Computer Communications 124 (2018) 17–30

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

EYES: Mitigating forwarding misbehavior in energy harvesting motivated networks

Cong Pu[*,1,a], Sunho Lim[2,b], Byungkwan Jung[3,b], Jinseok Chae[1,c]

[a] Weisberg Division of Computer Science, Marshall University, Huntington, WV 25755, United States
[b] T[2]WISTOR: TTU Wireless Mobile Networking Laboratory, Department of Computer Science, Texas Tech University, Lubbock, TX 79409, United States
[c] Department of Computer Science and Engineering, Incheon National University, Incheon 22012, South Korea

ARTICLE INFO

Keywords:
Denial-of-Service
Energy harvesting motivated networks
Forwarding misbehavior

ABSTRACT

Energy harvesting motivated networks (EHNets) has been becoming increasingly popular in the presence of Internet-of-Things (IoT). Each self-sustainable node periodically harvests energy from an immediate environment but it is admittedly vulnerable to a Denial-of-Service (DoS) attack in the EHNets. In this paper, we propose a novel countermeasure, called EYES, to the forwarding misbehavior of multiple colluding malicious nodes in the realm of EHNets. Under the charge-and-spend harvesting policy, we first establish a set of adversarial scenarios, analyze its forwarding operations, and identify vulnerable cases. The EYES consists of two schemes, *SlyDog* and *LazyDog*, and cooperatively detects the forwarding misbehavior. In the SlyDog, each node actively disguises itself as an energy harvesting node and stealthily monitors the forwarding operations of adjacent nodes. In the LazyDog, however, each node periodically requests the number of overheard packets from its adjacent nodes and validates any prior uncertain forwarding operation. The combination of two schemes can efficiently detect the forwarding misbehaviors of colluding malicious nodes and quickly isolate them from the network. We also present a simple analytical model and its numerical result in terms of detection rate. We evaluate the proposed countermeasure through extensive simulation experiments using the OMNeT++ and compare its performance with two existing schemes, the hop-by-hop cooperative detection (HCD) and Watchdog. Simulation results show that the EYES provides 70–92% detection rate and achieves 23–60% lower detection latency compared to the HCD and Watchdog. The EYES also shows a competitive performance in packet delivery ratio.

1. Introduction

Internet-of-Things (IoT) and its applications are rapidly proliferating, where a myriad of multi-scale sensors and devices (later in short, nodes) are seamlessly blended [1]. Nodes are resource constrained in terms of computing, storage, and battery power but they are often required to operate a long-term sensing and communication in a remote or unattended area. Since wireless communication could be responsible for more than half of total energy consumption [2], a significant amount of effort has been devoted to develop energy efficient mechanisms in resource constrained networks [3]. Due to the limited battery power, however, it is ultimately unavoidable to replace or replenish batteries. In order to remove batteries or at least reduce the frequency of replacing batteries, energy harvesting from an immediate environment (e.g., solar, wind, thermal, kinetic, etc.) has been becoming increasingly popular and is being deployed for IoT [4–6]. We envision that an energy harvesting motivated network (EHNet), where a set of self-sustainable nodes equipped with energy harvesting capabilities communicate directly or indirectly via multi-hop relays, will be a major part of ubiquitous computing and communication infrastructure.

Due to the shared medium and lack of resource and centralized coordination, however, EHNet is indeed vulnerable to a Denial-of-Service (DoS) attack [7]. A malicious node compromised by an adversary can easily overhear, duplicate, corrupt, or alter an on-flying packet. In particular, the malicious node can also randomly or selectively drop any incoming packet to disrupt network protocols and interfere with on-going communications on purpose or strategically. Note that it is not trivial to differentiate such a misbehavior (or attack) from a temporal node failure or packet loss. Diverse countermeasures and

* Corresponding author.
E-mail addresses: puc@marshall.edu (C. Pu), sunho.lim@ttu.edu (S. Lim), byung.jung@ttu.edu (B. Jung), jschae@inu.ac.kr (J. Chae).
[1] Member, IEEE.
[2] Senior Member, IEEE.
[3] Student Member, IEEE.

https://doi.org/10.1016/j.comcom.2018.04.007
Received 8 December 2016; Received in revised form 28 February 2018; Accepted 9 April 2018
Available online 12 April 2018
0140-3664/ © 2018 Elsevier B.V. All rights reserved.

their variants [8–21] have been proposed to avoid and/or detect the misbehavior but they implicitly assume a battery-powered network.

In this paper, we investigate a forwarding misbehavior in EHNets, where a set of self-sustainable nodes periodically harvests energy and repeats on- and off-periods for communication. We identify a new type of selective forwarding attacks, propose a cooperative countermeasure to mitigate the forwarding misbehavior, and build a simple analytical model of the countermeasure. In the proposed countermeasure, we consider multiple malicious nodes that operate as legitimate nodes for most of the time and drop any incoming packet during a vulnerable period. Our major contribution is summarized in three-fold:

- First, we investigate a set of adversarial scenarios and analyze its forwarding operations under the *charge-and-spend* energy harvesting policy in EHNets. Then we identify four vulnerable scenarios and their corresponding potential forwarding misbehaviors.
- Second, we propose a cooperative countermeasure to efficiently detect the forwarding misbehavior in EHNets, called *EYES*, consisting of two mechanisms: *SlyDog* and *LazyDog*. In the SlyDog, each node actively disguises itself as an energy harvesting node but it in fact monitors its adjacent nodes to detect the forwarding misbehavior of lurking deep malicious nodes. In the LazyDog, however, each node periodically requests its adjacent nodes of a limited history of forwarding operations, and validates any prior uncertain forwarding operation to detect the forwarding misbehavior.
- Third, we propose an analytical model of the EYES and show its numerical results in terms of detection rate. We also revisit prior detection approaches, hop-by-hop cooperative detection (HCD) [22] and Watchdog [8], and modify them to work in EHNets. Both single and two malicious nodes cases are applied to the HCD and Watchdog, and no malicious node case is also considered as the performance upper bound of packet delivery ratio. In addition, detection strategies of forwarding misbehavior are comprehensively compared in terms of six properties.

We conduct extensive simulation experiments using the OMNeT++ [23] for performance evaluation and analysis. Compared to the HCD and Watchdog, the EYES can not only efficiently detect forwarding misbehavior but also significantly improve the performance in terms of detection rate, detection latency, and packet delivery ratio.

The rest of paper is organized as follows. Prior detection strategies of forwarding misbehavior are reviewed and analyzed in Section 2. Section 3 describes both system and adversarial models. The proposed adversarial scenarios and countermeasure are presented in Sections 4 and 5, respectively. An analytical model of the proposed countermeasure is presented in Section 6. Section 7 is devoted for performance evaluation and analysis. In Section 8, we discuss the applicability of proposed approach to other attacks and consider its possible enhancements. Finally, we conclude the paper with future research direction in Section 9.

2. Related work

Both Watchdog and Pathrater techniques [8,9] and their variants have been widely deployed in different types of networks. A watchdog technique detects a misbehaving node by overhearing its transmission to see whether it forwards a packet with some maximum delay. Simple watchdog and pathrater techniques are extended in implicit acknowledgment and overhearing, in which each node monitors its neighbor nodes communication activities, such as forwarding operations. Since nodes are required to stay in an active mode, it is not feasible especially in a battery-powered network because of non-negligible energy consumption. In this section, we categorize and analyze existing detection techniques against forwarding misbehaviors in terms of monitor-, acknowledgment-, inducement-based, and other approaches.

2.1. Monitor-based approach

Each node observes the network condition and communication activities, such as a channel condition, network traffic, or forwarding operation of its adjacent nodes, and checks if there is any abnormality. In the channel-aware detection (CAD) [14], both channel estimation and traffic monitoring are conducted to identify a stand-alone attacker in wireless mesh networks (WMNs). Each node monitors the communication activities of its adjacent nodes by observing downstream and upstream network traffic and estimates packet loss rate. If a node shows higher monitored packet loss rate than the sum of estimated packet loss rate and its corresponding detection threshold, it is suspected as an attacker. A channel-aware reputation system (CRS) [18] is an extended version of the CAD, where each node maintains a reputation table to evaluate the forwarding behavior of its adjacent nodes. This reputation value is calculated based on the deviation of the normal packet loss rate caused by the time- and location-variant channel quality and the link layer collisions as well as the monitored packet loss rate during a long term. The adjacent node is prosecuted as a malicious node if its reputation value is less than the predefined threshold value. Chae et al. [24] proposes a countermeasure to on-off attacks with selective forwarding, in which a forwarding misbehavior is seldom detected and is confused with a temporary error. Each node monitors the forwarding operation of its adjacent nodes and records good and bad behaviors based on a dynamic sliding window, respectively. This scheme can recognize a pattern of malicious node behavior and support to flexibly design a system that can accept a certain level of security risk based on the accumulated behaviors. In the HCD [22], each node records a limited set of traces about forwarding operations and exchanges it with its adjacent nodes to identify a forwarding misbehavior in EHNets. Each node can gradually reduce the forwarding probability of malicious node in order to exclude the malicious node from participating in the routing operation. In the CRS with adaptive detection threshold (CRS-A) [25], each node maintains a reputation table with adaptive detection threshold to evaluate the forwarding behavior of its adjacent nodes in wireless sensor networks (WSNs). The reputation value is calculated based on the deviation of the monitored packet loss rate as well as the estimated normal loss rate caused by the time- and location-variants channel quality and the link layer collisions. The nodes with low reputation values are detected and isolated from the routing. A monitor-based approach (CMD) [26] is proposed to mitigate the forwarding misbehaviors in low power and lossy networks (LLNs), where each node monitors the forwarding behaviors of the preferred parent node to observe the packet loss rate, compares the observation result with the collected packet loss rate from one-hop neighbor nodes, and detects the forwarding misbehaviors of the preferred parent node.

