A Light-Weight Countermeasure to Forwarding Misbehavior in Wireless Sensor Networks: Design, Analysis, and Evaluation

Cong Pu, *Student Member, IEEE*, and Sunho Lim, *Member, IEEE*

***Abstract*—Due to the lack of centralized coordination, physical protection, and security requirements of inherent network protocols, wireless sensor networks (WSNs) are vulnerable to diverse denial-of-service (DoS) attacks that primarily target service availability by disrupting network routing protocols or interfering with on-going communications. In this paper, we propose a light-weight countermeasure to a selective forwarding attack, called SCAD, where a randomly selected single checkpoint node is deployed to detect the forwarding misbehavior of malicious node. The proposed countermeasure is integrated with timeout and hop-by-hop retransmission techniques to quickly recover unexpected packet losses due to the forwarding misbehavior or bad channel quality. We also present a simple analytical model and its numerical result in terms of false detection rate. We conduct extensive simulation experiments for performance evaluation and comparison with the existing CHEMAS and CAD schemes. The simulation results show that the proposed countermeasure can improve the detection rate and packet delivery ratio (PDR) as well as reduce the energy consumption, false detection rate, and successful drop rate.**

***Index Terms*—Checkpoint-based detection, denial of service (DoS), forwarding misbehavior, selective forwarding attack, wireless sensor networks (WSNs).**

I. Introduction

WIRELESS sensor networks (WSNs) have been receiving a considerable attention as an alternative solution for scalable monitoring and data collection in a hostile or unattended area. A WSN consists of resource-constrained sensor nodes (later nodes) in terms of sensing, computing, or communicating capability. As a part of rapidly emerging Internet of Things (IoT), where a myriad of multiscale nodes and devices are seamlessly blended, WSNs will play an important role in building a ubiquitous computing and communication infrastructure. With the prevalence of cloud, social media, and wearable computing as well as the reduced cost of processing power, storage, and bandwidth, it is envisaged that wirelessly connected smart nodes and devices under IoT will enhance flexible information accessibility and availability as well as change our life further.

Due to the harsh environmental conditions of deployment and the lack of physical protection, however, nodes can be easily captured, tampered, or destroyed by an adversary in WSNs. An open nature of wireless communication can also enable the adversary to overhear, duplicate, corrupt, or alter sensory data. In addition, most conventional network routing protocols are not originally designed to consider the security requirements for malicious attacks. Thus, WSNs are vulnerable to a well-known denial-of-service (DoS) attack that primarily targets service availability by disrupting network routing protocols or interfering with on-going communications.

In this paper, we investigate a selective forwarding attack and propose its countermeasure in multihop WSNs, where single or multiple malicious nodes randomly or strategically drop any incoming packet. The selective forwarding attack primarily targets the network routing vulnerabilities of multihop WSNs by violating an implicit assumption, i.e., all nodes faithfully and collaboratively route packets to a sink. Unlike a blackhole attack [1], where a malicious node blindly drops any incoming packet, it is a nontrivial problem to detect the forwarding misbehavior from temporal node failures or packet collisions. In light of these, we propose a light-weight countermeasure and its corresponding techniques to energy efficiently detect the selective forwarding attack and measure its security resiliency and performance tradeoff through an analytical model and extensive simulation experiments. Our major contribution is briefly summarized in twofold.

1) We propose a single checkpoint-based countermeasure, called SCAD, in WSNs. Unlike prior detection schemes [2]–[6], where multiple checkpoint nodes are deployed, the SCAD deploys a single checkpoint-assisted approach, and its security resiliency and communication performance are measured. The SCAD is also incorporated with timeout and hop-by-hop retransmission techniques to recover unexpected packet losses due to the forwarding misbehavior or bad channel quality.
2) We propose a simple analytical model of the SCAD and show its numerical result in terms of false detection rate. We also revisit prior checkpoint-based and monitor-based detection approaches, CHEMAS [3] and CAD [5], and modify them to work in WSNs for performance comparison.

We develop a customized discrete-event simulation framework using the OMNeT++ [7] and evaluate its performance through extensive simulation experiments in terms of detection rate, successful drop rate, packet delivery ratio (PDR), energy consumption, number of forwarded and overheard packets, and false detection rate. The simulation results indicate that the

Manuscript received September 24, 2015; revised February 16, 2016; accepted February 23, 2016. Date of publication March 17, 2016; date of current version March 23, 2018.

The authors are with the T^{2}WISTOR: TTU Wireless Mobile Networking Laboratory, Department of Computer Science, Texas Tech University, Lubbock, TX 79409 USA (e-mail: cong.pu@ttu.edu; sunho.lim@ttu.edu).

Digital Object Identifier 10.1109/JSYST.2016.2535730

proposed countermeasure is a viable detection approach to a selective forwarding attack.

This paper is organized as follows. The prior approaches are summarized and analyzed in Section II. The proposed countermeasure and its simple analytical model are presented in Section III. Section IV is devoted to extensive simulation experiments and performance comparison and analysis. We further explore the potential extensions of proposed countermeasure in Section V. Finally, we conclude the paper in Section VI.

II. Related Work

Both Watchdog and Pathrater approaches [8] are proposed to detect and mitigate routing misbehavior. The Watchdog technique detects a misbehaving node by overhearing its transmission to see whether it correctly forwards a packet. Each node assigns a misbehavior rate to its neighbor nodes and monitors other nodes by updating a table of misbehavior rate in the network. When a source node selects a path for routing, it avoids to include a misbehaving node in the path based on the table. The Watchdog and Pathrater are extended by deploying implicit acknowledgment [9] and overhearing [10] techniques, in which each node monitors the forwarding operations of its neighbor nodes and detects a forwarding misbehavior. In order to seamlessly monitor the forwarding operations, nodes are required to stay in active but it is not feasible especially in a battery-powered network because of energy consumption.

Several acknowledgment-based countermeasures to selective forwarding attacks have been proposed in [2] and its variants [3], [4]. The basic idea is that a set of intermediate nodes located along the forwarding path to a sink acts as a checkpoint node and monitors the forwarding misbehavior by replying an acknowledgment (Ack) packet to a source node. If an intermediate node does not receive the required number of Ack packets, it suspects the next node located in the path as a malicious node and generates an *Alarm* packet, which is forwarded back to the source node. In [4], Ack packets are forwarded to the sink instead. Since multiple checkpoint nodes generate *Ack* packets, intermediate nodes may excessively receive and forward Ack packets and consume non-negligible battery energy in resource-constrained WSNs.

