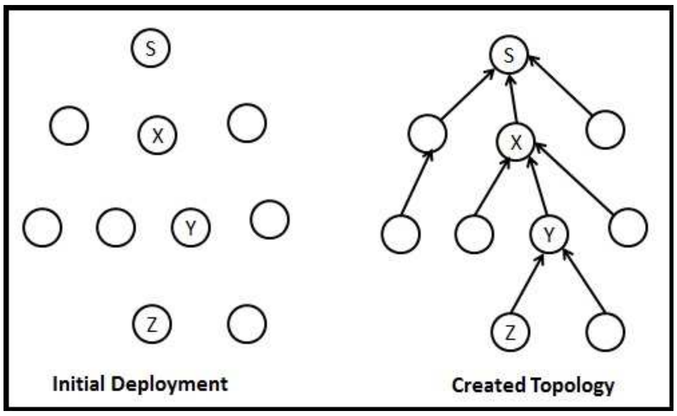

Fig. 2. Deployment and Topology

In figure 2, either node Y modifies the packet before sending to node X as in case ii or node X modifies the received packet before adding marker and forwarding further. Problem is to find the actual modifier between the pair of $< X, Y >$ nodes which are equally suspected for packet modification.

In figure 2, node Z transmits packet to Y to forward towards Sink and node Y transmits packet to node X to forward towards Sink. If node Z is a source node then following are the problems to be detected. i) If node Y performs sinkhole attack, then node Z does not hear any packet forwarding from node Y. ii) If node Y performs power control attack, then node Z hears the packet forwarding from node Y but node X does not receive the packet to be forwarded towards Sink. iii) Node *X* does not send the 2-hop acknowledgement even though received the packet successfully from node *Y*, just to frame the node *Y* as malicious. iv) In above three scenarios, either node *Y* is malicious or node *Z* is malicious as both can perform attacks and restrain from sending acknowledgement. Problem is to detect malicious nodes among such pair of nodes *<X,Y>*.

Each sensor node detects the malicious activities of parent node such as packet modification, packet dropping, power control attack, sinkhole attack and misrouting as a node cannot detect sybil attack and bad mouthing attacks from parent node. Sink detects the bad mouthing attack, sybil attack, packet modification from any node during packet decryption process and packet dropping from any parent node based on the trust value report sent from each child node. Problem is to achieve the detection capabilities of sensor nodes and also detection capabilities of Sink node. Finally use both the capabilities to detect the malicious sensor nodes in the network.

System Assumptions: i) DMN2SC assumes that the network is static and the links are bidirectional. ii) DMN2SC assumes that pair wise keys are shared between Sink and each network node before deployment. iii) Assumed no malicious activity during topology creation. iv) In DMN2SC each node knows the current *(X,Y)* location and also the location of the neighbor nodes. v) Source nodes are assumed to be genuine. vi) Assumed that during network initialization a node establishes pair wise keys with two hop distance nodes.

IV. DMN2SC

The proposed method has several steps of operation. DMN2SC starts with creating a network topology, selecting parent node, generation of traffic, and identifying the malicious nodes by Sink.

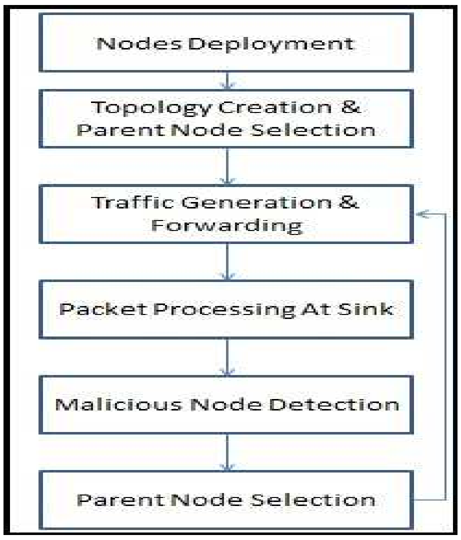
Nodes Deployment
Topology Creation & Parent Node Selection
Traffic Generation & Forwarding
Packet Processing At Sink
Malicious Node Detection
Parent Node Selection

Fig. 3. DMN2SC Operation Steps

A. Topology Creation

Sink node starts with sending a tuple $<$ *node ID, distance to Sink* $>$ = $<$ *S, 0* $>$ to all the one hop neighbor nodes. On receiving a tuple $<$ *u*, d_u $>$ = $<$ *S, 0* $>$ where *u* is the node id and d_u is the distance to Sink from node *u*, node *X* records its distance to Sink as d_u+1. If d_u+1 is less than the distance information node *X* has seen, then clears all the recorded parent list and adds node *u* to parent list and update the distance info to d_u+1. If d_u+1 is equal to the distance information node *X* has, then adds the node *u* to parent list. Intern node *X* sends a tuple $<$ *X*, d_u+1 $>$ to all its neighbor nodes.