2.2. Acknowledgment-based approach

The key operation is that the intermediate nodes located along the forwarding path between source and destination (e.g., sink) are responsible for monitoring the forwarding operation of its next node and sending an explicit message (i.e., acknowledgment (*Ack*) packet) to the source. In [11] and its extension, called checkpoint-based multi-hop acknowledgment scheme (CHEMAS) [12], a set of checkpoint nodes is randomly selected from a source per packet basis and monitors the forwarding operation by replying an *Ack* packet to the source in WSNs. If an intermediate node located in the forwarding path does not receive the required number of *Ack* packets, it suspects the next located node in the path as a malicious node, generates an *Alarm* packet, and sends it back to the source. However, since multiple number of checkpoint nodes generate *Ack* packets, intermediate nodes may receive and forward the excessive number of packets and consume non-negligible amount of energy. A single checkpoint-based countermeasure (SCAD) [21] is proposed to detect a selective forwarding attack in resource-constrained WSNs, where a randomly selected single checkpoint node

along the forwarding path is deployed to detect forwarding misbehaviors. The proposed countermeasure is integrated with timeout and hop-by-hop retransmission techniques to efficiently cover unexpected packet losses due to the forwarding misbehavior or bad channel quality. A forwarding assessment based detection (FADE) [17] is a variant of the CAD and detects a collaborative selective forwarding attack in WMNs. After each node forwards a packet, it overhears a link-layer acknowledgment and waits for a two-hop acknowledgment from its downstream nodes. Each node also adds its acknowledging information towards the downstream nodes to a separate monitoring packet originally sent from source.

2.3. Inducement-based approach

The basic idea is that a piece of information is hidden or a fake information is utilized to lure the potential malicious nodes to show its possible forwarding misbehavior. A cooperative bait detection scheme (CBDS) [19] based on the dynamic source routing (DSR) is proposed to detect both selective forwarding and blackhole attacks in mobile ad hoc networks (MANETs). This approach 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. Then the malicious node can be identified by using a reverse tracing technique. In the sequence number based bait detection scheme (SNBDS) [20], each node observes the difference between the sequence numbers of the received RREP packets and that of stored in the routing table based on the ad hoc on-demand distance vector routing (AODV) to detect the next hop located node in MANETs. If the maximum difference is larger than the predefined threshold value, the next node is added to a suspicious node table for malicious node discovery and verification process by using fictitious destination address and destination sequence number. In the camouflage-based active detection scheme (CAM) [27], each node hides its current operational status and pretends not to overhear the on-going communications, but in fact monitors the forwarding operations of its adjacent nodes to detect a deep lurking malicious node in EHNets. Since malicious nodes are seldom in harvest state and can selectively drop any incoming packet in a short period of time, it is not trivial to detect the forwarding misbehavior.

2.4. Other approach

In [28], a fine-grained analysis (FGA) is conducted to investigate the cause of packet loss. The FGA profiles wireless links between nodes as well as their adjacent nodes by leveraging resident parameters based on the received signal strength indicator and link quality indicator. According to the profiles, the FGA can determine whether packet loss is caused by an attacker or not. An audit-based misbehavior detection (AMD) [29] is proposed to isolate continuous or selective packet droppers. The AMD integrates reputation management, trustworthy route discovery, and identification of misbehaving nodes based on behavioral audits. Node behavior is evaluated based on per-packet basis, without employing energy-expensive overhearing or intensive acknowledgment techniques. A collaborative detection approach [30] is proposed to countermeasure a selective forwarding attack, where each node is monitored and evaluated by adjacent nodes in two time windows. At the end of each time window, all monitored neighbor nodes are evaluated by an introduction detection system (IDS) node and if an attack was detected, a voting process is executed to identify a malicious node.

In summary, most prior countermeasures rely on an implicit overhearing that requires nodes to stay in active state for an extended period in battery-supported networks. Most explicit acknowledgment based approaches also force intermediate nodes to generate and forward a large number of packets (e.g., *Ack* packet), resulting in additional energy consumption. Little attention has been paid for self-sustainable nodes in the realm of EHNets, where each node operates under the charge-and-spend harvesting policy. The proposed countermeasure deploys combined monitor- and inducement-based approaches.

3. System and adversarial models

In this section, we first present a system model in terms of energy harvesting motivated node and its state, energy harvesting policy, and a simple data forwarding approach. Then we present an adversarial model in terms of a potential misbehavior of malicious node in the network.

In this paper, we consider an energy harvesting motivated network (EHNet), where each node is assumed to equip with an energy harvesting device to replenish its rechargeable battery [31] and its processing power and memory storage are not a major concern. For example, a piezoelectric fiber composite bi-morph (PFCB) W14 based energy harvesting from an immediate environment (e.g., disturbance or typical body movements) can generate sufficient power (i.e., 1.3 mW – 47.7 mW) for wireless sensors[4] [34–36]. It is envisaged that multi-scale piezo devices and integrated self-charging power cells (SCPCs) [37] will enhance the efficiency of energy harvesting.

In EHNet, an energy harvesting process is modeled as a two-state Markov process with active (s_a) and harvest (s_h) states. Each node stays in either active or harvest state for a certain period of time, which is exponentially distributed with a mean λ_a (e.g., between 50 and 80) or λ_h (e.g., between 15 and 40) respectively, and changes to the other state. In this paper, each node has the same λ_a or λ_h in active or harvest state, respectively. Here, λ_a and λ_h are system parameters and their changes and impact on the performance are observed in Section 7. Note that frequent state changes incur a non-negligible on-off switch cost in terms of energy consumption and operational delay. Thus, we also adopt the *charge-and-spend* energy harvesting policy [22,27,38,39], where a node in harvest state is unable to listen and receive any packet until a certain level of energy is harvested. Although the node minimizes its communication activity during harvest state, it periodically broadcasts a one-hop *State* packet to prevent other nodes from mistakenly forwarding a packet to the nodes in harvest state. We observe the impact of *State* packet intervals and number of neighbor nodes on packet delivery ratio (PDR) in Fig. 1, where both uniform and exponential packet intervals are used. Short packet intervals show the low PDR because frequently broadcasted *State* packets can be collided with *Data* packets. As the number of neighbor nodes increases, the PDR reduces because more nodes may broadcast *State* packets in harvest state, resulting in more collisions with *Data* packets. When the packet interval is close to 1.0 (s), the PDR is still above 80% even with the large number of neighbor nodes. This packet interval is acceptable without significantly affecting the performance in EHNets.

When a node detects an event, it initiates to generate and forward a data packet toward a sink. A simple broadcast-based forwarding [40] can be deployed to deliver the data packet toward the sink. To realize this, the sink broadcasts a one-time *Hello* packet piggybacked with a number of hops (initially zero) during a network deployment phase. Whenever a node receives the *Hello* packet, it increments the number of hops by one and rebroadcasts the packet, only if it receives the packet first or the packet has a shorter number of hops. Then each node can be aware of its one-hop neighbor nodes and how many hops away from the sink, and forward the *Data* packet to a node that is closely located to a sink.

The primary goal of adversary is to attack service availability by

[4] For instance, the IEEE 802.15.4-compliant Texas Instrument Chipcon CC2420 radio [32] supports eight different transmission power levels ranging from 3 μW to 1 mW. The IEEE 802.11 a/b/g-compliant Cisco Aironet 340 and 350 series [33] also support four (1, 5, 10, and 30 mW) and six (1, 5, 20, 30, 50, and 100 mW) transmission power levels, respectively.