In the CAD [5], forwarding misbehaviors are filtered from packet losses due to a bad channel quality or packet collisions based on the observation of channel and network traffic in wireless mesh networks (WMNs). Each node monitors the network traffic of its neighbor nodes and estimates a packet loss rate. A neighbor node is suspected as a malicious node if it shows higher loss rate compared to a detection threshold incorporated with the estimated loss rate. Since the channel quality tends to temporarily fluctuate, it becomes an issue to adaptively set the detection threshold based on time-varying estimated loss rates. The FADE [6] is a variant of the CAD, where each node overhears a link-layer acknowledgment and waits for a two-hop acknowledgment from its downstream nodes after it forwards a packet. This approach still suffers from the number of received and forwarded Ack packets and its energy consumption.

In [11], a DSR-based cooperative bait detection scheme (CBDS) is proposed to detect both selective forwarding and blackhole attacks in mobile *ad hoc* networks (MANETs). The basic idea is that a source node selects an adjacent node and uses its address as a bait destination address to entice a malicious node to send back a forged or fake route reply (RREP) packet. The malicious node can be detected using a reverse tracing technique.

The HCD [12] is proposed to detect the forwarding misbehaviors of malicious nodes in energy harvesting WSNs. To the best of our knowledge, [12] is the first approach to explore a countermeasure to selective forwarding attack in the realm of energy harvesting WSNs. In the HCD, each node records the trace of forwarding operations through overhearing and exchanges this trace information with its adjacent nodes to detect any forwarding misbehavior. Then each node can reduce the forwarding probability of malicious node to exclude the malicious node from the network. However, the HCD shows high detection latency because of a long window size of trace information exchanged. In [13], a camouflage-based detection scheme, called CAM, is proposed to detect the forwarding misbehavior in energy harvesting motivated networks (EHNets). The basic idea is that each node hides its current operational status and pretends not to overhear or monitor any on-going forwarding operation of its adjacent nodes to detect a deep lurking malicious node.

In summary, most prior approaches primarily focus on how to increase the detection rate of malicious nodes in the network, where multiple checkpoint nodes are deployed. However, little attention has been paid for a light-weight countermeasure by considering security resiliency and performance tradeoff.

III. Proposed Countermeasure

In this section, we first present both system and adversary models and then propose a light-weight countermeasure to a selective forwarding attack in WSNs, called SCAD. A simple analysis of the SCAD and its numerical result in terms of the false detection rate are also presented.

A. System and Adversary Models

When a node detects an event, it becomes a source node, generates a data packet, and forwards the packet toward a sink in WSNs. To deliver the data packet toward the sink, a simple broadcast-based forwarding [14], directed diffusion [15], or geographic-based routing [16] techniques can be deployed. Each node is aware of its one-hop neighbor nodes by exchanging a one-time single-hop *Hello* packet piggybacked with its node *id* [14]. We assume that the network is dense enough to find multiple forwarding candidate nodes. Thus, a single node connecting two subnetworks is not considered because it could be a single point of failure or a malicious node.

A primary goal of the adversary is to attack service availability and degrade the network performance by interfering with on-going communications. An adversary is able to capture and compromise a legitimate node to behave maliciously. A malicious node located along the forwarding path may selectively drop or forward any incoming packet to deafen a sink. The malicious node may also eavesdrop on an on-flying packet and

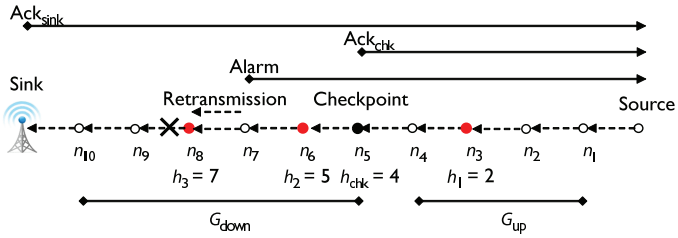


Fig. 1. Snapshot of network, where a forwarding path from a source node to a sink is depicted with a single checkpoint node. Here, both black and red dots represent a checkpoint node and malicious nodes, respectively.

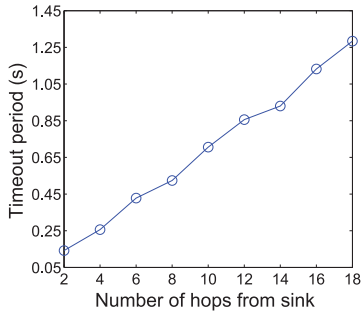


Fig. 2. Timeout period increases as the number of hops from the sink increases.

inject false information or modify its packet header to mislead network traffic. However, if a sender can authenticate a packet with a light-weight digital signature [17], a receiver can easily verify the packet and detect any modification. In this paper, we primarily focus on the selective forwarding attacks or the adversarial scenarios [2]–[6] that cannot be detected by digital signatures and cryptographic primitives.

B. Single Checkpoint-Based Detection

The SCAD deploys a single checkpoint-assisted approach and consists of three major operations: single checkpoint node selection, timeout, and retransmission. First, when a source node generates a data packet, it randomly selects one of the intermediate nodes located along the forwarding path to a sink as a checkpoint node and piggybacks a random number into the packet. Since the source node independently and randomly selects a checkpoint node per-packet basis, it is not trivial for an adversary to predict the checkpoint node for the next data packet. Here, we do not consider dynamically changing routing paths for the same packet during the transmission, because it can exclude the checkpoint node selected by the source node in the path. When a node receives the data packet, it caches the packet in its local storage and checks whether it is selected as a checkpoint node by comparing its one-way hash and map functions [3]. If both functions are equal to one (e.g., selected as a checkpoint node), the node forwards the data packet to the next node and replies an Ack) packet back to the source node. In Fig. 1, a randomly selected checkpoint node (e.g., n_5) divides the forwarding path into two streams: upstream (G_{up}: e.g., n_1 to n_4) and downstream (G_{down}: e.g., n_5 to n_{10}). Since both the sink and checkpoint node reply an Ack packet, any intermediate node located between the source node and the sink receives one or two Ack packets depending on the location of checkpoint node. Note that a malicious node could be selected as a checkpoint node, and thus, it could drop a data packet but reply a fake Ack packet.

Second, when a node forwards a data packet, it sets a timer for an Ack packet originated either from the sink or a checkpoint node, or an *Alarm* packet generated from a downstream node. If the node does not receive the Ack or *Alarm* packet before its timer expires, because of possible forwarding misbehavior or bad channel quality, it generates an *Alarm* packet to prosecute the next node for the forwarding misbehavior and forwards the *Alarm* packet back to the source node. The more malicious nodes drop Ack or even *Alarm* packets, the more forwarding misbehaviors can be detected because upstream nodes experience more timeouts. The malicious node may fabricate an Ack packet but it can be easily detected. This is because each intermediate node can check whether the Ack packet was replied from an illegal node by checking its buffered checkpoint seed [3], which is originally generated from the source node and piggybacked to the data packet. A similar light-weight digital signature [17] can also be used to check whether the packet has been modified.