Once all distance information is processed, every node contains smallest distance to the Sink node and also parent nodes list through which Sink can be reached with equal and least distance. Each node selects a parent among the recorded parents for transmitting the data to Sink. Each node *V* picks a random number V_s in the range 0 to N_p where N_p is the maximum number of parents recorded and uses the random number V_s as a short id of node *V*. Each node sends its ID, short id, selected parent node ID, and recorded parents list to Sink node. Based on the information received from each node, Sink builds a tree topology with all parent-child relations and uses these relations for step by step data decryption and for finding the malicious nodes. Alongside each node *V* broadcasts to its one hop neighbors about the selected parent. This helps the children of *V* to identify the misrouting attacks from *V* and helps children to handle acknowledgement from 2-hop distance node which is parent of *V*.

B. Traffic Generation and forwarding

When a source node *Z* has data to send, node *Z* creates a message m_1 = $<$ Z_s, *Z*, Z_{seq}, *D* $>$ and encrypts the message m_1 with Z_{key} to generate m_z. Where Z_s is the short ID of node Z, *Z* is ID of node, Z_{seq} is the sequence number of the packet, *D* is the data generated from source node Z, and Z_{key} is the key shared with Sink. Node *Z* sends the message m_z to parent *Y*, *Y* being intermediate node prepares marker information $<$ Y_s $>$ and encrypts with Y_{key} to create m_2, where Y_s is the short id of node *Y* and Y_{key} is the key shared with Sink. Node *Y* creates message m_y = $<$ m_2, m_z $>$ by adding encrypted marker information to m_z. Similarly all forwarding nodes add the encrypted marker information to the packet. A child node observes the parent node after dispatching the packet to parent for a timeout period to determine the packet modification, dropping, sinkhole and misrouting attacks from parent. And handles acknowledgement from 2-hop distance node to detect the power control attack. A node maintains the count of packets forwarded to parent and sends the count information to Sink in the trust value report.

i) When parent node *Z* sends the packet to next hop node *Y*, node *Z* buffers the sent packet and observes node *Y*. When node *Y* forwards the packet to node *X*, node *Z* as well receives a copy of the packet, removes marker data added by node *Y* and compares the content with its buffer. If there is a mismatch, node *Z* determines the packet modification from node *Y* and reduces the trust value on node *Y*.

ii) If node *Z* does not hear any packet forwarding from node *Y* in a timeout period, then node *Z* determines the packet dropping and sinkhole attack from node Y and reduces the trust value on Y.

iii) When node Z hears the packet forwarding from node Y, node Z determines the misrouting attack from node Y and reduces trust value, If node Y forwards the packet to unintended next hop node as node Z has knowledge of 2-hop distance node.

iv) Even if node Z hears the packet forwarding from node Y, node Z waits to receive acknowledgement from 2-hop distance node X. If no acknowledgement is received, then node Z determines the power control attack from node Y as node X has not received the packet to acknowledge back. Node Z reduces the trust value on Y.

v) If node X does not provide the acknowledgement back due to its malicious activity. Node Y reduces the trust value on node X.

C. Packet Processing at Sink

The received packet at the Sink consists of sequence of marker information added by each forwarding node and message from source node. On receiving a data packet m, Sink starts decryption of the packet.

i) First marker information of message m is decrypted with key of first level child node say X of Sink to generate m'. If m' starts with $< X_s >$ then X is the forwarded node. Else Sink decrypts with key of next immediate first level child node and tries to match the marker information.

ii) If marker information does not match with any of the first level children, then Sink decrypts the complete message with key of first level child say X to generate m'. If m' starts with $< X_s$, X $>$ then X is the source node. Else Sink decrypts with key of next first level child node and tries to check for source.

iii) If marker matches a node say X in step i, then m' is updated $m' = m' - < X_s >$ by removing the marker added by X. Sink increments the packet forwarded count for node X. This count value is compared with packet count value in the trust report sent by each node later to detect malicious nodes. Now the step i and step ii are performed for all children of X to match for forwarding node or source node.

iv) if step i and step ii fails for all children nodes at same level, that confirms the packet modification either from current parent or any child of the current parent. So the suspicious pair $<$ *parent, child* $>$ is added to suspicious list for all immediate children nodes of the parent.

v) In step i and step ii after decryption, if short ID in packet say V_s does not match with the node whose key used for decryption then Sink checks whether other siblings of node V has the matching short id. With this, Sink can detect the usage of others identity by node V. Sink maintains the count of wrong identity usage for every node. If count exceeds a threshold then such a node is declared to be performing sybil attack.