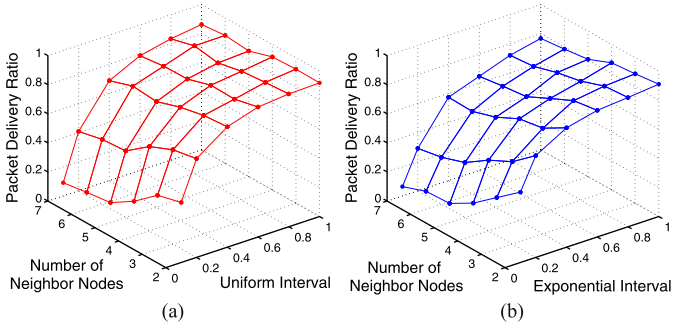


Fig. 1. The impact of uniform and exponential *State* packet intervals and number of neighbor nodes.

disrupting network protocols or interfering with on-going communications. An adversary is able to capture and compromise a legitimate node to behave maliciously. A malicious node involved in packet forwarding operation may selectively or strategically drop or forward any packet to deafen a sink. The malicious node may also eavesdrop on an on-flying packet and inject false information or modify its packet header information to mislead network traffic. However, if a sender authenticates a packet with a light-weight digital signature [41], a receiver can easily verify the packet and detect any modification. In this paper, we consider a network, where there is at least more than one neighbor node to forward a packet. Two sub-networks connected with a single node are not considered because it can be a malicious node or a single-point of failure. We assume that a malicious node has no energy constraints and it can stay in active state for an extended period. In this paper, we primarily focus on the selective forwarding attack and energy harvesting motivated adversarial scenarios that cannot be detected by digital signature and cryptographic techniques. We do not consider cryptographic primitives.

4. Energy harvesting motivated attack scenarios and analyses

In this section, we investigate potential forwarding misbehaviors by identifying a set of adversarial scenarios and observing its vulnerable cases in EHNets. We present eight adversarial scenarios using a snapshot of network consisting of five energy harvesting enabled nodes in Fig. 2, where two malicious nodes (n_{m_A} and n_{m_B}) are located along the forwarding path, monitor network traffic, and collude selective forwarding attacks together. The different states of legitimate and malicious nodes corresponding to the adversarial scenarios is also presented in Table 1. Since the proposed adversarial scenarios are carefully chosen by observing the interactions between legitimate and malicious nodes in EHNets, we focus on the cases in which two malicious nodes are located along the forwarding path in the network. This is primarily because two malicious nodes can easily collude together and conduct selective forwarding attacks without being detected. We do not solely consider interleaved malicious nodes in the network, for example, where each malicious node is located along the forwarding path every other legitimate node. There are prior approaches [12,21] to countermeasure the forwarding misbehavior of interleaved malicious nodes. In addition, we expect that IoT and its variants, where highly populated legitimate nodes are interconnected together for communications, will reduce the ratio of malicious nodes (or compromised nodes) to legitimate nodes. However, we investigate how to countermeasure a simple collusion with a small number of malicious nodes that can conduct selective forwarding attacks and significantly impact the network performance in this paper.

Table 1
The summary of states for legitimate and malicious nodes in the adversarial scenarios.

n_a	n_{m_A}	n_b	n_{m_B}	n_c	Scenario
Active	Active	Active	Active	Active	Fig. 2(a)
Active	Active	Active	Harvest	Active	Fig. 2(b)
Active	Active	Active	Active	Harvest	Fig. 2(c)
Active	Active	Active	Harvest	Harvest	Fig. 2(d)
Active	Active	Harvest	Active	Active	Fig. 2(e)
Active	Active	Harvest	Harvest	Active	Fig. 2(f)
Active	Active	Harvest	Active	Harvest	Fig. 2(g)
Active	Active	Harvest	Harvest	Harvest	Fig. 2(h)

Suppose a node (n_a) forwards a *Data* packet to n_c through intermediate nodes, n_b, n_{m_A}, and n_{m_B} as shown in Fig. 2. n_b and n_{m_A} are one-hop neighbor nodes of n_a, after sending a Data packet, n_a can monitor the subsequent forwarding operation of n_b and n_{m_A}. n_b is the common neighbor nodes of n_a, n_{m_A}, and n_{m_B}, and it can overhear and monitor the forwarding operation of all these three nodes. However, n_b cannot

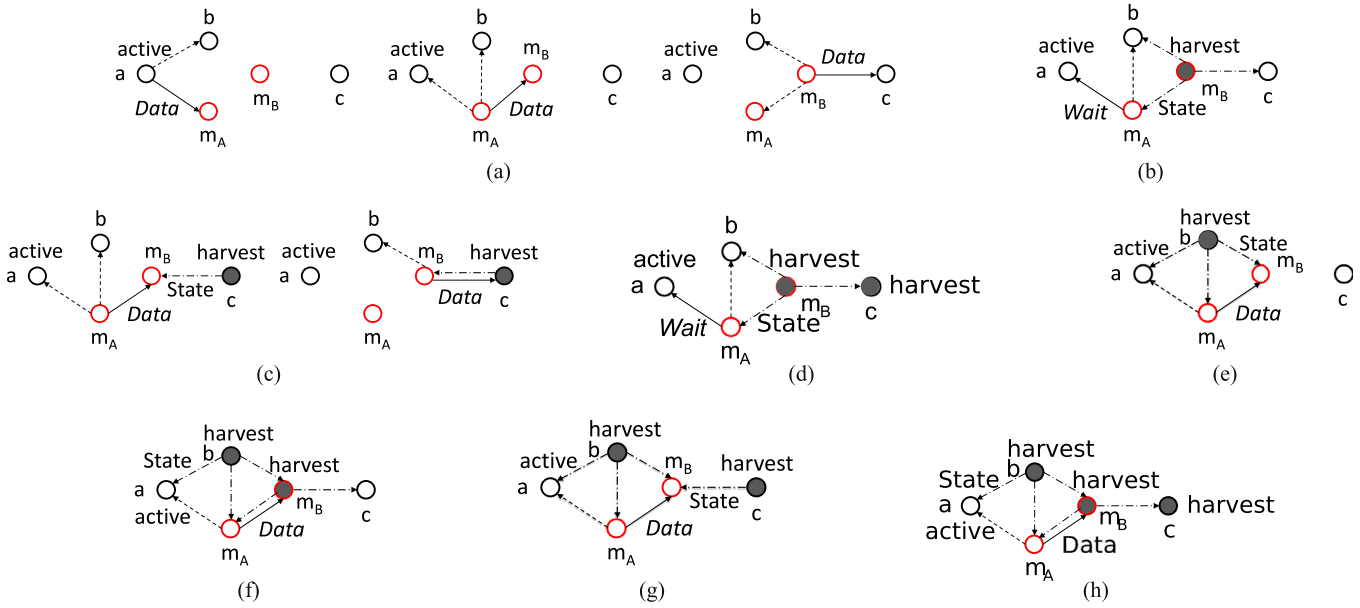


Fig. 2. A set of adversarial scenarios and its vulnerable cases in the presence of malicious nodes in EHNets. Here, a malicious node and a node in harvest state are marked as red and shade, respectively. Solid, dotted, and dash-dotted lines represent forwarding, overhearing, and periodic broadcast operations, respectively.

overhear the packet forwarding through n_c, since n_c is two-hop neighbor node of n_b and can only receive the Data packet from n_{m_B}. Here, we implicitly assume that packet sender n_a is always in active state, otherwise, it cannot send the Data packet. In order to see the potential forwarding misbehavior, we show that n_a selects n_{m_A} as a forwarding node. We also assume that n_{m_A} is always in active state, otherwise, the packet sender n_a will select another active neighbor node, e.g., n_b, as a forwarding node. In this paper, for the sake of simplicity, we focus on three nodes (n_b , n_{m_B}, and n_c) and their states (active and harvest), leading to eight representative adversarial scenarios. However, most forwarding misbehaviors can be covered and categorized into one of our proposed adversarial scenarios. For example, although we consider two forwarding nodes (e.g., n_b and n_{m_A}), it could be extended to multiple nodes mixed with normal and malicious nodes. In order to see the potential forwarding misbehavior, we show that n_a selects n_{m_A} as a forwarding node. Similarly, more than two malicious nodes can consecutively be located for potential collusions. Multiple normal and malicious nodes can be located around malicious nodes and overhear on-flying packets. Note that we do not pursue a formal proof for possible adversarial scenarios with any number of nodes, which is out of scope of this paper.

4.1. Potential forwarding behaviors

We show two adversarial scenarios, where malicious nodes can potentially show their forwarding misbehavior. First, when a packet sender (e.g., n_a, n_{m_A}, or n_{m_B}) forwards a received Data packet, its neighbor nodes (e.g., n_a, n_b, or n_{m_A}) can overhear and store it in its local cache as shown in Fig. 2(a). If n_{m_A} drops the packet on purpose, n_b cannot overhear it within a timeout period and forwards its cached copy to n_{m_B}. If n_a overhears the packet forwarded from n_b, which is different from the original forwarder (n_{m_A}), it suspects the forwarding behavior of n_{m_A}. Thus, n_{m_A} does not drop the packet when n_a and n_b are in active state. When n_{m_B} forwards the packet to n_c, both n_b and n_{m_A} can overhear it. Then n_b assumes that n_{m_B} has successfully forwarded the packet to the next hop, n_c.

Second, n_{m_B} switches to harvest state on purpose and periodically broadcasts a State packet as shown in Fig. 2(b). If n_{m_A} forwards a received Data packet to n_{m_B}, the packet will be lost because n_{m_B} is in harvest state and unable to receive any incoming packet. However, this forwarding misbehavior can be detected because n_b is in active state and can overhear the packet forwarded. Thus, n_{m_A} does not forward the packet on purpose but holds it until n_{m_B} switches back to active state. Instead n_{m_A} replies a Wait packet to the packet sender, n_a, to delay the packet transmission. This Wait packet is used to inform the packet sender of the state of next hop node, which is in harvest state and cannot receive any incoming packet. Upon receiving the Wait packet, n_a selects an alternative forwarding node, such as n_b. If n_c is in harvest state instead of active, as shown in Fig. 2(d), n_{m_A} still need to perform the same operations as mentioned above, otherwise, the forwarding misbehavior can be easily detected.