In the SCAD, we propose a timeout technique to reduce unnecessary packet delivery latency, which can be caused by unexpected packet loss due to the forwarding misbehavior or bad channel quality. We define a timeout period as a tuple, [T^C, T^S], where T^C and T^S are timeout periods of an *Ack* packet originated from a checkpoint node (C) and the sink (S), respectively. If a node is located in G_{down}, its T^C is zero. In order to estimate the timeout period, we consider a single-hop based estimated trip time (T_{ETT}) that can be measured from when a node forwards a data packet ($T_{F,\text{data}}$) to when it receives an *Ack* packet either from the checkpoint node or the sink ($T_{R,Ack}$). Then T_{ETT} is divided by H_k, which is the number of hops counted from the node to the checkpoint node or the sink when a node forwards a data packet with sequence number k. T_{ETT} is updated by the low-pass filter with a filter gain constant α

$$T^{\ell}_{\text{ETT}} = \alpha \cdot T^{\ell}_{\text{ETT}} + (1-\alpha) \cdot T_{\text{ETT},k-1} \tag{1}$$

where $\ell \in \{\text{C}, \text{S}\}$. $T_{\text{ETT},k-1}$ is the measurement from the most recently received Ack packet, and it is expressed as

$$T_{\text{ETT},k-1} = \frac{T_{R,Ack} - T_{F,\text{data}}}{H_{k-1}}. \tag{2}$$

Thus, the timeout period is expressed as

$$T^{\ell} = T^{\ell}_{\text{ETT}} \cdot H_k + H_k \cdot \delta \tag{3}$$

where δ is an adjustment factor and $H_k \cdot \delta$ is added to consider the packet delivery latency. Fig. 2 shows the changes of timeout period against the number of hops from the sink.

Third, we deploy a hop-by-hop retransmission approach to reduce the packet delivery latency and expedite in detecting the forwarding misbehavior in the SCAD. If a node does not receive an Ack or *Alarm* packet before its timer expires, it retransmits a cached data packet to the next node after forwarding an *Alarm* packet to the source node. If the node still does not receive an

Ack or *Alarm* packet again, it forwards another *Alarm* packet again, quits the retransmission, and discards the cached data packet. For example, suppose n_8 drops a data packet forwarded from n_7 in Fig. 1. Then n_7 generates an *Alarm* packet and retransmits its cached data packet to n_8. If n_8 drops the retransmitted data packet again, n_7 generates another *Alarm* packet. The more malicious nodes drop retransmitted data packets, the sooner the source node detects their forwarding misbehaviors. Note that the source node may isolate a suspected node from the network after receiving a number of *Alarm* packets by broadcasting a packet piggybacked with the *id* of suspected node, or reducing a forwarding probability of the suspected node [12]. However, this is out of the scope of this paper. Major operations of the proposed countermeasure are summarized in Fig. 3.

C. Analysis of the Proposed Countermeasure

In this paper, we analyze the SCAD in terms of average false detection rate. When a packet (e.g., data, *Ack*, or *Alarm*) is lost because of the bad channel quality, however, a node may mistakenly prosecute the next located legitimate node as a malicious node, resulting in the false detection. In Fig. 1, e.g., n_6 drops a data packet forwarded from n_5. Then, n_5 generates an *Alarm* packet to prosecute the forwarding misbehavior of n_6 when its timer expires, and forwards the *Alarm* packet back to the source node. Due to the bad channel quality, the *Alarm* packet can be lost again during the transmission from n_5 to n_4. Then n_4 generates another *Alarm* packet to prosecute the forwarding misbehavior of n_5 when its timer expires, resulting in a false detection. In this analysis, we assume that the bad channel quality in terms of channel error primarily causes packet loss without considering packet drop conducted by malicious nodes to clearly see the impact on the false detection.

Suppose total N nodes excluding the sink and source node are located in the forwarding path, where m ($\geq$1) of them are malicious nodes. φ is a channel error rate, either 10% or 20%. Let P_F be an average false detection rate, which is the sum of average false detection rates of data (P_{FD}), *Ack* (P_{FA}), and *Alarm* (P_{FM}) packet losses. Then P_F is expressed as

$$P_F = P_{\text{FD}} + P_{\text{FA}} + P_{\text{FM}}. \tag{4}$$

First, P_{FD} is expressed as

$$P_{\text{FD}} = \frac{1}{n-m-1}(P_{\text{FD1}} + P_{\text{FD2}}) \tag{5}$$

where

$$P_{\text{FD1}} = \sum_{i=1}^{m} \sum_{j=0}^{h_i - h_{i-1} - 2} (1-\varphi)^{2j+2h_{i-1}}\varphi \tag{6}$$

$$P_{\text{FD2}} = \sum_{j=0}^{n-h_m-2} (1-\varphi)^{2j+2h_m}\varphi. \tag{7}$$

Here, h_i ($0 \leq i \leq m$, and $h_0 = 0$) is the number of hops from the ith malicious node to the first node (e.g., n_1). P_{FD} is the average false detection rate of data packet loss between the first and the last nodes (e.g., n_1 to n_{10} in Fig. 1). In (6), P_{FD1} is the total false detection rates between the first node and the last malicious node (e.g., n_1 to n_8). Note that a data packet loss can lead to both false and correct detection cases. In a false detection case based on Fig. 1, if a data packet is lost during the transmission from n_3 to n_4, a malicious node n_3 generates an *Alarm* packet to prosecute the forwarding misbehavior of a normal node n_4. If this *Alarm* packet is forwarded to the source node, a false detection can occur. In case of a correct detection, however, suppose a data packet is lost during the transmission from n_2 to n_3. Then a legitimate node n_2 generates an *Alarm* packet to prosecute a malicious node n_3, which can lead to a correct detection. In (7), P_{FD2} is the total false detection rates

Notations:
- $[T^C, T^S]$: Defined before.
- $pkt[type, seq, chk, x]$: A packet with a sequence number, seq, checkpoint node id, chk, malicious node id, x and packet type, $type$. Here, $type$ is data, *Ack* or *Alarm*.
- $Q_i[pkt[seq]]$, $flag_{seq}$, c^i_{mis}, τ: A queue of received data packets in n_i, a data packet $pkt[seq]$ retransmission flag, the number of detected forwarding misbehaviors of n_i and a forwarding misbehavior threshold.

