CMNTS:Catching Malicious Nodes with Trust Support in Wireless Sensor Networks

Prathap U, Deepa Shenoy P and Venugopal K R
Department of Computer Science and Engineering
University Visvesvaraya College of Engineering
Bangalore University, India
prathap.u@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 - packet modification, packet dropping, sybil attack, packet misrouting, and bad mouthing attack, and provided a solution to detect attacks. 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 and trust on parent as a routing path marker and encrypts only the bytes added by node in packet before forwarding to parent. Sink can identify the malicious node based on trust value and node identities marked in packet. Child node observes the parent 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 chooses the parent node at the beginning of a round based on its own observation on parent. Simulated the algorithm in NS-3 and performance analysis is discussed by comparing the results with other two recently proposed approaches. With the combination of trust factor 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.**

***Keywords*—WSN, trust based, malicious node, packet modification, sybil attack, misrouting, packet dropping, bad mouthing attack**

I. Introduction

A wireless sensor network (WSN) 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.

Sensor nodes are vulnerable to a wide range of attacks [1]. 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[2], misrouting, power control, wormhole, rushing attacks can be launched without the help of cryptography keys [3].

In this paper, we propose simple yet effective scheme 'Catching Malicious Nodes with Trust Support (CMNTS)' to identify malicious nodes which performs any or all of packet modification, packet dropping, misrouting, using wrong identity, 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 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.

Intermediate node prepares marker data containing node identity and trust factor on its parent node, encrypts the marker data and adds to the packet before forwarding the packet to parent node. Each node builds trust value on parent by observing the parent node for malicious activities such as packet modification, dropping and misrouting. Marker data added by each node helps sink to trace the nodes in the routing path and helps to detect sybil attacks, misrouting and packet modification. Sink finds the nodes which are responsible for malicious activity while processing the packet. If Sink fails to process the packet then uses the trust value to filter the malicious node among the suspicious pair of nodes. Sink uses aggregated trust value collected by each child node on parent to avoid bad mouthing attack.

In order to find the packet modifiers and droppers, Catching Packet Droppers and Modifiers(CPDM) [4] has been proposed

978-1-5090-0931-2/16/$31.00 ©2016 IEEE

recently in literature. But CPDM frames the source node even 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)[5] 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. The proposed approach CMNTS in this paper provide a solution for detecting attacks not considered in CPMTS. We provide a simulated performance analysis showing the comparison among CPDM, CPMTS and our approach with various parameters. 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 handle the packet droppers and modifiers, multi-path routing [6], [7], [8], [9] approach is widely adopted in which copies of a packet are forwarded along multiple paths to the destination Sink. Neighbor node observation or monitoring is another approach [10], [11], [12] used to find the packet modifiers, droppers and routing misbehavior in sensor networks. In monitoring approach, nodes monitor their neighborhoods promiscuously to collect information about the behaviors to identify the malicious activity and take future forwarding decisions. Monitoring method requires nodes to buffer the packets which are forwarded to next hope node and compares the packet forwarded by next hop node with its buffered packet to find out packet modifications.

Energy consumption in both multipath routing and neighborhood monitoring is not affordable for sensor networks since, many nodes observe each hop while a packet being forwarded. In [3], 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. CPDM [4] 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 [5] 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.

III. Problem Definition

Goal of the proposed CMNTS method is to identify the nodes with malicious activities such as packet modification, packet dropping, using wrong identity, misrouting the packet and framing other nodes as bad in the wireless sensor networks. Child node observes the parent node for successful and unsuccessful transactions, builds a trust value on parent and shares the trust value 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.