4.2. Undetected vulnerable cases

In the following five adversarial scenarios, we show the forwarding misbehaviors of malicious nodes that cannot be detected. First, suppose n_b is in harvest state and periodically broadcasts a *State* packet as shown in Fig. 2(e). If n_{m_A} drops a received *Data* packet on purpose, n_a can suspect n_{m_A} of the forwarding misbehavior when a timeout period expires. If n_{m_A} simply forwards the packet to n_{m_B}, which will hold it without forwarding to the next hop node, the packet will be lost without being detected. Since n_b is in harvest state and n_c cannot overhear the packet, the forwarding misbehaviors of n_{m_A} and n_{m_B} cannot be detected.

Second, both n_b and n_{m_B} are in harvest state and periodically broadcast a State packet as shown in Fig. 2(f). Since only n_a can overhear the packet, n_{m_A} simply forwards the packet to n_{m_B}, resulting in packet loss without being detected. Although n_a can overhear the packet, the forwarding misbehavior of n_{m_A} cannot be detected. If n_c is in harvest state instead of active, as shown in Fig. 2(h), n_{m_A} can perform same operations mentioned above to drop the packet without being detected.

Third, n_c is in harvest state and periodically broadcasts a *State* packet as shown in Fig. 2(c). Since both n_a and n_b are in active state, n_{m_A} forwards a received Data packet to n_{m_B}. If n_{m_B} holds the packet without forwarding to the next hop node, n_b can suspect n_{m_B} of the forwarding behavior when a timeout period expires. In case of when n_c is in harvest state, n_{m_B} simply forwards the packet to n_c, resulting in packet loss without being detected.

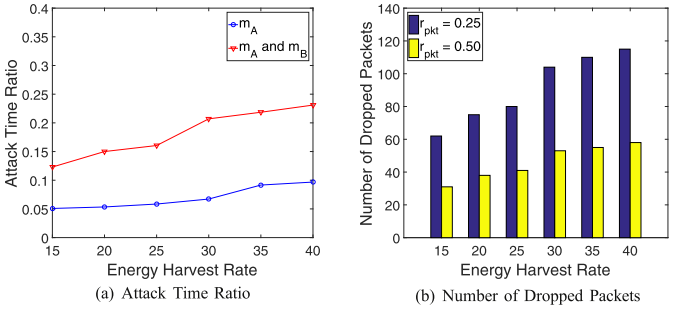
Lastly, both n_b and n_c are in harvest state and periodically broadcast a *State* packet as shown in Fig. 2(g). n_{m_B} can either forward the packet to n_c or hold the packet without forwarding to the next hop node, resulting in packet loss without being detected. This is because only n_{m_A} can overhear the packet.

4.3. Lurking deep malicious nodes

Based on the aforementioned five undetected vulnerable cases, we measure the number of dropped packets and how frequently malicious nodes can collude together and conduct undetected forwarding misbehaviors in terms of *attack time ratio* (ATR), $\frac{t_{at}}{t_{tot}}$, in Fig. 3. Here, t_{at} and t_{tot} are total attack time of undetected forwarding misbehaviors and total observation time (e.g., 1000 (s)), respectively. t_{at} is measured by accumulating the periods when either adjacent node (n_b) or receiver (n_c), or both of them are in harvest state as shown in Fig. 2(c), (e), (f), (g) and (h). In Fig. 3(a), the ATR of n_{m_A} slightly increases from 5% to 10% as energy harvest rate increases. However, the ATR of two colluding malicious nodes (n_{m_A} and n_{m_B}) can quickly increase upto 24%. As nodes stay in harvest state for longer period, the chance of malicious nodes conducting the forwarding misbehaviors without being detected increases. In Fig. 3(b), the number of dropped packets is measured against energy harvest rate and packet injection rate (r_{pkt}). More number of packets are dropped with smaller $r_{pkt} = 0.25$ (pkt/s). This is because two colluding malicious nodes receive more packets with smaller r_{pkt}, resulting in more packets dropped due to undetected forwarding misbehaviors.

5. The proposed countermeasure

We propose a countermeasure, called *EYES*, to efficiently detect forwarding misbehaviors of colluding malicious nodes in the EHNets. The proposed countermeasure consists of two schemes: *SlyDog* and *LazyDog*. The EYES is different from the prior approaches, [8–18,21], where each node only *passively* monitors any forwarding misbehavior witnessed in an adversarial case for detection in the battery-powered networks.



(a) Attack Time Ratio (b) Number of Dropped Packets

Fig. 3. The changes of attack time ratio and number of dropped packets against energy harvest rate and packet injection rate.

5.1. SlyDog: inducement-based detection

The basic idea of SlyDog is that each node *actively* disguises itself as an energy harvesting node on purpose and pretends not to overhear any on-flying packet. However, each node in fact stealthily monitors any forwarding operation of its adjacent nodes to detect a lurking deep malicious node. Here, the SlyDog is significantly extended from our previous work, called CAM [27], to detect a collusion of malicious nodes.

5.1.1. Basic operations

We present four basic operations and their corresponding information to transceive and maintain in the SlyDog. First, when a node receives a *Data* packet, it randomly selects one of its adjacent nodes as a forwarding node. If none of adjacent nodes is in active state, the node replies a *Wait* packet to the prior packet sender and caches the *Data* packet in its local storage. When the node receives a *State* packet from an adjacent node in active state, it forwards the cached *Data* packet. When a node switches its state, it broadcasts a one-time *State* packet. If the node is in harvest state, however, it periodically broadcasts a *State* packet. Since the node in harvest state is unable to receive any incoming packet based on the charge-and-spend harvesting policy, this periodic *State* packet prevents its adjacent nodes from mistakenly forwarding a packet. The node in active state does not periodically broadcast a *State* packet. Here, a *State* packet consists of three components: node id (*nid*), state ($s \in \{s_a, s_h\}$), and timestamp (t_{cur}). When a node receives a *State* packet, it records the packet in a state trace table (*ST*). For example, when a node n_b receives a *State* packet from n_a, it updates the state of n_a (s), $ST_b = ST_b \cup [a, s, t_{cur}]$. If n_b receives a *State* packet from n_a again but the state of n_a has not been changed, it discards the packet without updating the table. The operations of SlyDog are summarized in Fig. 5.

Second, each node also maintains an audit table (*AT*), where each entry consists of five components: one-hop neighbor node's id (*sid*), two-hop neighbor node's id (*rid*), and number of overheard packets sent from one-hop neighbor node to two-hop neighbor node (*op*) during a time interval between t_{begin} and t_{end}. For example, when a node n_a overhears a packet sent from its one-hop neighbor node (n_b) to two-hop neighbor node (n_c), it sets t_{begin} to the current time and increases $AT_a[b, c].op$ by one. When a node switches to harvest state, it sets t_{end} to the current time.

Third, when a node detects a forwarding misbehavior, it records a number of forwarding misbehaviors of suspected node and updates its monitor probability. In this paper, a monitor probability indicates how actively a node monitors the forwarding operation of suspected node, and it is used to decide whether to perform the SlyDog on the suspected node. Initially, each node sets the equal monitor probability to all its one-hop neighbor nodes (G^*), $\frac{1}{|G^*|}$. Note that the rationale behind this initialization is to consider a network density. In a dense network, the probability decreases because more number of one-hop neighbor nodes are available to monitor the forwarding operation of suspected node. In a sparse network, however, the probability increases because a less number of neighbor nodes are available. A set of monitor probabilities is stored and updated in a monitor table (*MT*). Each entry of *MT* consists of three components: node id (*nid*), a number of detected forwarding misbehaviors (c_{mis}), and monitor probability (p).

Fourth, whenever a node detects a forwarding misbehavior, it increments the number of detected forwarding misbehaviors of suspected node by one and increases the monitor probability by δ. Here, δ is a system parameter and its impact on the performance is observed in Section 7. When the number of detected forwarding misbehaviors of a suspected node reaches a threshold τ, the node broadcasts an *Alarm* packet to its one-hop neighbor nodes to prevent the suspected node from being selected as a forwardee node. This isolation operation of malicious node is also summarized in Fig. 5.

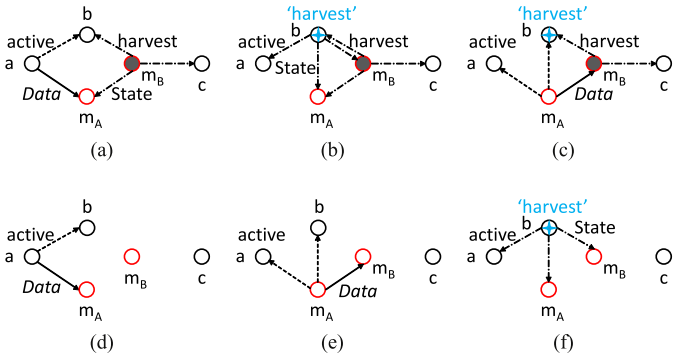


Fig. 4. The proposed *SlyDog* detection operations.