◇ When a source node, n_s, senses an event:
  Send out $pkt[data, seq, chk, none]$;
◇ When the sink, n_{sink}, receives an event packet:
  Reply $pkt[Ack, seq, sink, none]$;
◇ When a node, n_i, detects a forwarding misbehavior of a malicious node, n_m ($m = i+1$): T^C or T^S expires;
if $flag_{seq}$ is **false** /* Has not retransmitted $pkt[seq]$ */
  Reply $pkt[Alarm, seq, none, m]$;
  Retransmit $pkt[data, seq, chk, none]$;
  $flag_{seq}$ = **true**;
else
  Reply $pkt[Alarm, seq, none, m]$;
◇ When a node, n_i, receives a $pkt[type, seq, chk, x]$,
if $pkt[type]$ == *data*
  if $i == chk$
    Enqueue the $pkt[seq]$ into Q_i;
    Set up [none, T^S]; /* Eq. 1 */
    Forward $pkt[data, seq, chk, none]$;
    Reply $pkt[Ack, seq, chk, none]$;
  else
    if $i < chk$ /* n_i is in the upstream of chk */
      Enqueue the $pkt[seq]$ into Q_i;
      Set up [T^C, T^S];
      Forward $pkt[data, seq, chk, none]$;
    else /* n_i is in the downstream of chk */
      Enqueue the $pkt[seq]$ into Q_i;
      Set up [none, T^S];
      Forward $pkt[data, seq, chk, none]$;
end if
if $pkt[type]$ == *Ack*
  if $sink \in pkt$ /* Ack transmitted from the sink */
    Dequeue the $pkt[seq]$ from Q_i;
    Forward $pkt[Ack, seq, sink, none]$;
    Cancel T^S;
  else /* Ack is from a checkpoint */
    Forward $pkt[Ack, seq, chk, none]$;
    Cancel T^C;
end if
if $pkt[type]$ == *Alarm* /* x is a malicious node */
  if n_i is the source node
    $c^x_{mis} = c^x_{mis}$ + 1;
    if $c^x_{mis} \geq \tau$
      Broadcast $isolation$ packet;
  else
    Dequeue the $pkt[seq]$ from Q_i;
    Cancel T^S or T^C and T^S;
    Forward $pkt[Alarm, seq, none, x]$;
end if

Fig. 3. Pseudocode of proposed countermeasure.

between the last malicious node and the last node on the forwarding path (e.g., n_8 to n_{10}). Unlike to P_{FD1}, only a false detection can occur because there is no malicious node between n_8 to n_{10}.

Second, P_{FA} is expressed as

$$P_{\text{FA}} = P_{\text{FA1}} + P_{\text{FA2}} \tag{8}$$

$$P_{\text{FA1}} = \frac{RD_{\text{chk}}}{h_{\text{chk}} - k}(P_{\text{FA1, 1}} + P_{\text{FA1, 2}}) \tag{9}$$

where

$$RD_{\text{chk}} = (1-\varphi)^{h_{\text{chk}}} \tag{10}$$

$$P_{\text{FA1, 1}} = \sum_{j=0}^{h_{\text{chk}}-h_k-1} (1-\varphi)^{h_{\text{chk}}-1}\varphi \tag{11}$$

$$P_{\text{FA1, 2}} = \sum_{i=k}^{1} \sum_{j=0}^{h_i-h_{i-1}-2} (1-\varphi)^{h_{\text{chk}}-1}\varphi. \tag{12}$$

Also

$$P_{\text{FA2}} = \frac{RD_{\text{sink}}}{n-m-1}(P_{\text{FA2, 1}} + P_{\text{FA2, 2}}) \tag{13}$$

where

$$RD_{\text{sink}} = (1-\varphi)^{n-1} \tag{14}$$

$$P_{\text{FA2, 1}} = \sum_{j=0}^{n-h_m-2} (1-\varphi)^{n-2}\varphi \tag{15}$$

$$P_{\text{FA2, 2}} = \sum_{i=m}^{1} \sum_{j=0}^{h_i-h_{i-1}-2} (1-\varphi)^{n-2}\varphi. \tag{16}$$

Here, h_{chk} is the number of hops from the checkpoint node to the first node (e.g., n_5 to n_1, $h_{\text{chk}} = 4$). k is the number of malicious nodes located in G_{up}. $P_{\text{FA1, 2}}$ becomes zero when $k = 0$. In (8), P_{FA} is an average false detection rate of the first and second *Ack* packet losses from the checkpoint node or the sink to the first node (e.g., n_5 to n_1, or sink to n_1), respectively. RD_{chk} and RD_{sink} are the probabilities that a data packet reaches to the checkpoint node and the sink in (10) and (14), respectively. In (9), P_{FA1} is an average false detection rate of the first *Ack* packet loss during the transmission between the checkpoint node and the first node (e.g., n_5 to n_1).

In (11), $P_{\text{FA1,1}}$ is the total false detection rates between checkpoint node and the first malicious node (e.g., n_5 to n_3). Similar to data packet loss, an *Ack* packet loss can lead to both false and correct detections. For example, an *Ack* packet loss during the transmission from n_4 to n_3 can lead to a false detection because a malicious node n_3 generates an *Alarm* packet to prosecute the forwarding misbehavior of a normal node n_4. If an *Ack* packet is lost during the transmission from n_3 to n_2, a correct detection can occur because a normal node n_2 generates an *Alarm* packet to prosecute the malicious node n_3. In (12), $P_{\text{FA1,2}}$ is the total false detection rates between the first malicious node and the first node on the forwarding path (e.g., n_3 to n_1). Since no malicious node exists between n_3 and n_1, only a false detection can occur.

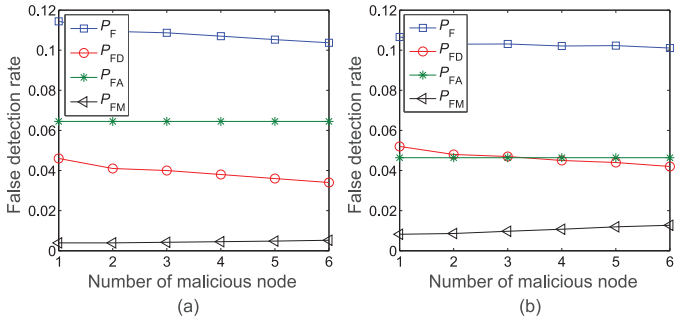


Fig. 4. False detection rate against the number of malicious nodes and channel error rates. (a) 10% Channel error rate. (b) 20% Channel error rate.

In (13), P_{FA2} is an average false detection rate of the second *Ack* packet loss during the transmission between the sink and the first node (e.g., sink to n_1). Similar to the first *Ack* packet loss, both false and correct detections of *Ack* packet loss can occur during the transmission between the sink and the first malicious node. Thus, only a false detection can occur during the transmission between the first malicious node and the first node. In (15) and (16), $P_{\text{FA2,1}}$ is the total false detection rates between the sink and the first malicious node (e.g., sink to n_3), while $P_{\text{FA2,2}}$ is the total false detection rates between the first malicious node and the first node in the forwarding path (e.g., n_3 to n_1).