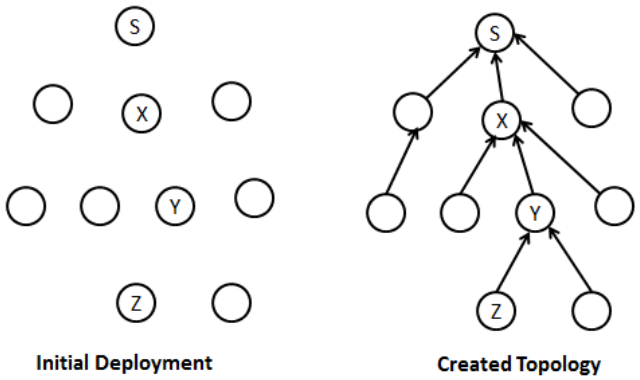

Fig. 1: Deployment and Topology

In figure 1, 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. Each node detects the malicious activities of parent node such as packet modification, packet dropping and misrouting as a node cannot detect sybil attack and bad mouthing attacks from parent node. Sink detects the bad mouthing attack, sybil attack, and packet modification from any node during packet decryption process. 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: CMNTS assumes the network is static and the links are bidirectional. CMNTS assumes that pair wise keys are shared between Sink and each network node before deployment. Assumed no malicious activity during topology creation. In CMNTS each node knows the current *(X,Y)* location and also the location of the neighbor nodes. Source nodes are assumed to be genuine. Assumed that packet forwarding node uses the transmission power level such that both current sender and next hop node hear the packet transmission.

IV. CMNTS

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

A. Topology Creation

Sink node starts with sending a tuple $<$ node ID, distance to Sink $> = < S, 0 >$ to all the one hop neighbor nodes.

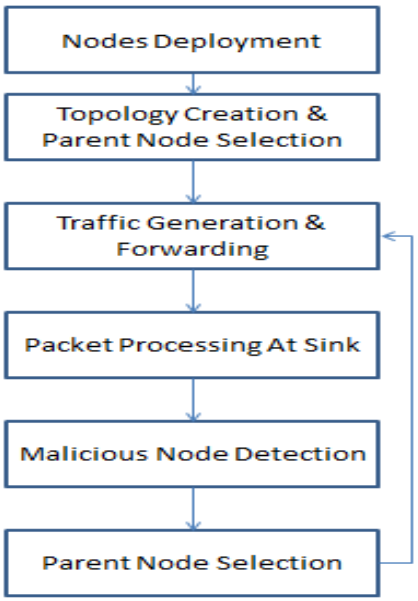

Fig. 2: Steps of Operation

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 this 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*.

B. Traffic Generation and forwarding

When a source node *Z* has data to send, node *Z* creates a message $m_1 = < Z_s, Z, T_y, Z_{seq}, D >$ 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, T_y is node *Z*'s trust value on parent *Y*, 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 $< Y_s, T_x >$ and encrypts with Y_{key} to create m_2, where Y_s is the short id of node *Y*, T_x is the node *Y*'s trust value on parent *X*, 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 node's adds 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, and misrouting attacks from parent.

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 $< X_s, T_s >$ 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 $m^{'} = m^{'} - < X_s, T_s >$ by removing the marker added by *X*. 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*.

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

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 two units*/*
5: ***if*** *P starts with [V_s, T]* ***then***
6: record the T on U from V;
7: trim [Vs, T] from P and get m = P-[V_s, T];
8: U = V; go to line 3;

9: else
*10: **if** P starts with siblingID(V) **then***
11: add suspicious pair <U, V> to suspicious list;
*12: **continue;***
*13: **for** each child node V of node U **do***
*14: P = decSourceMsg(V_{key}, m); /*decrypts source message which is five units*/*
*15: **if** P starts with [V_s, V] **then** /*V is the source node*/*
16: record the T on U from V;
*17: success = true; **break**;*
*18: **if** success = false **then***
19: drop this packet;
*20: **for** each child node V of node U **do***
21: add suspicious pair <U, V> to suspicious list;

The malicious node is identified from suspicious pairs with the help of trust value recorded for each node. During the success of step *i* and step *ii*, Sink records the trust value shared by child on its parent.