Notations:
- F_i: The set of forwardee nodes of n_i, e.g., F_a is $[n_b, n_{m_A}]$.
- FS_i: The set of active forwardee nodes of n_i.
- $ST_i[nid, s, t_{cur}]$, $MT[nid, c_{mis}, p]$, nid, s, t_{cur}, c_{mis}, p, τ, s_a: Defined before.
- $pkt[type, fwd, rec, seq]$: A packet is forwarded from n_{fwd} to n_{rec}, with sequence number, seq. Here, type is $Data$, $Wait$, or $Alarm$. If type is $Alarm$, rec is considered as malicious node id.

Operations:
◇ n_i overhears a $State$ packet of neighbor node, n_j:
  Update ST_i if the state of n_j changes.
◇ n_i receives a *Data* packet $pkt[Data, s, i, seq]$ from n_s:
  $FS_i = \emptyset$;
  for $n_k \in F_i$
    if $ST_i[k].s == s_a$ /* n_k *is in active state* */
      $FS_i = FS_i \cup n_k$;
  if $FS_i \neq \emptyset$ /* *At least one active forwardee node exists* */
    Randomly select a forwardee node n_f, $n_f \in FS_i$;
    Forward the *Data* packet $pkt[Data, i, f, seq]$ to n_f;
  else /* *No active forwardee node exists* */
    Cache the *Data* packet;
    Forward the *Wait* packet $pkt[Wait, i, s, seq]$ to n_s;
◇ n_i isolates the malicious node n_j from network:
  if $MT_i[j].c_{mis} >= \tau$
    Broadcast the *Alarm* packet $pkt[Alarm, i, j, seq]$;

Fig. 5. The pseudo code of SlyDog.

5.1.2. Detection operations

Under the basic operations, we present detection operations of SlyDog with a set of snapshots of networks in Fig. 4. First, suppose a node n_b is a legitimate node and overhears a *Data* packet sent from n_a to n_{m_A} as shown in Fig. 4(a). Then n_b checks the state of its one-hop neighbor nodes based on the state table, ST_b. In Fig. 4(b), if a forwardee node (n_{m_B}) is in harvest state, n_b decides whether to perform the SlyDog based on the monitor probability of n_{m_A}, p_{m_A}. n_b generates a random number (e.g., rand[0, 1]) and if it is less than or equal to p_{m_A}, then n_b performs the SlyDog on n_{m_A} and disguises itself as an energy harvesting node. n_b stealthily monitors the forwarding operation of n_{m_A} while periodically broadcasting a *State* packet piggybacked with its harvest state. In Fig. 4(c), when n_{m_A} overhears a harvest *State* packet from n_b, n_{m_A} believes that n_b is in harvest state currently, which is the aforementioned vulnerable case (see Fig. 2(f)). If n_{m_A} forwards the *Data* packet to n_{m_B} without replying a *Wait* packet back to the packet sender (n_a), this *Data* packet will be lost because n_{m_B} is in harvest state. However, this forwarding misbehavior of n_{m_A} can be detected by n_b. If n_b does not perform the SlyDog on n_{m_A}, it stays in active state and monitors the forwarding behavior of n_{m_A}. If n_{m_A} replies a *Wait* packet, it is considered as a legitimate node. Upon overhearing the *Wait* packet, n_b broadcasts a *State* packet piggy backed with its active state and stops performing the SlyDog on n_{m_A}.

Second, suppose both n_b and n_{m_B} stay in active state as shown in

Notations:

- S_i: The set of packet senders of n_i, e.g., S_b is $[n_a]$.
- F_i: The set of forwardee nodes of n_i, e.g., F_a is $[n_b, n_{m_A}]$.
- $C_{i,j}$: The set of common neighbor nodes between n_i and n_j, e.g., C_{b,m_A} is $[n_a, n_{m_B}]$.
- G_i^*: The set of monitored neighbor nodes of n_i, e.g., G_b^* is $[n_{m_A}, n_{m_B}]$.

- $pkt[type, fwd, rec, seq]$, δ, s, s_a, s_h, $AT[sid, rid, op, t_{begin}, t_{end}]$, nid, $ST[nid, s, t_{cur}]$, $MT[nid, c_{mis}, p]$, t_{cur}, c_{mis}, p, sid, rid, op, t_{begin}, t_{end}: Defined before.

Operations:

```
◇ n_i overhears a Data packet pkt[Data, x, y, seq]:
  if n_x ∈ S_i and n_y ∈ G_i^*
      for n_z ∈ C_{i,y} and n_z ∈ F_i
          if ST_i[z].s == s_h /* Forwardee node in harvest state */
              flag_slyA = true; vim = z; src = x; tget_slyA = y;
  if flag_slyA == true and MT_i[tget_slyA].p <= rand[0, 1]
      /* n_i performs the SlyDog on n_{tget_slyA} */
      Broadcast bogus harvest State packet;
      Monitor forwarding behavior of n_{tget_slyA};

◇ n_i performs the SlyDog on n_{tget_slyA}: flag_slyA = true.
  ▷: n_i overhears a Data packet pkt[Data, tget_slyA, vim, seq].
      if ST_i[vim].s == s_h and n_vim ∈ C_{i,tget_slyA}

          MT_i[tget_slyA].p += δ; MT_i[tget_slyA].c_mis += 1;
  ▷: n_i overhears a Wait packet pkt[Wait, tget_slyA, src, seq].
      if ST_i[vim].s == s_h and n_vim ∈ C_{i,tget_slyA}
          Stop the SlyDog on n_{tget_slyA};
      else
          MT_i[tget_slyA].p += δ; MT_i[tget_slyA].c_mis += 1;
  ▷: n_i does not overhear any packet within timeout period.
      MT_i[tget_slyA].p += δ; MT_i[tget_slyA].c_mis += 1;
◇ n_i overhears a Data packet pkt[Data, tget_slyA, rec, seq].
  if rec ∈ F_i and ST_i[rec].s == s_a
      if MT_i[rec].p <= rand[0, 1]
          /* Performs the SlyDog on n_rec */
          Broadcast bogus harvest State packet;
          Monitor forwarding behavior of n_rec;
          flag_slyB = true; tget_slyB = rec;

◇ n_i performs the SlyDog on n_{tget_slyB}: flag_slyB = true.
  ▷: n_i overhears a Data packet pkt[Data, tget_slyB, nrec, seq].
      AT_i[tget_slyB, nrec].op += 1;
      if t_begin == 0
          t_begin = current time;
  ▷: n_i overhears a Wait packet pkt[Wait, tget_slyB, tget_slyA, seq].
      Stop the SlyDog on n_{tget_slyB};
  ▷: n_i does not overhear any packet within timeout period.
      MT_i[tget_slyB].p += δ; MT_i[tget_slyB].c_mis += 1;
```

Fig. 6. The pseudo code of *SlyDog*.

Fig. 4(d). Since n_b is aware of the state of n_{m_B} and monitors the forwarding behavior of its one-hop neighbor nodes, n_{m_A} will behave as a legitimate node and forward a received *Data* packet to n_{m_B}. In Fig. 4(e), n_b overhears the packet sent from n_{m_A} to n_{m_B} and decides whether to perform the SlyDog on n_{m_B} based on the monitor probability of n_{m_B}, p_{m_B}. If n_b decides to perform the SlyDog, it disguises itself as an energy harvesting node and periodically broadcasts a *State* packet piggybacked with its harvest state. In Fig. 4(f), when n_{m_B} overhears a harvest *State* packet from n_b, it is the aforementioned vulnerable case (see Fig. 2(e) or (g)). If n_{m_B} holds the *Data* packet without forwarding, n_b can detect this forwarding misbehavior since n_b stealthily monitors the forwarding operation of n_{m_B}. If n_{m_B} replies a *Wait* packet back to the packet sender (n_{m_A}), it is considered as a legitimate node by n_b. However, this forwarding behavior can be suspected by n_c because it is in active state and can overhear the *Wait* packet. Major operations of the SlyDog are summarized in Fig. 6.

5.2. LazyDog: monitor-based detection

The basic idea of LazyDog is that each node requests its one-hop neighbor nodes to advertise the number of packets forwarded to its two-hop neighbor nodes during a certain period of time. Since each node can simply count and record the number of overheard or received packets, this information can be used as a clue to detect the forwarding misbehavior. For example, n_b can overhear the packet sent from n_{m_B} to n_c, but it cannot make sure whether the packet has been successfully

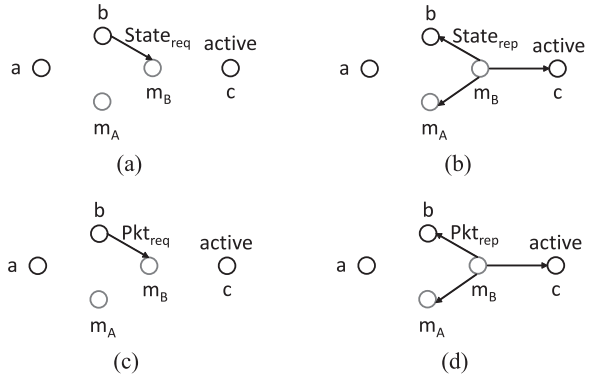


Fig. 7. The proposed *LazyDog* detection operations.

received by n_c because n_b is not aware of the state of n_c as shown in Fig. 2(e).