Notations:

m: received packet at Sink
U, V, S: node id
V_{key}*: shared key between Sink and node V*
V_s*: short id of node V*
success: boolean to track successful decryption
V_{pcount}*: packet count maintained by Sink for node V*
V_{scount}*: sybil attack count maintained by Sink for node V*

Algorithm 1: Packet Processing at Sink

```
1: Input: Packet <m>
2: U = S, m = m; success = false;
3: for each child node V of node U do
4:     P = decMarker($V_{key}$, m); /*decrypts only marker which is one unit*/
5:     if P starts with [$V_s$] then
6:         $V_{pcount}$++;
7:         trim [$V_s$] from m and get m = m-[$V_s$];
8:         U = V; go to line 3;
9:     else if P starts with siblingID(V) then
10:        $V_{scount}$++;
11:        add suspicious pair <U, V> to suspicious list;
12:        goto line 3;
13:    endif
14: endfor
15: for each child node V of node U do
16:     P = decSourceMsg($V_{key}$, m); /*decrypts source message which is four units*/
17:     if P starts with [$V_s$, V] then /*V is the source node*/
18:         $V_{pcount}$++;
19:         success = true; break;
20:     endif
21: endfor
22: if success = false then
23:     drop this packet;
24:     for each child node V of node U do
25:         add suspicious pair <U, V> to suspicious list;
26:     endfor
27: endif
```

The malicious node is identified from suspicious pairs with the help of trust value report sent from each node. During the success of step i and step ii, Sink records the packet count value for each node which participated in forwarding a packet.

D. Identifying Malicious Node

Each node prepares report containing a tuple $<$ V, V_c, P_v, T_p $>$ where V is node id, V_c is count of the packets, node V has forwarded and generated, P_v is the parent in the current round of operation, T_p is the trust value on parent P. Copies of the report is sent to Sink node through all the parent nodes selected during initialization of network. Multiple copies are sent to make sure atleast one copy of the report reaches Sink node in the presence of malicious nodes which perform the sinkhole attack. Sink considers one copy even though it receives more than one copy of the same report.

After a round of traffic generation, Sink has a list of suspicious pair $<$ *Parent Node, Child Node* $>$ of nodes and also trust report on a parent from their child nodes. For each parent node Sink does the below.

i) Sink calculates the total number of packets a parent must have forwarded by adding the packet count in trust value report received from each child. Sink compares the packet count found from reports with the packet count maintained during packet processing by Sink node. If the difference in the packet count is greater than the threshold, Sink compares the average trust of the parent node. If the average trust is less than the predefined threshold then parent node is marked as malicious node. Sink also checks sybil attack count for each node, if sybil attack count is greater than sybil attack threshold, then node is declared as malicious.

Notations:

R: vector of reports collected from each node
R_v: report sent by node V
R_v[T]: trust value in the report sent by node V
U, V: node id
$U_{packetcount}$: sum of packet counts from all children of U
U_t: sum of the trust value from all children U
U_{avgt}: average trust value of U
$U_{children}$: total number of children of U
dropthreshold: dropping threshold due to environmental errors
U_{pcount}: packet count maintained by Sink for node U
V_c: count of packets node V has sent in report
$T_{threshold}$: trust threshold
$T_{sthreshold}$: sybil attack threshold
V_{scount}: sybil count attack from node V