Third, P_{FM} is expressed as

$$P_{\text{FM}} = \frac{1}{n-m-1}(P_{\text{FM1}} - P_{\text{FM2}}) \tag{17}$$

where

$$P_{\text{FM1}} = \sum_{i=1}^{n-2}(1-\varphi)^{2i-1}\varphi^2 \tag{18}$$

$$P_{\text{FM2}} = \sum_{i=1}^{m}(1-\varphi)^{2h_i-1}\varphi^2. \tag{19}$$

P_{FM} is an average false detection rate of *Alarm* packet loss between the first and the last nodes. In (18), P_{FM1} includes the probabilities of both false and correct detections for *Alarm* packet loss, respectively. In case of a false detection based on Fig. 1, suppose n_6 intentionally drops a data packet and n_5 generates an *Alarm* packet to prosecute the forwarding misbehavior of n_6. If the *Alarm* packet is lost during the transmission from n_5 to n_4, n_4 generates another *Alarm* packet to prosecute the forwarding misbehavior of n_5. If this *Alarm* packet is forwarded to the source node, then a false detection can occur. In case of a correct detection, denoted as P_{FM2} in (19), suppose a data packet is lost during the transmission from n_3 to n_4, n_3 generates an *Alarm* packet to prosecute the forwarding misbehavior of n_4, and this *Alarm* packet is lost during the transmission from n_3 to n_2. Then n_2 generates another *Alarm* packet to prosecute the forwarding misbehavior of n_3, leading to a correct detection.

In Fig. 4, we show a numerical result of the impact of number of malicious nodes (m) and channel error rate (φ) on the average false detection rate based on the aforementioned analyses. Here, 20 intermediate nodes are located in the forwarding

TABLE I
SIMULATION PARAMETERS

Parameter	Value
Network area	300×300 m^2
Number of nodes	250
Number of malicious nodes	1–6
Channel error rate	0–10%
Radio data rate	250 Kbps
Packet injection rate	0.5 packet/s
Packet size	1 KByte
Packet drop rate	10% or 20%
Radio range	12.3 m
Radio model	CC2420
Simulation time	1000 s

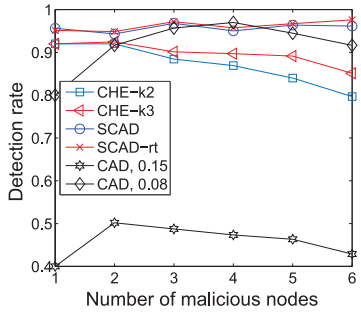


Fig. 5. Detection rate against the number of malicious nodes (10% packet drop rate).

path, where one to six malicious nodes are randomly located. As the m increases, overall P_F decreases with different φ in Fig. 4(a) and (b). In particular, higher φ leads to higher P_{FD} in Fig. 4(b). The more data packets are lost, the harder nodes detect whether the packets are lost or dropped. As the m increases, P_{FD} decreases because data packet has higher probability of being dropped by malicious nodes than that of being lost during the transmission. In P_{FA}, malicious nodes are reluctant to drop any *Ack* packet because this forwarding misbehavior may enforce nodes to generate a series of *Alarm* packets. In Fig. 4(b), lower P_{FA} is observed with $\varphi = 20\%$ compared to 10% channel error rate in Fig. 4(a). This is because more data packets are lost during the transmission and thus, the number of *Ack* packets reduces and the m does not affect P_{FA} much. Both m and φ affect P_{FM}. Higher φ leads to higher P_{FM} in Fig. 4(b). As the m increases, P_{FM} slightly increases because $\frac{1}{n-m-1}$ increases in P_{FM}.

IV. PERFORMANCE EVALUATION

A. Simulation Testbed

We conduct extensive simulation experiments using the OMNeT++ [7] for performance evaluation and analysis. A 300×300 (m^2) rectangular network area is considered, where 250 nodes are uniformly distributed. The communication range of each node is 12.3 (m). The radio model simulates CC2420 with a normal data rate of 250 Kbps [18]. The channel error rate is randomly changed from 0% to 10% with a step size 2% during the simulation. A packet injection rate is 0.5 packet/s and each packet size is 1 KByte. One to six malicious nodes are randomly located along the forwarding path between a source node and the sink. A set of malicious nodes selectively drops any incoming packet with a packet drop rate, either 10% or 20%. The simulation parameters are summarized in Table I.

In this paper, we measure the performance in terms of detection rate, successful drop rate, PDR, energy consumption, number of forwarded and overheard packets, and false detection rate by changing key simulation parameters, including number of malicious nodes, packet drop rate, and channel error rate. For performance comparison, we denote the proposed countermeasure without or with retransmission as SCAD or SCAD-rt, respectively. They are compared with the CHEMAS [3] that is configured with two or three segments (k), denoted as CHE-k2 or CHE-k3, respectively, where an *Ack* packet traverses k segments before being dropped by a checkpoint node. The proposed countermeasure is also compared with the CAD [5], where the detection threshold values are set between 0.08 and 0.15.

B. Performance Comparison

In Fig. 5, as the number of malicious nodes (m) increases, the detection rate decreases in both CHE-k2 and CHE-k3. The probability of multiple malicious nodes being selected as a checkpoint node increases, and they may not report the forwarding misbehavior witnessed from adjacent nodes to the source. The lower detection rate is observed with the smaller k. Since *Ack* packet traverses the less number of hops along the forwarding path, each intermediate node receives less number of *Ack* packets forwarded from the downstream. The CAD is sensitive to the detection threshold value and shows about 95% and 50% detection rates in low (0.08) and high (0.15) threshold values, respectively. Due to the temporarily fluctuating channel quality, it becomes an issue to adaptively set the detection threshold value based on the time-varying estimated loss rates. Thus, the detection rate highly depends on the detection threshold value. Unlike to the CAD, both SCAD and SCAD-rt show high and stable detection rates for entire m. Since a single checkpoint node is selected and replies an *Ack* packet, more intermediate nodes are supposed to receive and forward the *Ack* packet to the source. If an upstream legitimate node does not receive an *Ack* packet before its timeout period, it generates an *Alarm* packet to prosecute the next node for forwarding misbehavior.