5.2.1. Detection operations

Under the basic operation of SlyDog, we present detection operations of LazyDog with a set of snapshots of networks in Fig. 7. Since n_b is not aware of the state of n_c, it requests n_{m_B} to broadcast the states of one-hop neighbor nodes by sending a $State_{req}$ packet as shown in Fig. 7(a). Then n_b is aware of the active state of n_c after n_{m_B} broadcasts the $State_{rep}$ packet, containing the states of one-hop neighbor nodes as shown in Fig. 7(b). Then n_b requests n_{m_B} to advertise the number of packets forwarded to n_c during a time period, $AT_b[m_B, c].(t_{begin}, t_{end})$, by sending a Pkt_{req} packet as shown in Fig. 7(c). If n_{m_B} refuses to advertise within a timeout period, n_b suspects n_{m_B} of the forwarding misbehaviors and increments $MT_b[m_B].c_{mis}$ by $AT_b[m_B, c].op$. However, if n_{m_B} advertises, this can be overheard by both n_b and n_c. Then both n_b and n_c compare the received advertisements with their number of overheard packets and number of received packets from n_{m_B} during the time period, respectively as shown in Fig. 7(d). If the difference is greater than a predefined threshold value Det_{th} , either n_b or n_c can detect the forwarding misbehavior of n_{m_B}. Major operations of the LazyDog are summarized in Fig. 8.

6. Analysis of the proposed countermeasure

In this section, we analyze the EYES in terms of detection rate of malicious node in the EHNets. Based on the set of adversarial scenarios observed in Section 4, a forwarding misbehavior of malicious node can often be detected in the following three cases: (i) An adjacent node overhears a *Data* packet which is forwarded to a node in harvest state; (ii) An adjacent node does not overhear the *Data* packet forwarded before the timeout expires; and (iii) An adjacent node receives an inconsistent summary of *Data* packet forwarded. In the cases, two malicious nodes consecutively located along the forwarding path between the packet sender and the sink are assumed to monitor the network traffic and collude selective forwarding attacks. A neighbor node located within the communication range of the packet sender and two consecutively located malicious nodes performs the EYES to detect a potential forwarding misbehavior of two malicious nodes as shown in Fig. 9. In this analysis, we also assume that a packet loss is primarily caused by a bad channel quality. The packet loss rate is denoted as C_{ls}. For the sake of clarity in demonstrating the analysis process, we refer Figs. 4 and 7 to analyze the detection rate of SlyDog and LazyDog, respectively.

6.1. Detection rate of SlyDog

We conduct the analysis of SlyDog based on the scenarios used for detection operations in Fig. 4. In Fig. 4(a)–(c), n_a is relaying a *Data*

Notations:

- RP_j: The set of the number of received *Data* packets of n_j, e.g., $RP_c[m_B]$ is the number of received *Data* packets from n_{m_B}.
- FP_i: The set of the number of forwarded *Data* packets of n_i, e.g., $FP_{m_B}[c]$ is the number of forwarded *Data* packet to n_c from n_{m_B}.
- DN_i: The set of one-hop neighbor nodes of n_i.
- THN_i: The set of two-hop neighbor nodes of n_i.
- SL_i: The set of current state of one-hop neighbor nodes of n_i.
- LD_i: The set of currently active two-hop neighbor nodes of n_i.
- $t_{out\ lazy}$: The *LazyDog* detection window interval.
- $State_{req}$, $State_{rep}$, Pkt_{req}, Pkt_{rep} and Det_{th}: Defined before.

Operations:

◇ n_i starts the *LazyDog* detection: t_{out_lazy} expires.
  Randomly selects n_t, $n_t \in DN_i$ **and** $ST_i[t].s == s_a$;
  Forwards the state request packet $pkt[State_{req}, i, t, seq]$ to n_t;
◇ n_i overhears the state reply packet $pkt[State_{rep}, t, SL_t, seq]$.
  for $n_x \in THN_i$
    if $SL_t[x] == s_a$
      $LD_i = LD_i \cup n_x$;
      $flag_{lazy}$ = **true**; /* *Performs the LazyDog on* n_t */
  if $flag_{lazy} ==$ **true**
    Forward the packet number request packet $pkt[Pkt_{req}, i, t, seq]$ to n_t;
◇ n_i performs the $LazyDog$ on n_t: $flag_{lazy} ==$ **true**.
  ▷: n_i doesn't overhear the reply packet $pkt[Pkt_{rep}, t, FP_t, seq]$
    for $n_x \in LD_i$
      $MT_i[t].c_{mis}$ += $AT_i[t, x].op$; $AT_i[t, x].op = 0$;
  ▷: n_i overhears the reply packet $pkt[Pkt_{rep}, t, FP_t, seq]$.
    for $n_x \in LD_i$
      if $(|FP_t[x] - AT_i[t, x].op|) \leq Det_{th}$, $FP_t[x] \in FP_t$
        $AT_i[t, x].op = 0$; /* n_t *behaves well on* n_x */
      else /* n_i *detects the forwarding misbehavior of* n_t */
        $MT_i[t].c_{mis}$ += $(|FP_t[x] - AT_i[t, x].op|)$;
◇ n_x overhears the reply packet $pkt[Pkt_{rep}, t, FP_t, seq]$, $n_x \in THN_i$:
  if $(|RP_x[t] - FP_t[x]|) \leq Det_{th}$, $FP_t[x] \in FP_t$
    Discard the packet $pkt[Pkt_{rep}, t, FP_t, seq]$;
  else
    $MT_x[t].c_{mis}$ += $(|RP_x[t] - FP_t[x]|)$;

Fig. 8. The pseudo code of *LazyDog*.

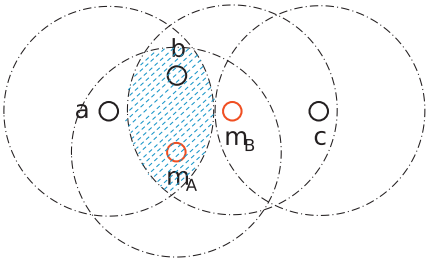


Fig. 9. A common neighbor node (e.g., n_b) of n_a, n_{m_A}, and n_{m_B} is located in the dashed area and performs the EYES to detect a potential forwarding misbehavior of malicious nodes.

packet (pkt[*Data, a,* m_A]) to a malicious node, n_{m_A}. Then n_{m_A} colludes with another malicious node, n_{m_B}, which is in harvest state, to drop the packet on purpose without being detected. There are six cases for n_b to detect any forwarding misbehavior:

(1) n_b misses the pkt[*Data, a,* m_A] and stays in active state; n_{m_A} replies the pkt[*Wait,* m_A, *a*] to n_a; and n_b misses the pkt[*Wait,* m_A, *a*] ⇒ normal
(2) n_b misses the pkt[*Data, a,* m_A] and stays in active state; n_{m_A} replies the pkt[*Wait,* m_A, *a*] to n_a; and n_b overhears the pkt[*Wait,* m_A, *a*] ⇒ normal
(3) n_b overhears the pkt[*Data, a,* m_A], performs the SlyDog on n_{m_A}, and changes to harvest state; n_{m_A} forwards the pkt[*Data,* m_A, m_B] to n_{m_B}; and n_b misses the pkt[*Data,* m_A, m_B] ⇒ detected as a packet loss
(4) n_b overhears the pkt[*Data, a,* m_A], performs the SlyDog on n_{m_A}, and

changes to harvest state; n_{m_A} forwards the *pkt*[*Data, m_A, m_B*] to n_{m_B}; and n_b overhears the *pkt*[*Data, m_A, m_B*] ⇒ detected by the SlyDog

(5) n_b overhears the *pkt*[*Data, a, m_A*] but does not perform the SlyDog on n_{m_A}, and stays in active state; n_{m_A} replies the *pkt*[*Wait, m_A, a*] to n_a; and n_b misses the *pkt*[*Wait, m_A, a*] ⇒ detected as a packet loss

(6) n_b overhears the *pkt*[*Data, a, m_A*] but does not perform the SlyDog on n_{m_A}, and stays in active state; n_{m_A} replies the *pkt*[*Wait, m_A, a*] to n_a; and n_b overhears the *pkt*[*Wait, m_A, a*] ⇒ normal

In the SlyDog, cases (1), (2) and (6) are the forwarding operations of legitimate node. However, cases (3)–(5) show the forwarding misbehaviors and the malicious node can be detected.