Algorithm 2: malicious node detection at Sink
```
1: for each node U do
2:     $U_{packetcount}$ = 0;
3:     $U_{avgt}$ = 0;
4:     for each child V of U do
5:         $R_v$ = R[V];
6:         $U_{packetcount}$ = $U_{packetcount}$ + $R_v[V_c]$;
7:         $U_t$ = $U_t$ + $R_v$[T];
8:     endfor;
9:     $U_{avgt}$ = $U_t$ / $U_{children}$;
10:    if ($U_{packetcount}$ - $U_{pcount}$ > dropthreshold) then
11:        if ($U_{avgt} < T_{threshold}$) then
12:            mark node U as malicious;
13:    if ($U_{scount} > T_{sthreshold}$) then
14:        mark node U as malicious;
15:endfor;
```

ii) Among nodes in the suspicious pair, if a node is not declared as malicious in step *i*, Sink validates the average trust value of a node. If average trust value is less than the threshold, then node is declared as malicious node. If average Trust value is greater than the threshold then find a child whose average trust value is less than the threshold. If such a child is found, then child is the malicious node. If average trust values of both parent and children are greater than threshold then they are still suspicious pairs but not malicious yet.

Notations:

*SPairs: set /*set of tuples <ParentId, ChildId>, identified suspicious pairs*/*
threshold: pre-declared system level threshold value
ParentId, ChildId: node id
AvgTrust: function averages the trust from all children of a node

Algorithm 3: Malicious Node Identification
```
1: for each pair SPair in SPairs do
2:     if AvgTrust(SPair.ParentId) < threshold then
3:         Declare SPair.ParentId as Malicious;
4:     else if AvgTrust(SPair.ChildId) < threshold then
5:         Declare SPair.ChildId as Malicious;
6:     else
7:         do nothing
8:         /*both child, parent are still suspicious, need to handle
more packets in next round to identify*/
```

E. Changing Parent For Next Round

Traffic generation happens in several equal duration rounds of malicious node identification phases. After a round, child chooses a parent as per the below priority. Even Sink follows the same priority to know parent based on the parents list sent by each node. i) Child selects the next parent in its list with which it never had an interaction. ii) Child selects the parent for which trust value is high. Alongside each node say *Y* broadcasts to its one hop neighbors about the selected parent *X*. This helps the children of *Y* to identify the misrouting attacks from *Y*.

Notations:

selected: boolean
*ParentIds: set /*parent node ids*/*
*ParentID: node id /*selected parent id in this round*/*
*TempParentIds: set /*parent ids whose trust is greater than threshold*/*

Algorithm 4: Parent Selection at Node
```
1: selected = false;
2: for each ID in ParentIds do
3:     if ID was never a forwarding node then /*select a parent
node which never chosen for data forwarding*/
4:         selected = true;
5:         ParentID = ID;
6:         break;
7: if selected == false then
8:     for each ID in ParentIds do
9:         if Trust of ID >= Threshold then
10:            add ID to TempParentIds;
11: ParentID = Random (TempParentIds);/*select any node
whose trust greater than threshold*/
12: broadcast ParentID to one-hop neighbours
```

V. Performance Analysis

The efficiency and effectiveness of DMN2SC are evaluated in NS-3 simulator. We have compared proposed approach with CPDM [10] and CMNTS [12]. 100 static nodes are randomly deployed in a square area. Each node is installed with 802.15.4 MAC protocol, with channel delay 2 milli seconds. Simulation ran with generating 50 packets per node in each round. Non leaf nodes are randomly selected as malicious nodes. All nodes act as a source node and generate the data to forward towards Sink. Obtained simulation results from the algorithm for various number of malicious nodes. Malicious behavior of nodes achieved with equal probability of packet modification, packet dropping, misrouting, using wrong identity, framing parent with low trust, power control attack, sinkhole attack and successful transmission. It is observed that performance of both CPDM and CMNTS degrades with the injection of all attacks considered in the DMN2SC approach.

A. *Percentage of Detection*

Simulated and analyzed the detection rate when the number of malicious nodes are 10, 20, 30, and 40 out of 100 nodes in the network.

P: Number of malicious nodes detected
Q: Number of malicious nodes in network
R: Percentage of detection
*R=(P/Q)*100*