In Fig. 6, both successful drop rate and PDR are measured by varying the m and packet drop rate. In Fig. 6(a), the m significantly affects the successful drop rate in both CHE-k2 and CHE-k3. The CHE-k2 shows higher successful drop rate than that of the CHE-k3. This is because an *Ack* packet travels less number of hops and each intermediate node receives less number of *Ack* packets compared to that of the CHE-k3. Multiple malicious checkpoint nodes can cooperate each other and drop data packets without being detected. Depending on the k, the CHEMAS has a performance tradeoff between security resilience and communication overhead. Note that the SCAD, SCAD-rt, and CAD show zero successful drop rate. In Fig. 6(b), under 10% packet drop rate, PDR quickly decreases as the m increases because more data packets are randomly dropped by malicious nodes. The SCAD, SCAD-rt, and CAD show higher PDR than that of the CHE-k2 and CHE-k3 for entire m because the collusion of multiple malicious nodes selected

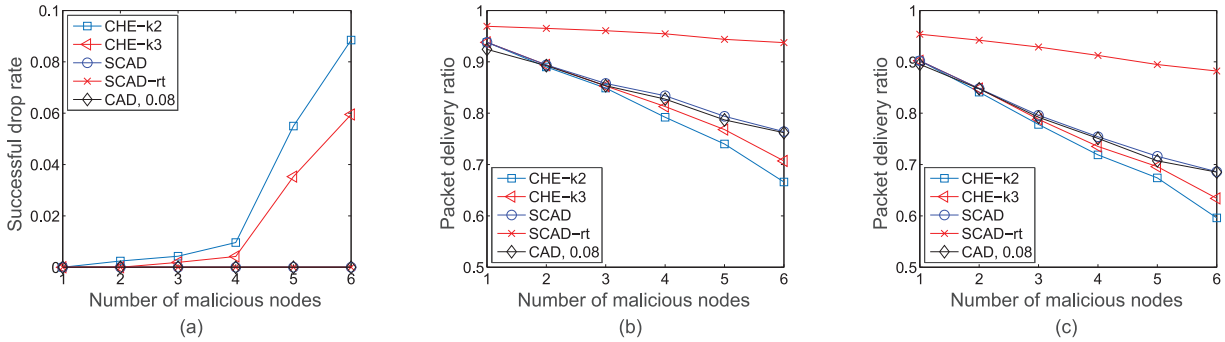


Fig. 6. Successful drop rate and PDRs against the number of malicious nodes. (a) 10% packet drop rate. (b) 10% packet drop rate. (c) 20% packet drop rate.

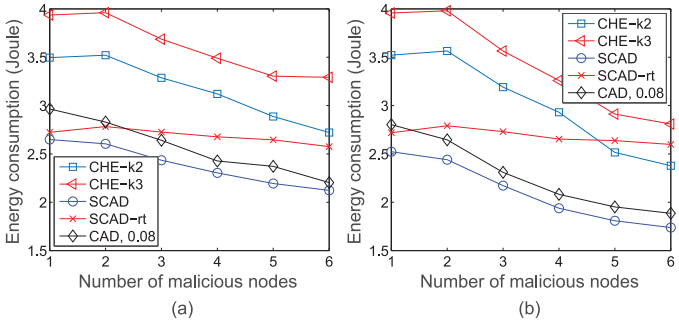


Fig. 7. Energy consumption against the number of malicious nodes and packet drop rate. (a) 10% packet drop rate. (b) 20% packet drop rate.

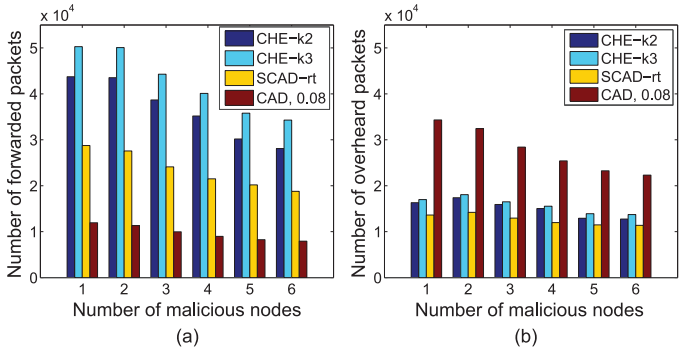


Fig. 8. Number of forwarded and overheard packets against the number of malicious nodes (10% packet drop rate).

as a checkpoint node does not affect to the SCAD, SCAD-rt, and CAD. The SCAD-rt shows the best performance (about 90% or more) because each intermediate node can quickly retransmit its cached data packet to the next node if the data packet is dropped or lost. In Fig. 6(c), overall PDRs decrease with a larger packet drop rate 20%. However, the SCAD-rt still shows the best performance and the PDR decreases gracefully compared to that of the CHE-k2, CHE-k3, and CAD.

In Fig. 7, the energy consumption is measured based on the number of forwarded and overheard packets [19] by varying the m and packet drop rates. In Fig. 7(a), both SCAD and SCAD-rt show lower energy consumption than that of the CHE-k2 and CHE-k3 because of less number of *Ack* packets traversed along the forwarding path. Since an *Ack* packet traverses three and two segments before being dropped by a checkpoint node in the CHE-k3 and CHE-k2, respectively, the CHE-k3 consumes more energy than that of the CHE-k2. The SCAD-rt also consumes more energy than that of the SCAD to retransmit lost or dropped data packets. In Fig. 7(b), overall energy consumptions decrease with higher packet drop rate (20%) because more data packets are dropped by malicious nodes. Note that we measure the number of forwarded and overheard packets in Fig. 8(a) and (b), respectively. The CHE-k2, CHE-k3, and SCAD explicitly send *Ack* packets for detecting forwarding misbehaviors, but the CAD implicitly monitors the network traffic. Thus, intermediate nodes in the CHE-k2, CHE-k3, and SCAD forward more packets but ultimately the CAD overhears more packets, because each node always needs to wake up and observe any on-going packet.

In Fig. 9, we measure the false detection rates by varying the m and channel error rates (e). In Fig. 9(a), both SCAD

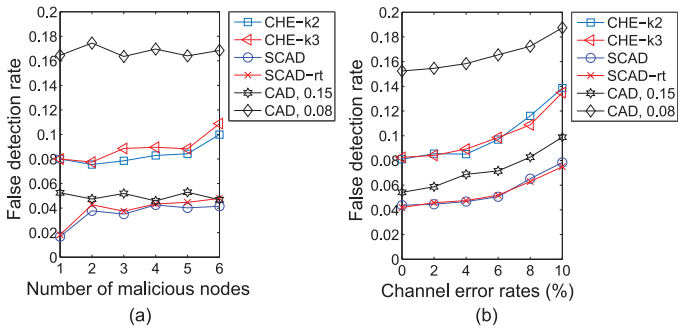


Fig. 9. False detection rate against the number of malicious nodes and channel error rate (10% packet drop rate).

and SCAD-rt show the lowest false detection rate because the number of *Ack* packets generated by a single checkpoint node reduces. Note that this false detection rate is lower than that of the aforementioned analysis (see Fig. 4). Since the analysis extensively counts all packet losses due to the bad channel quality, it shows an upper bound of false detection rate. Multiple checkpoint nodes generate *Ack* packets and each intermediate node frequently forwards them to the source in the CHE-k2 and CHE-k3. Thus, more *Ack* packets can be lost due to the bad channel quality, resulting in higher false detection rate. The CAD with higher detection threshold value (i.e., 0.15) shows the highest false detection rate, because more intermediate nodes mistakenly consider a packet loss as a forwarding misbehavior. In Fig. 9(b), as the e increases, overall false detection rates increase because it becomes harder to detect the forwarding misbehavior of malicious nodes from packet loss due to the bad channel quality.