The probability of forwarding behavior shown in the cases (3)–(5) can be expressed as P_3, P_4, and P_5 below, respectively.

$$P_3 = (1 - C_{ls})\cdot MT\,[m_A].\ p\cdot C_{ls}. \tag{1}$$

$$P_4 = (1 - C_{ls})\cdot MT\,[m_A].\ p\cdot(1 - C_{ls}). \tag{2}$$

$$P_5 = (1 - C_{ls})\cdot(1 - MT\,[m_A].\ p)\cdot C_{ls}. \tag{3}$$

The probability that there is at least one active adjacent node (i.e., n_b) of n_a, n_{m_A}, and n_{m_B} can be expressed as,

$$P_G = 1 - (1 - P_a)^r. \tag{4}$$

Here, r is the total number of adjacent nodes of n_a, n_{m_A}, and n_{m_B}. P_a is the probability of node staying in active state. Thus, the detection rate for a malicious node n_{m_A} can be expressed as,

$$\begin{aligned} P_{dt_{sly_A}} &= P_G\cdot(P_3 + P_4 + P_5) \\ &= P_G\cdot(1 - C_{ls})\cdot(MT\,[m_A].\ p\cdot(1 - C_{ls}) + C_{ls}). \end{aligned} \tag{5}$$

We also consider the cases shown in Fig. 4(d)–(f), where n_{m_B} is in active state. There are additional nine cases for n_b to detect any forwarding misbehavior:

(7) n_b misses the *pkt*[*Data, a, m_A*] and stays in active state; n_{m_A} forwards the *pkt*[*Data, m_A, m_B*] to n_{m_B}; n_b misses the *pkt*[*Data, m_A, m_B*] and stays in active state; n_{m_B} forwards the *pkt*[*Data, m_B, c*] to n_c; and n_b misses the *pkt*[*Data, m_B, c*] ⇒ normal

(8) n_b misses the *pkt*[*Data, a, m_A*] and stays in active state; n_{m_A} forwards the *pkt*[*Data, m_A, m_B*] to n_{m_B}; n_b misses the *pkt*[*Data, m_A, m_B*] and stays in active state; n_{m_B} forwards the *pkt*[*Data, m_B, c*] to n_c; and n_b overhears the *pkt*[*Data, m_B, c*] ⇒ normal

(9) n_b misses the *pkt*[*Data, a, m_A*] and stays in active state; n_{m_A} forwards the *pkt*[*Data, m_A, m_B*] to n_{m_B}; n_b overhears the *pkt*[*Data, m_A, m_B*] and performs the SlyDog on n_{m_B}, changes to harvest state; and n_{m_B} holds the packet ⇒ detected by the SlyDog

(10) n_b misses the *pkt*[*Data, a, m_A*] and stays in active state; n_{m_A} forwards the *pkt*[*Data, m_A, m_B*] to n_{m_B}; n_b overhears the *pkt*[*Data, m_A, m_B*] but does not perform the SlyDog on n_{m_B} and stays in active state; n_{m_B} forwards the *pkt*[*Data, m_B, c*] to n_c; and n_b misses the *pkt*[*Data, m_B, c*] ⇒ detected as a packet loss

(11) n_b misses the *pkt*[*Data, a, m_A*] and stays in active state; n_{m_A} forwards the *pkt*[*Data, m_A, m_B*] to n_{m_B}; n_b overhears the *pkt*[*Data, m_A, m_B*] but does not perform the SlyDog on n_{m_B} and stays in active state; n_{m_B} forwards the *pkt*[*Data, m_B, c*] to n_c; and n_b overhears the *pkt*[*Data, m_B, c*] ⇒ normal

(12) n_b overhears the *pkt*[*Data, a, m_A*] and stays in active state; n_{m_A} forwards the *pkt*[*Data, m_A, m_B*] to n_{m_B}; and n_b misses the *pkt*[*Data, m_A, m_B*] ⇒ detected as a packet loss

(13) n_b overhears the *pkt*[*Data, a, m_A*] and stays in active state; n_{m_A} forwards the *pkt*[*Data, m_A, m_B*] to n_{m_B}; n_b overhears the *pkt*[*Data, m_A, m_B*], performs the SlyDog on n_{m_B}, and changes to harvest state; and n_{m_B} holds the packet ⇒ detected by the SlyDog

(14) n_b overhears the *pkt*[*Data, a, m_A*] and stays in active state; n_{m_A} forwards the *pkt*[*Data, m_A, m_B*] to n_{m_B}; n_b overhears the *pkt*[*Data, m_A, m_B*] but does not perform the SlyDog on n_{m_B} and stays in active state; n_{m_B} forwards the *pkt*[*Data, m_B, c*] to n_c; and n_b misses the *pkt*[*Data, m_B, c*] ⇒ detected as a packet loss

(15) n_b overhears the *pkt*[*Data, a, m_A*] and stays in active state; n_{m_A} forwards the *pkt*[*Data, m_A, m_B*] to n_{m_B}; n_b overhears the *pkt*[*Data, m_A, m_B*] but does not perform the SlyDog on n_{m_B} and stays in active state; n_{m_B} forwards the *pkt*[*Data, m_B, c*] to n_c; and n_b overhears the *pkt*[*Data, m_B, c*] ⇒ normal

The cases (9), (10), (12), (13), and (14) show the forwarding misbehaviors and the malicious node can be detected. The probability of forwarding behavior shown in the cases (9), (10), (12), (13), and (14) can be expressed as P_9, P_{10}, P_{12}, P_{13}, and P_{14} below, respectively.

$$P_9 = C_{ls}\cdot(1 - C_{ls})\cdot MT\,[m_B].\ p. \tag{6}$$

$$P_{10} = C_{ls}\cdot(1 - C_{ls})\cdot(1 - MT\,[m_B].\ p)\cdot C_{ls}. \tag{7}$$

$$P_{12} = (1 - C_{ls})\cdot C_{ls}. \tag{8}$$

$$P_{13} = (1 - C_{ls})\cdot(1 - C_{ls})\cdot MT\,[m_B].\ p. \tag{9}$$

$$P_{14} = (1 - C_{ls})\cdot(1 - C_{ls})\cdot(1 - MT\,[m_B].\ p)\cdot C_{ls}. \tag{10}$$

Thus, the detection rate for a malicious node n_{m_B} can be expressed as,

$$\begin{aligned} P_{dt_{sly_B}} &= P_G\cdot(P_9 + P_{10} + P_{12} + P_{13} + P_{14}) \\ &= P_G\cdot(1 - C_{ls})\cdot(2C_{ls} + (1 - C_{ls})\cdot MT\,[m_B].\ p). \end{aligned} \tag{11}$$

Finally, the detection rate of the SlyDog can be expressed as,

$$P_{detect,SlyDog} = P_{dt_{sly_A}} + P_{dt_{sly_B}}. \tag{12}$$

6.2. Detection rate of LazyDog

We conduct the analysis of LazyDog based on the scenarios used for detection operations in Fig. 7. An adjacent node (i.e., n_b or n_c) of malicious node (i.e., n_{m_B}) compares its number of overheard or received packets counted within a time period (t_{begin}, t_{end}) with the number of announcements counted by the malicious node. If the difference is greater than a predefined threshold value Det_{th}, the forwarding misbehaviors of malicious node are detected. Within the time period (t_{begin}, t_{end}), we assume that F_{m_B} number of packets are forwarded from n_{m_B} to n_c. F_{m_B} should be greater than the predefined threshold value Det_{th}. Otherwise, the detection rate is zero. We denote that O_b and R_c are the number of packets overheard by n_b sent from n_{m_B} to n_c and the number of packets received by n_c sent from n_{m_B}, respectively. Also D_c is the number of dropped packets forwarded to n_c in harvest state. Thus, the maximum number of packets received by n_c is F_{m_B} - D_c, which is represented as M_{rec,n_c}.