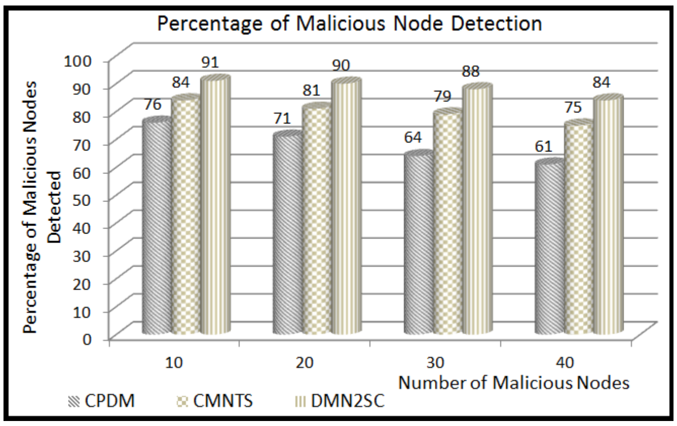

Fig. 4. Percentage of malicious node detection

For each quantity of malicious nodes, traffic is generated in 5 trails and averaged the detected malicious nodes in 5 trails. As shown in figure 4, percentage of detection is improved in DMN2SC when compared to CPDM and CMNTS approaches. In CPDM, the percentage of detection deteriorates as the number of malicious nodes increases. The improved performance of the DMN2SC is due to the handling of power control and sinkhole attacks.

B. *Percentage of False Isolation*

Simulated and analyzed the false detection when the number of malicious nodes are 10, 20, 30, and 40.

P: Number of genuine nodes isolated
Q: Number of genuine nodes in network
R: Percentage of false detection
*R=(P/Q)*100*

As shown in figure 5, percentage of false detection is high in CPDM approach. In CPDM approach, even though intermediate node modifies the packet, Sink considers the

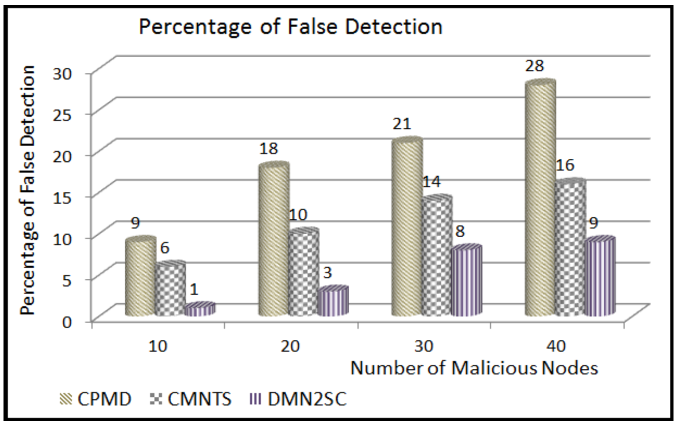

Fig. 5. Percentage of false isolation

source node for malicious detection as the approach is solely based on sequence number of the packet. CMNTS reduces the false isolation compared to CPDM, but false isolation increases on injecting power control and sinkhole attacks. In proposed approach, only the current parent and children where the packet decryption fails are considered for identifying the malicious node. And considered trust from all children node to avoid bad mouthing attack from a particular child which tries to frame the parent as malicious by sending low trust value to Sink. 2-hop acknowledgement helps in detecting power control attacks from parent node. Trust value report helps Sink to identify the packet droppers with the difference in the packet count forwarded by a node with the packet count processed by Sink.

C. *Early Detection Rate*

Simulated and analyzed the early detection when the number of malicious nodes are 20. In all CPDM, CMNTS, and DMN2SC traffic is generated in multiple rounds of equal duration and tries to find the malicious nodes after each round. CPDM needs several rounds of operation to confirm the bad nodes among suspiciously bad nodes. CPDM cannot detect most bad nodes after each round as it suspects many nodes on the path from source to Sink. Both CPDM and CMNTS cannot detect malicious nodes if node performs power control and sinkhole attacks.

As shown in figure 6, DMN2SC detects the malicious nodes early compare to CPDM and CMNTS so that network cannot afford to loose lot of meaningful information before all malicious nodes are detected.

D. *Percentage of Detection for Sinkhole Attack*

Simulated and analyzed the detection rate when the number of malicious nodes are 10, 20, 30, and 40 out of 100 nodes in the network for sinkhole attack. CPMD and CMNTS approaches do not detect the sinkhole attack. Analysis is made by simulating sinkhole attack only in DMN2SC to compare the results with DSHA[13] approach which detects the Sinkhole attack.

