2015 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)
19-20 December 2015, BUET, Dhaka, Bangladesh

CPMTS:Catching Packet Modifiers 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. Packet modification is a common attack in wireless sensor networks. In literature, many schemes have been proposed to mitigate such an attack but very few detect the malicious nodes effectively. 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 determine the modifiers 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. With the combination of trust factor and fixed path routing to detect malicious activity, analytical results show that proposed method detect modifiers efficiently and early, and also with low percentage of false detection.**

Keywords*—WSN, trust based, malicious node, packet modification

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, 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. Packet dropping and modification is a basic problem which has 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 wormhole, rushing attacks can be launched without the help of cryptography keys [2].

In this paper, we propose simple yet effective scheme 'Catching Packet Modifiers with Trust Support (CPMTS)' to identify packet modifiers. After deployment, each node selects a list of parent nodes which have equal and shortest distance to sink node. Each node choose a parent node among the selected 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. 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. Marker data added by each node helps sink to trace the nodes in the routing path. Sink finds a pair of nodes which are responsible for packet modification during packet decryption process if packet decryption fails and uses the trust value to filter the malicious node among the pairs. 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) [3] has been proposed 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 provide a theoretical performance analysis showing the comparison between the CPDM and our approach with various parameters. The rest of the paper is organized as follows, section 2 discusses about the related work, section 3 describes the problem statement, section 4 presents the solution and algorithm, section 5 provides the performance analysis and results, and section 6 concludes the work and discusses the future challenges.

II. Related Work

To handle the packet droppers and modifiers, multi-path routing [4], [5], [6], [7] 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 [8], [9], [10] 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

978-1-4673-8786-6/15/$31.00 ©2015 IEEE

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 multi-path routing and neighborhood monitoring is not affordable for sensor networks. In [2], energy efficient sleep-wake approach along with local monitoring method is used to detect malicious nodes. CPDM [3] proposed a scheme to detect packet modifiers and droppers without using multi-path forwarding or monitoring approach, but the method identify the malicious nodes after long time operation of network and also has high false positive detection.

III. Problem Definition

Goal of the proposed CPMTS method is to identify the packet modifiers in the wireless sensor networks by sink node while processing the data packet. 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.

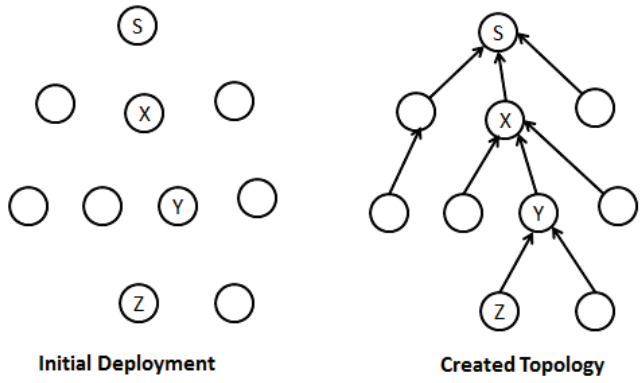

Fig. 1. Deployment and Topology

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.

System Assumptions: CPMTS assumes the network is static and the links are bidirectional. CPMTS assumes that pair wise keys are shared between sink and each network node before deployment. Assumed no malicious activity during topology creation. In CPMTS each node knows the current *(X,Y)* location and also the location of the neighbor nodes. Malicious behavior is manifested through misrouting or modifying packets. Source nodes are assumed to be genuine. Assumed that parent node adjusts the transmission power level such that child node hears the packet transmitted by parent.

IV. CPMTS

The proposed method has several steps of operation. CPMTS 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. 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.

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.

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. 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.

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

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.

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 child. i) Child chooses the next parent in its list with which it never had an interaction. ii) Child chooses the parent for which trust value is high.

V. Performance Analysis

The efficiency and effectiveness of CPMTS are evaluated in NS-3 simulator. We have compared proposed approach with CPDM [3]. 100 nodes are randomly deployed in a square area. 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 and success transmission.