D. Identifying Malicious Node

After a round of traffic generation, Sink has a list of suspicious pair < Parent Node, Child Node > of nodes and also trust on a parent from their child nodes. For each parent node Sink does the below. i) Calculate the average trust based on the trust value received from each child. If average trust value is less than the threshold, then Parent node is the malicious node. ii) If average Trust value is greater than the threshold then find a child whose average trust value less than the threshold. If such a child is found, then child is the malicious node. iii) 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 2: Malicious Node Identification
*1: **for** each pair SPair in SPairs **do***
*2: **if** AvgTrust(SPair.ParentId) < threshold **then***
3: Declare SPair.ParentId is Malicious;
*4: **else if** AvgTrust(SPair.ChildId) < threshold **then***
5: Declare SPair.ChildId is 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 3: 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 CMNTS are evaluated in NS-3 simulator. We have compared proposed approach with CPDM [4] and CPMTS [5]. 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. 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 and success transmission. It is observed that performance of both CPDM and CPMTS degrades with the injection of misrouting attack in malicious nodes.

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.

*% detection = (No. of malicious nodes detected / No. of malicious nodes in network)*100*

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 3, percentage of detection is improved in CMNTS when compare to CPDM and CPMTS approaches. In CPDM, the percentage of detection deteriorates as the number of malicious nodes increases. The improved performance of the CMNTS is due to the handling of misrouting attack.

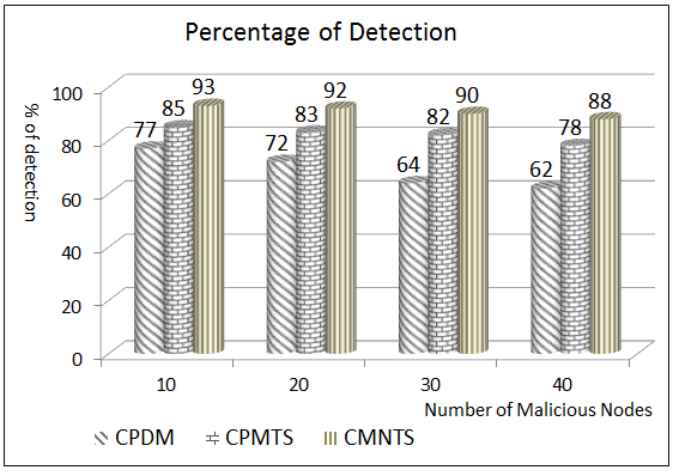

Fig. 3: Percentage of malicious node detection

B. Percentage of False Isolation

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

*% false detection = (No. of genuine nodes isolated / No. of genuine nodes in network)*100*

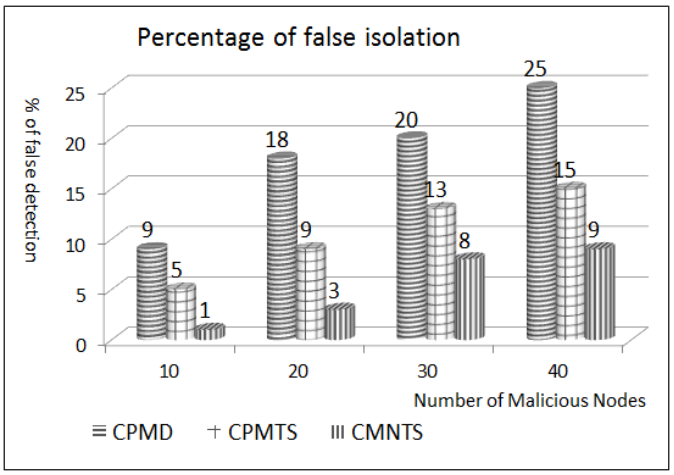

Fig. 4: Percentage of false isolation

As shown in figure 4, percentage of false detection is high in CPDM approach. In CPDM approach, even though intermediate node modifies the packet, Sink considers the source node for malicious detection as the approach is solely based on sequence number of the packet. CPMTS reduces the false isolation compare to CPDM, but false isolation increases on injecting misrouting attack. 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.