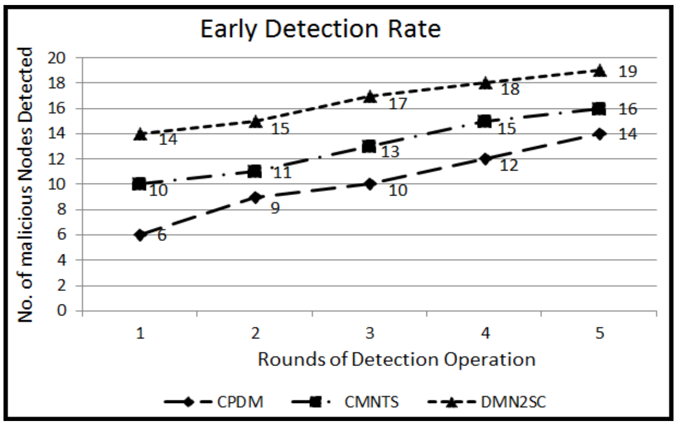

Fig. 6. Early Detection Rate

P: Number of malicious nodes detected
Q: Number of malicious nodes in network
R: Percentage of detection
*R=(P/Q)*100*

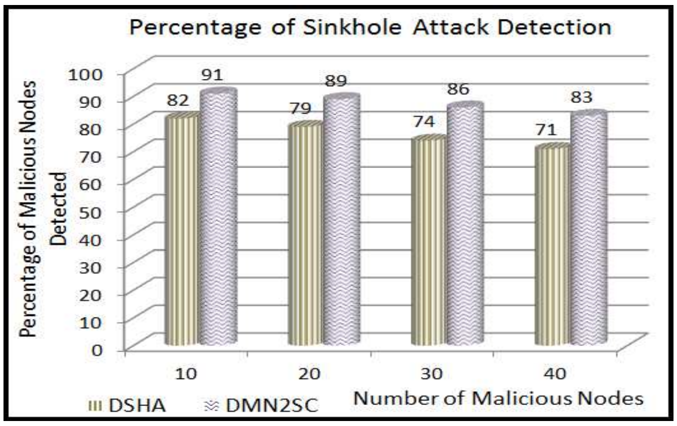

Fig. 7. Percentage of malicious node detection for sinkhole attack

For each quantity of malicious nodes, traffic is generated in 5 trails and averaged the detected malicious nodes in 5 trails. As shown in figure 4, percentage of detection is improved in DMN2SC when compared to DSHA[13] approach. In DSHA, the percentage of detection deteriorates as the number of malicious nodes increases. The improved performance of the DMN2SC is due to handling 2-hop acknowledgement and also collective decision from Sink node.

E. Analysis of Various Security Attacks

Table 1 shows the different approaches and the list of attacks considered to detect the malicious nodes. Each node say *Z* in figure 2 sends the packet to parent *Y* and observes *Y*, till *Y* forwards the packet to next hop node *X*.

Packet Modification: Node *Z* keeps the packet in buffer till parent *Y* forwards the packet to next hop and listens to the packet that *Y* forwards. *Z* compares the trailing bits(without path marker) in the packet forwarded by *Y* with the packet in buffer. If there is any change in the forwarded packet then *Z* determines that parent *Y* modified the packet and accordingly reduces the trust. Even Sink adds both child and parent into a suspicious pair list when the packet decryption process fails.

Packet Dropping: Node *Z* keeps the packet in buffer, if node *Z* does not hear the packet forwarding from parent *Y* with in pre-configured timeout then *Z* determines the packet dropping from *Y* and accordingly reduces the trust on parent *Y*.

TABLE I
SECURITY ATTACKS COMPARISON

Attack Type	CPDM	DSHA	CMNTS	DMN2SC
Packet Modification	Yes	No	Yes	Yes
Packet Dropping	Yes	No	Yes	Yes
Sybil Attack	No	No	Yes	Yes
Bad Mouthing Attack	No	No	Yes	Yes
MisRouting Attack	No	No	Yes	Yes
Power Control Attack	No	No	No	Yes
Sinkhole Attack	No	Yes	No	Yes

Packet misrouting: Packet misrouting is an attack where a node forwards the packet to unintended next hop node. Before starting a round of traffic generation each node announces its parent node information with one hop neighbour nodes. In figure 2, When *Y* announces its selected parent *X* to one hop neighbour nodes, node *Z* maintains the next hop node *X* of selected parent *Y* in memory along with selected parents list. Node *Z* compares the next hop node id *X* with the node id to which *Y* forwards the packet to identify the packet misrouting from parent and consider for calculating the trust value. Even packet decryption process at Sink fails and adds the genuine nodes to suspicious pair list. But the trust on genuine nodes saves them from being framed as malicious.