A. Percentage of Detection

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

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

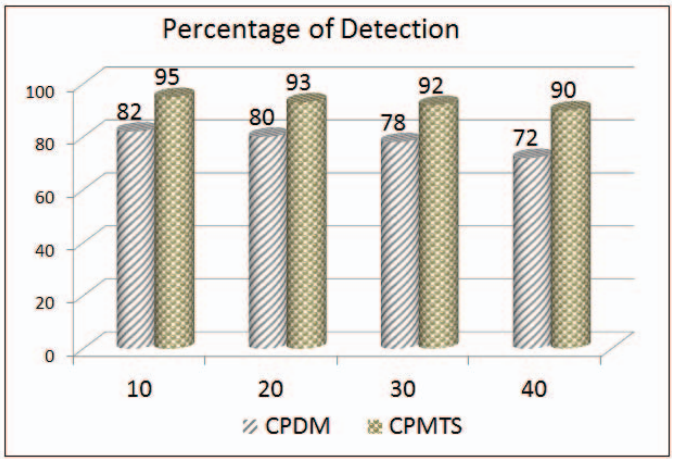

Fig. 2. Percentage of malicious node detection

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 2, percentage of detection is improved in CPMTS when compare to CPDM approach. In CPDM, the percentage of detection deteriorates as the number of malicious nodes increases.

B. Percentage of False Isolation

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

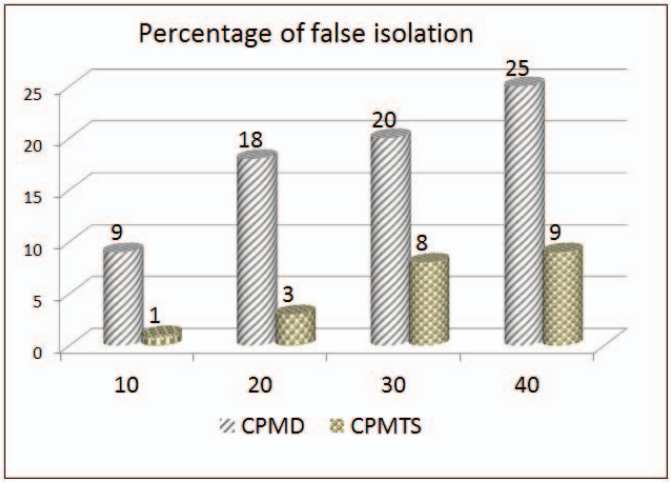

Fig. 3. Percentage of false isolation

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

As shown in figure 3, 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. 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 mouth 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 both CPDM and CPMTS, 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.

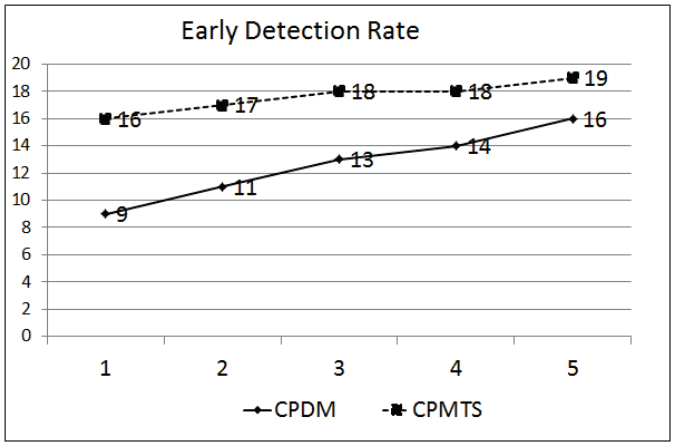

Fig. 4. Early Detection Rate

As shown in figure 4, CPMTS detects the malicious nodes early compare to CPDM, so that network cannot afford to loose lot of meaningful information before all malicious nodes are detected.

VI. Conclusions and Future Directions

Packet modification is a usual security attack to disrupt the data in wireless sensor networks. Proposed method is proven to be efficient to detect packet modifiers compare to CPDM approach. CPMTS 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. CPDM identifies few bad nodes after each round and keeps suspiciously bad nodes till the completion of all rounds. At the end of all rounds, CPDM tries to find the bad nodes from suspiciously bad nodes. But CPMTS adds suspiciously bad nodes during each packet decryption process if packet decryption fails and identifies the most bad nodes after first round. Performance results show that CPMTS detects the malicious nodes early with high detection rate and low false detection. CPMTS can be further improved to avoid misrouting, black hole attack and sybil attacks.

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, "ELMO: Energy Aware Local Monitoring in Sensor Networks," In *IEEE Transactions on dependable and secure computing*, volume 8, pages 523–536, August 2011.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.