C. Early Detection Rate

Simulated and analyzed the early detection when the number of malicious nodes are 20. In all CPDM, CPMTS, and CMNTS 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. CPMTS cannot detect malicious nodes if node performs misrouting attack.

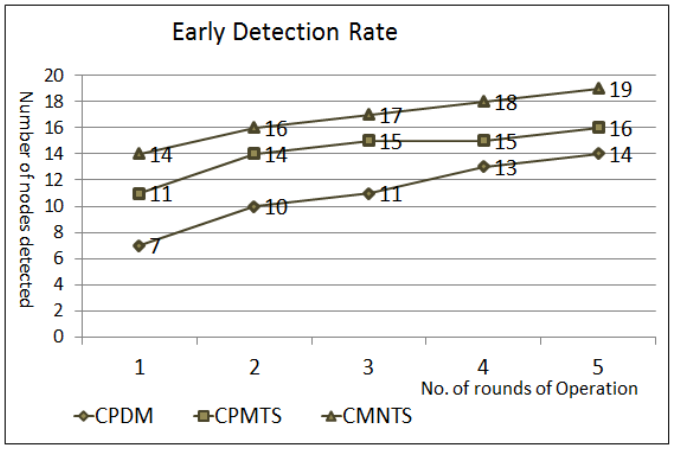

Fig. 5: Early Detection Rate

As shown in figure 5, CMNTS detects the malicious nodes early compare to CPDM and CPMTS so that network cannot afford to loose lot of meaningful information before all malicious nodes are detected.

D. 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 1 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 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*.

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 the parent node information with one hop neighbour nodes. In figure 1, When *Y* announces its selected parent *X* to one hop

TABLE I: Security Attacks Comparison

Attack Type	CPDM	CPMTS	CMNTS
Packet Modification	Yes	Yes	Yes
Packet Dropping	Yes	Yes	Yes
Sybil Attack	No	Yes	Yes
Bad Mouthing Attack	No	Yes	Yes
MisRouting Attack	No	No	Yes

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 CMNTS 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.

E. 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 two bytes of data in the beginning of the packet and removes two bytes in the tail end. Packet size depends on the maximum hop distance from a child to sink node so that even after adding two bytes of data and removing two bytes 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 to detect security problems such as packet modification, dropping, misrouting, and using wrong identity. CMNTS starts with creating a tree topology having parent-child relation information in Sink node. Data transmission happen across multiple rounds of equal time duration. Each node chooses its parent node at the beginning of a round. CMNTS identifies bad nodes after each round and keeps few suspiciously bad nodes till the completion of next round. At the end of each round, CMNTS tries to find the bad nodes from suspiciously bad nodes. But CMNTS 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 CMNTS detects the malicious nodes early with high detection rate and low false detection compare to CPMD and CPMTS. CMNTS can be further improved to avoid transmission power level control attack 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, pages 103–105, Oct 2003.

[2] Issa M. Khalil, Saurabh Bagchi, "Stealthy Attacks in Wireless Ad Hoc Networks: Detection and Countermeasure," In *IEEE Transactions On Mobile Computing*, volume 10, pages 1096–1112, August 2011.

[3] Issa M. Khalil, "ELMO: Energy Aware Local Monitoring in Sensor Networks," In *IEEE Transactions on Dependable and Secure Computing*, volume 8, pages 523–536, August 2011.

[4] Chuang W, Taiming F, Jinsook K, Guiling W, and Wensheng Z, "Catching Packet Droppers and Modifiers in Wireless Sensor Networks," In *IEEE Transactions on Parallel and Distributed Systems*, volume 23, pages 835–843, May 2012.

[5] 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 2015*, 2015.

[6] C. Karlof and D. Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures," In *Proc. IEEE First Intl Workshop Sensor Network Protocols and Applications*, 2003.

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

[8] 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.

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

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

[11] 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*, 2004.

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