Sybil Attack: A node uses wrong identity or others identity to frame other node as malicious. In DMN2SC approach while adding the marker information, malicious node can add wrong identity. The packet description process at Sink detects that marker is not matching with any children at same level and add the nodes to suspicious pair list.

Bad Mouthing Attack: Even though a node intentionally shares low trust on parent with Sink, Sink consider the average trust from all children to suppress the bad mouthing attack.

Power Control Attack and Sinkhole Attack: If a node does power control attack for every packet then it resembles the sinkhole attack. Every node has a information about the 2-hop distance node. When a node *Z* forwards the packet to parent node *Y*, node *Z* expects an acknowledgement from 2-hop distance node *X*. If node *X* does not receive any packet due to power control attack from node *Y*, then node *X* does not provide any acknowledgement back to *Z* via *Y*. Thus node *Z* determines the power control attack and sinkhole attack from parent node *Y* and reduces the trust value on parent *Y*. If node *X* intentionally does not provide the acknowledgement back due to its malicious behavior, then node *Y* reduces the trust value on its parent *X*.

F. Additional Cost Involved

The overall packet size is kept constant to avoid a node to perform selective dropping attack based on packet size. So parent node cannot decide to drop a packet received from two or more different children based on packet size as the received packet sizes from different children are same. This adds overhead of transmitting lot of extra data added at each forwarding node. Each forwarding node adds one byte of data in the beginning of the packet and removes one byte in the tail end. Packet size depends on the maximum hop distance from a child to sink node so that even after adding one byte of data and removing one byte of data at each forwarding node the packet size remains same.

VI. Conclusions and Future Directions

Sensor nodes with malicious activities disrupt the data and operations in wireless sensor networks. With the malicious activities, aggregated sensed data becomes meaningless due to dropping the valid data and injecting the wrong data. Proposed method is proven to be efficient integrated solution to detect security attacks such as packet modification, dropping, misrouting, using wrong identity, power control attack, sink hole attack and bad mouthing attack. DMN2SC starts with creating a tree topology having parent-child relation information in Sink node. Data transmission happens across multiple rounds of equal time duration. Each node chooses its parent node at the beginning of a round. DMN2SC identifies bad nodes after each round and keeps few suspiciously bad nodes till the completion of next round. At the end of each round, Sink in DMN2SC approach receives a report from each node containing the packet count and trust value information and tries to find the bad nodes from suspiciously bad nodes. But DMN2SC detects suspiciously bad nodes during each packet decryption process if packet decryption fails and identifies the most bad nodes after each round of operation. Performance results show that DMN2SC detects the malicious nodes early with high detection rate and low false detection compared to CPMD and CMNTS. DMN2SC can be further improved to apply the solution to hierarchical topology(Clustered) and to optimize the overall packet size in the network.

References

[1] H. Chan, and A. Perrig, "Security and Privacy in Sensor Networks," In *Computer*, volume 36, pp. 103–105, Oct 2003.

[2] I. Butan, S. D. Morgera, and Ravi Sankar, "A Survey of Intrusion Detection Systems in Wireless Sensor Networks," In *IEEE Communications Surveys & Tutorials*, volume 16, no. 1, pp. 266–282, First Quarter 2014.

[3] I. M. Khalil, and S. Bagchi, "Stealthy Attacks in Wireless Ad Hoc Networks: Detection and Countermeasure," In *IEEE Transactions On Mobile Computing*, volume 10, no. 8, pp. 1096–1112, August 2011.

[4] M. Bendjima, and M. Feham, "Wormhole Attack Detection in Wireless Sensor Networks," In *Proc. SAI Computing Conference*, pp. 1319–1326, July 2016.

[5] I. M. Khalil, "ELMO: Energy Aware Local Monitoring in Sensor Networks," In *IEEE Transactions on Dependable and Secure Computing*, volume 8, no. 4, pp. 523–536, August 2011.

[6] R. John, J. P. Cherian, and J. J. Kizhakkethottam, "A Survey of Techniques to Prevent Sybil Attacks," In *Proc. IEEE ICCCI*, Jan 2015.

[7] X. Li, F. Zhou, and J. Du, "LDTS: A Lightweight and Dependable Trust System for Clustered Wireless Sensor Networks," In *IEEE Transactions on Information Forensics and Security*, volume 8, no. 6, pp. 924–935, June 2013.

[8] I. M. Khalil, "MPC: Mitigating Stealthy Power Control Attacks in Wireless Ad Hoc Networks," In *IEEE Global Telecommunications Conference*, pp. 1–6, August 2009.