TABLE II
COMPARISON† OF DETECTION STRATEGIES OF FORWARDING MISBEHAVIOR

Approach	Collusive attack	Computation overhead	Communication overhead	Detection latency	Punishment	Architecture
CHEMAS [3]	N	Medium	High	Low	N	Centralized
CAD [5]	N	Medium	Medium	Medium	N	Centralized
FADE [6]	Y	Medium	High	Low	N	Centralized
Watchdog [8]	N	Low	N	N	N	Stand-alone
CBDS [11]	Y	Medium	Medium	High	N	Distributed
HCD [12]	N	Medium	Low	High	Y	Distributed
CAM [13]	N	Low	N	N	Y	Stand-alone
SCAD	Y	Medium	Medium	Low	N	Centralized

†In this paper, we compare the proposed countermeasure with prior detection strategies of forwarding misbehavior in terms of six aspects: 1) *Collusive attack*: Against to two or more cooperative malicious nodes; 2) *Computation overhead*: Extra computation required for detection; 3) *Communication overhead*: Extra packets generated for detection; 4) *Detection latency*: Time delay to identify forwarding misbehaviors; 5) *Punishment*: Penalty of forwarding misbehaviors; and 6) *Architecture* [20]: *Centralized* is that the major operation of approach is running on the key node and the rest of nodes simply monitor and report the forwarding misbehavior to the key node. Here, *Distributed* implies that the same approach is running on each node and information is exchanged between nodes for detection. On the other hand, *Stand-alone* implies that the same approach is running on each node but no information is exchanged for detection.

V. DISCUSSION

In this section, we first investigate the SCAD of its applicability to other attacks and then further explore its design issues and extensions for future research.

A. Immunity to Other Attacks

We investigate the SCAD whether it can be applied to two well-known attacks: colluding collision attack and power control attack [21].

1) Colluding Collision Attack: A multiple number of malicious nodes may collude together and create a collision at the next hop on purpose by simultaneously sending packets. The IEEE 802.11 medium access control (MAC) protocol with request-to-send (RTS)/clear-to-send (CTS) exchange can be deployed to reduce packet collisions. However, the 802.11 MAC with RTS/CTS exchange is often disabled in many WSN applications because of its non-negligible energy consumption [21]. Thus, it is not trivial to avoid colluding collision attack, but this attack can be detected by the SCAD. In Fig. 1, suppose n_6 sends a data packet to n_7 and its colluding n_8 also simultaneously send any packet to n_7. Then n_7 fails to receive the data packet due to the collision. In the SCAD, since the data packet is lost, the sink will not reply an *Ack* packet back to the source node. Thus, n_5 cannot receive the *Ack* packet from the sink before its timer expires, and it will generate an *Alarm* packet to prosecute the forwarding misbehavior of n_6 and forward the *Alarm* packet back to the source node.

2) Power Control Attack: A malicious node may control its transmission power and forward a packet to exclude a legitimate node from its communication range. This power control attack is similar to selective forwarding attack, and it can be detected by the SCAD. In Fig. 1, suppose n_2 forwards a data packet to n_3 and the data packet is relayed to n_4. Then n_2 sets two timers for the *Ack* packets originated from the sink and n_5, respectively. If n_3 reduces its transmission power and forwards the data packet, n_4 fails to receive the data packet. In the SCAD, since n_5 cannot receive the data packet, it will not reply the *Ack* packet back to the source node. Thus, n_2 cannot receive the Ack packet from the checkpoint node before its timer expires, and it will generate an *Alarm* packet to prosecute the forwarding misbehavior of n_3 and forward the *Alarm* packet back to the source node.

B. Potential Enhancements

We explore design issues and extensions to see the full potential of our approach for efficiently mitigating the forwarding misbehavior.

1) Alternative Path for Retransmission: In the SCAD, if a node does not receive an *Ack* or *Alarm* packet before its timer expires, due to the forwarding misbehavior or bad channel quality, it generates an *Alarm* packet to prosecute the next node for its forwarding misbehavior. Then the node retransmits its cached data packet to the same next node based on the proposed hop-by-hop retransmission. If the next node drops the retransmitted data packet again, the source node will choose an alternative forwarding path without including this suspected node. Thus, we plan to deploy a bypass technique [22], [23] in the hop-by-hop retransmission by selecting an alternative forwarding path from the node that prosecutes the next node and generates an *Alarm* packet. This approach can avoid transmitting the cached data packet to the same suspected node over and over until the source node changes the path. For example, when a node detects the forwarding misbehavior of the next node, it selects another one-hop node as a forwarding node and transmits the cached data packet. However, an alternative path may exclude the checkpoint node already selected from the source node during the transmission. Then the node that generates an *Alarm* packet randomly chooses a checkpoint node, piggybacks the id of checkpoint node into the cached data packet, and forwards the data packet towards the sink. Note that when a malicious node selects an alternative path, it may chooses a path which is far longer than the shortest or optimal path to intentionally increase the packet delivery latency, called vampire attack [24].

2) Active Detection: In the SCAD, a single checkpoint node generates an *Ack* packet and each intermediate node located along the forwarding path *passively* monitors any forwarding behavior of its next node. Similar passive monitoring-based approaches are also found in [5], [8]–[10], and [12]. Since the detection rate highly depends on how frequently malicious nodes conduct the forwarding misbehavior, it can be significantly reduced if multiple malicious nodes collude together. Thus, we consider a camouflage-based detection [13], in which each node pretends not to overhear on-going communication

but monitors the forwarding behavior of its adjacent nodes to detect a deep lurking malicious node. We plan to extend the SCAD by deploying an active detection approach, where each intermediate node hides its operational status (i.e., a checkpoint node), counts the number of forwarding misbehaviors, and selects the next forwarding node. A suspected node recorded with a high number of forwarding misbehaviors will not be chosen very often as a forwarding node.

In summary, we compare the proposed countermeasure with prior approaches and summarize their detection strategies in terms of six criteria in Table II.

VI. Concluding Remarks

In this paper, we proposed a light-weight countermeasure, called SCAD, to mitigate the forwarding misbehavior in WSNs. In the SCAD, a single checkpoint-assisted approach incorporated with timeout and retransmission techniques can efficiently improve the detection rate as well as reduce the energy consumption, false detection rate, and successful drop rate. The SCAD can achieve more than 90% PDR with less energy consumption compared to prior CHEMAS and CAD schemes. A simple analytical model of the SCAD and its numerical result in terms of false detection rate are also presented. To see the full potential of our approach, we discuss the design issues and possible extensions of the SCAD. The numerical and simulation results indicate that the proposed countermeasure is a viable approach in WSNs.