[9] C. Chen, M. Song, and G. Hsieh, "Intrusion Detection of Sinkhole Attacks In Large-scale Wireless Sensor Networks," In *IEEE International Conference on Wireless Communications, Networking and Information Security (WCNIS)*, pp. 711–716, June 2010.

[10] C. Wang, T. Feng, J. Kim, G. Wang, and W. Zhang, "Catching Packet Droppers and Modifiers in Wireless Sensor Networks," In *IEEE Transactions on Parallel and Distributed Systems*, volume 23, no. 5, pp. 835–843, May 2012.

[11] Prathap U, P Deepa Shenoy, and Venugopal K R, "CPMTS:Catching Packet Modifiers with Trust Support in Wireless Sensor Networks," In *Proc. IEEE WIECON-ECE* , pp. 255–258, Dec 2015.

[12] Prathap U, P Deepa Shenoy, and Venugopal K R, "CMNTS: Catching malicious nodes with trust support in wireless sensor networks," In *Proc. IEEE TENSYMP* , pp. 77–82, May 2016.

[13] Ahmad S S, M. A. Razzaque, Parisa N, and A. Farrokhtala, "Detection of Sinkhole Attack in Wireless Sensor Networks," In *Proc. IEEE International Conference on Space Science and Communication*, pp. 361–365, July 2013.

[14] C. Karlof, and D. Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures," In *Proc. IEEE First Int. Workshop Sensor Network Protocols and Applications*, pp. 113–127, 2003.

[15] V. Bhuse, A. Gupta, and L. Lilien, "DPDSN: Detection of Packet-Dropping Attacks for Wireless Sensor Networks," In *Proc. Fourth Trusted Internet Workshop*, 2005.

[16] M. Kefayati, H.R. Rabiee, S.G. Miremadi, and A. Khonsari, "Misbehavior Resilient Multi-Path Data Transmission in Mobile Ad-Hoc Networks," In *Proc. Fourth ACM Workshop Security of Ad Hoc and Sensor Networks (SASN 06)*, 2006.

[17] R. Mavropodi, P. Kotzanikolaou, and C. Douligeris, "SECMRa Secure Multipath Routing Protocol for Ad Hoc Networks," In *Ad Hoc Networks*, volume 5, pp. 87-99, 2007.

[18] Pavithra B, and K. Satyanarayan R, "Energy Efficient Detection of Malicious Nodes Using Secure Clustering With Load Balance and Reliable Node Disjoint Multipath Routing in Wireless Sensor Networks," In *Proc. IEEE ICACCI,*, pp. 954–958, 2015.

[19] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical En-Route Filtering of Injected False Data in Sensor Networks," In *Proc. IEEE INFOCOM,*, pp. 839–850, 2004.

[20] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "An Interleaved Hop-by-Hop Authentication Scheme for Filtering False Data in Sensor Networks," In *Proc. IEEE Symp. Security and Privacy*, pp. 259–271, 2004.

[21] H. Yang, F. Ye, Y. Yuan, S. Lu, and W. Arbaugh, "Toward Resilient Security in Wireless Sensor Networks," In *Proc. Sixth ACM Int. Symp. Mobile Ad Hoc Networking and Computing (MobiHoc 05)*, 2005.

[22] Prathap U, Nisha K B, P Deepa Shenoy, and Venugopal K R, "SDLM: Source detection based local monitoring in wireless sensor networks," In *Proc. IEEE TENCON* , pp. 1–5, Nov 2015.

[23] S. Lim and L. Huie, "Hop-by-Hop Cooperative Detection of Selective Forwarding Attacks in Energy Harvesting Wireless Sensor Networks," In *Proc. IEEE International Conference on Computing, Networking and Communications (ICNC)*, pp. 315–319, 2015.

[24] K. Gerrigagoitia, R. Uribeetxeberriay, U. Zurutuzaz, and I. Arenaza, "Reputation-based Intrusion Detection System for wireless sensor networks," In *Proc. IEEE Complexity in Engineering (COMPENG)*, pp. 1-5, June 2012.

[25] L. Ju, H. Li, Y. Liu, W. Xue, K. Li, and Z. Chi, "An Improved Intrusion Detection Scheme based on Weighted Trust Evaluation for Wireless Sensor Networks," In *Proc. IEEE 5th International Conference on Ubiquitous Information Technologies and Applications*, pp. 1-6, 2010.