References

[1] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," *IEEE Comput.*, vol. 35, no. 10, pp. 54–62, Oct. 2002.

[2] B. Yu and B. Xiao, "Detecting selective forwarding attacks in wireless sensor networks," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, 2006, pp. 1–8.

[3] B. Xiao, B. Yu, and C. Gao, "CHEMAS: Identify suspect nodes in selective forwarding attacks," *J. Parallel Distrib. Comput.*, vol. 67, no. 11, pp. 1218–1230, 2007.

[4] Y. Kim, H. Lee, K. Cho, and D. Lee, "CADE: Cumulative acknowledgment based detection of selective forwarding attacks in wireless sensor networks," in *Proc. Int. Conf. Convergence Hybrid Inf. Technol.*, 2008, pp. 416–422.

[5] D. M. Shila, C. Yu, and T. Anjali, "Mitigating selective forwarding attacks with a channel-aware approach in WMNs," *IEEE Trans. Wireless Commun.*, vol. 9, no. 5, pp. 1661–1675, May 2010.

[6] Q. Liu, J. Yin, V. Leung, and Z. Cai, "FADE: Forwarding assessment based detection of collaborative grey hole attacks in WMNs," *IEEE Trans. Wireless Commun.*, vol. 12, no. 10, pp. 5124–5137, Oct. 2013.

[7] A. Varga, *OMNeT++*, 2014 [Online]. Available: https://www.omnetpp.org

[8] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proc. ACM MOBICOM*, 2000, pp. 255–265.

[9] D. R. Raymond and S. F. Midkiff, "Denial-of-service in wireless sensor networks: Attacks and defense," *IEEE Pervasive Comput.*, vol. 7, no. 1, pp. 74–81, Mar. 2008.

[10] T. H. Hai and E. Huh, "Detecting selective forwarding attacks in wireless sensor networks using two-hops neighbor knowledge," in *Proc. IEEE Int. Symp. Netw. Comput. Appl.*, 2008, pp. 325–331.

[11] J.-M. Chang, P.-C. Tsou, I. Woungang, H.-C. Chao, and C.-F. Lai, "Defending against collaborative attacks by malicious nodes in MANETs: A cooperative bait detection approach," *IEEE Syst. J.*, vol. 9, no. 1, pp. 65–75, Mar. 2015.

[12] S. Lim and H. Lauren, "Hop-by-Hop cooperative detection of selective forwarding attacks in energy harvesting wireless sensor networks," in *Proc. Int. Conf. Comput. Netw. Commun. (ICNC)*, 2015, pp. 315–319.

[13] C. Pu and S. Lim, "Spy vs. Spy: Camouflage-based active detection in energy harvesting motivated networks," in *Proc. Mil. Commun. Conf. (MILCOM)*, 2015, pp. 903–908.

[14] C. Pu, T. Gade, S. Lim, M. Min, and W. Wang, "Lightweight forwarding protocols in energy harvesting wireless sensor networks," in *Proc. Mil. Commun. Conf. (MILCOM)*, 2014, pp. 1053–1059.

[15] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proc. ACM MOBICOM*, 2000, pp. 264–275.

[16] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," in *Proc. ACM MOBICOM*, 2000, pp. 243–254.

[17] W. Stallings, *Cryptography and Network Security—Principles and Practices*, 6th ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2013.

[18] A. Boulis, *Castalia*, 2014 [Online]. Available: https://castalia.forge.nicta.com.au/index.php/en/

[19] X. Tang and J. Xu, "Extending network lifetime for precision-constrained data aggregation in wireless sensor networks," in *Proc. IEEE INFOCOM*, 2006, pp. 1–12

[20] S. Djahel, F. Nait-Abdesselam, and Z. Zhang, "Mitigating packet dropping problem in mobile ad hoc networks: Proposals and challenges," *IEEE Commun. Surv. Tuts.*, vol. 13, no. 4, pp. 658–672, Fourth Quart. 2011.

[21] I. Khalil and S. Bagchi, "Stealthy attacks in wireless ad hoc networks: detection and countermeasure," *IEEE Trans. Mobile Comput.*, vol. 10, no. 8, pp. 1096–1112, Aug. 2011.

[22] J. Deng, R. Han, and S. Mishra, "Insens: Intrusion-tolerant routing for wireless sensor networks," *Computer Commun.*, vol. 29, no. 2, pp. 216–230, 2006.

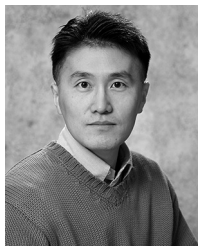
[23] Q. Fang, J. Gao, and L. J. Guibas, "Locating and bypassing holes in sensor networks," *Mobile Netw. Appl.*, vol. 11, no. 2, pp. 187–200, 2006.

[24] E. Y. Vasserman and N. Hopper, "Vampire attacks: Draining life from wireless ad hoc sensor networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 2, pp. 318–332, Feb. 2013.

Cong Pu (S'15) received the B.S. degree in computer science and technology from Zhengzhou University, Zhengzhou, China, in 2009, the M.S. degree in computer science from Texas Tech University, Lubbock, TX, USA, in 2013.

He is currently a Ph.D. candidate and a graduate part-time instructor with the Department of Computer Science, Texas Tech University. His research interests include wireless sensor networks, energy harvesting motivated networks, vehicular *ad hoc* networks, and denial-of-service attacks.

He was the recipient of the 2015 Texas Tech Helen DeVitt Jones Excellence in Graduate Teaching Award.



Sunho Lim (M'01) received the B.S. degree (*summa cum laude*) in computer science and the M.S. degree in computer engineering from Hankuk Aviation University (a.k.a. Korea Aerospace University), Goyang, Korea, in 1996 and 1998, respectively, and the Ph.D. degree in computer science and engineering from Pennsylvania State University, State College, PA, USA, in 2005.

He is currently an Assistant Professor with the Department of Computer Science, Texas Tech University (TTU). Lubbock, TX, USA. Before joining TTU, he was an Assistant Professor with the Department of Electrical Engineering and Computer Science, South Dakota State University, Brookings, SD, USA, from 2005 to 2009. His research interests include areas of cybersecurity, mobile data management and privacy, and wireless networks and mobile computing.

Dr. Lim was a Guest Editor of special issue on dependability and security for wireless *ad hoc* and senor networks and their applications in *International Journal of Distributed Sensor Networks*, and has served on the program committees of many conferences. He is leading the T2WISTOR: TTU Wireless Mobile Networking Laboratory. He was the recipient of the 2015 Texas Tech Alumni Association New Faculty Award, the 2013 Air Force Summer Faculty Fellowship, and the South Dakota Governor's 2010 Individual Research Seed Grant.