A malicious node n_{m_B} has two options to announce its counter value of forwarded packets to n_c: (i) M_{rec,n_c}, to match with the counter value of the number of received packets at n_c; or (ii) F_{m_B}, to match with the counter value of the number of overheard packets at n_b. We assume that n_{m_B} will randomly choose one of options to match with the counter value at n_c or n_b.

n_b can be one of five states based on the counter value announced by n_{m_B}:

(1) n_{m_B} announces F_{m_B}:
  (a) (F_{m_B} - Det_{th}) ≤ O_b ≤ F_{m_B} ⇒ normal
  (b) 0 ≤ O_b < (F_{m_B} - Det_{th}) ⇒ uncertain

(2) n_{m_B} announces M_{rec,n_c}:
  (a) M_{rec,n_c} < O_b ≤ F_{m_B} ⇒ detected by the LazyDog
  (b) (M_{rec,n_c} - Det_{th}) ≤ O_b ≤ M_{rec,n_c} ⇒ good
  (c) 0 ≤ O_b < (M_{rec,n_c} - Det_{th}) ⇒ uncertain

The probability that n_b overhears at least M_{rec,n_c} + 1 packets (case *2.a*) can be expressed as,

$$P_{2.a} = \sum_{i=M_{rec,n_c}+1}^{F_{m_B}} \binom{F_{m_B}}{i} (1 - C_{ls})^i \cdot (C_{ls})^{F_{m_B}-i}. \quad (13)$$

However, n_c can be one of four states based on the counter announced by n_{m_B}:

(3) n_{m_B} announces F_{m_B}:
  (a) $(F_{m_B}$ - $Det_{th}) \leq R_c \leq F_{m_B} \Rightarrow$ normal
  (b) $0 \leq R_c < (F_{m_B}$ - $Det_{th}) \Rightarrow$ detected by the LazyDog
(4) n_{m_B} announces M_{rec,n_c}:
  (a) $(M_{rec,n_c}$ - $Det_{th}) \leq R_c \leq M_{rec,n_c} \Rightarrow$ normal
  (b) $0 \leq R_c < (M_{rec,n_c}$ - $Det_{th}) \Rightarrow$ uncertain

The probability that n_c receives the less than F_{m_B} - Det_{th} number of packets (case *3.b*) can be expressed as,

$$P_{3.b} = \sum_{i=0}^{\eta=(F_{m_B}-Det_{th}-1)} \binom{\eta}{i} (1 - C_{ls})^i \cdot (C_{ls})^{\eta-i}. \quad (14)$$

Thus, the detection rate of the LazyDog can be expressed as,

$$P_{detect.LazyDog} = \frac{P_{2.a} + P_{3.b}}{2}. \quad (15)$$

In Fig. 10, we show the impact of monitor probability and the number of dropped packets on the detection rate of the SlyDog and LazyDog, respectively. In Fig. 10(a), the detection rate of SlyDog increases linearly as the monitor probability increases, because the node has more chances to perform the SlyDog on the suspected node and detects more forwarding misbehaviors with larger monitor probability. In Fig. 10(b), the number of dropped packets does not affect the detection rate of LazyDog much. This is because it is hard to detect whether the packets are dropped by malicious node or lost during the transmission due to a bad channel quality.

In summary, we analyze the proposed approach by comprehensively investigating the forwarding operation cases for potential forwarding misbehavior and measure the detection rates. Since the proposed analysis is to estimate the performance trend, we conduct detail performance evaluation through extensive simulation experiments in the following section.

7. Performance evaluation

7.1. Simulation testbed

We conduct extensive simulation experiments using the OMNeT++ [23] to evaluate the performance of proposed approach. A 200 × 200 (m^2) rectangular network area is considered, where 150 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 [42], and the channel error rate is set to 5%. A single node generates data traffic with injection rate 0.33 or 0.66 (pkt/s) and

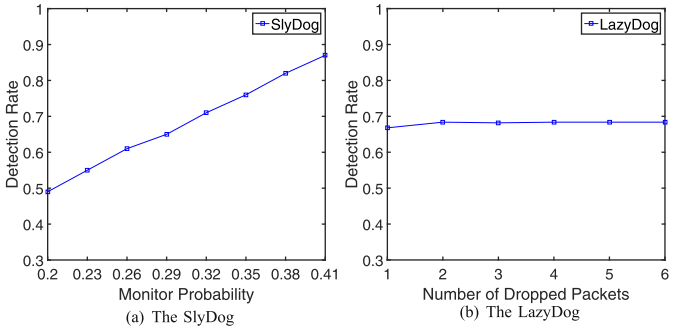


Fig. 10. The detection rates against monitor probability and the number of dropped packets. Here, $r = 3$, $P_a = 67\%$, $C_{ls} = 5\%$, $Det_{th} = 2$, and $F_{m_B} = 10$.

Table 2
Simulation parameters.

Parameter	Value
Network size	200 × 200 m^2
Number of nodes	150
Number of malicious nodes along the route	1 or 2
Channel error rate	5%
Radio data rate	250 Kbps
Packet injection rate	0.33 or 0.66 pkt/s
Packet size	1 KB
Packet drop rate of HCD and Watchdog	30%
Radio range	12.3 m
Radio model	CC2420
Simulation time	1000 s
Active time period	50–80 s
Harvest time period	15–40 s

the data packet size is 1 KB. The inter-arrival time of traffic is assumed to be exponentially distributed. The total simulation time is 1000 s. The periods of active and harvest states vary between 50–80 (s) and 15–40 (s), respectively. In the proposed approach, two malicious nodes are consecutively located along the forwarding path between the packet sender and the sink, in which malicious nodes are assumed to monitor network traffic and local network condition, and perform selective forwarding attacks cooperatively.

For performance comparison, we compare our proposed schemes, *EYES*, with a hop-by-hop cooperative detection scheme, called *HCD* [22], which is the first countermeasure to selective forwarding attack in EHNets. The EYES is also compared with the well-known scheme, *Watchdog* [8]. We adjust and implement the Watchdog in the EHNets, where a single and two malicious nodes are consecutively located, denoted as *1-M* and *2-M*, respectively. Here, a malicious node is set to randomly drop received packets with 30% dropping rate in the HCD and Watchdog. The simulation parameters are summarized in Table 2.

7.2. Simulation results

We measure the performance in terms of detection rate, detection latency, packet delivery ratio (PDR), energy consumption, and monitor probability by changing key simulation parameters, including energy harvest time (t_h), the number of malicious node, and monitor probability (δ).

7.2.1. Detection rate

We first measure the detection rate by changing harvest time (t_h), packet injection rate (r_{pkt}) and δ in Fig. 11(a) and (b). In Fig. 11(a), as t_h increases with $r_{pkt} = 0.33$ (pkt/sec), the detection rates of both EYES and HCD increase while that of the Watchdog decreases. In the Watchdog, each node passively changes its state between active and harvest and monitors the forwarding behavior only during active state. As t_h increases, more nodes stay in harvest state for longer time period and the detection rate decreases even though malicious nodes can drop packets with 30% dropping rate. Thus, lower detection rate is observed with two malicious nodes located consecutively in the forwarding path, because more packets are dropped by two malicious nodes and these forwarding misbehaviors cannot be detected. Both EYES and HCD show higher detection rate than that of the Watchdog in high t_h. This is because the SlyDog can actively disguise each node as an energy harvesting node and monitor any forwarding behavior of its adjacent nodes, or exchange the trace information with its adjacent nodes and detect more forwarding misbehaviors. In particular, the EYES shows higher detection rate than that of the HCD because prior uncertain packet forwarding operations can be verified by the LazyDog, and more forwarding misbehaviors can be detected. In the HCD, the detection rate increases slowly compared to that of the EYES, because the forwarding probability of the malicious node is reduced whenever a

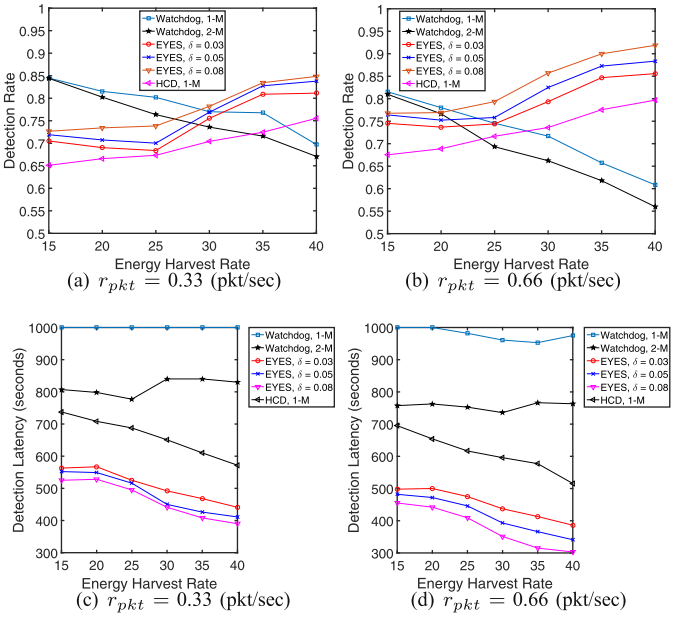


Fig. 11. The performance impact against energy harvest rate, number of malicious nodes, packet injection rate, and δ.

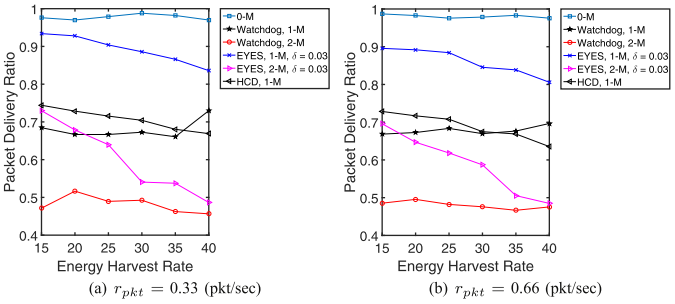


Fig. 12. The packet delivery ratio against energy harvest rate, packet injection rate, and δ.

forwarding misbehavior is detected. Since the malicious node seldom receives the packet, its forwarding misbehaviors can be significantly reduced. In the EYES, the detection rate increases as δ increases. This is because the monitor probability (p) increases quickly with larger δ and thus, nodes have more chances to disguise themselves as an energy harvesting node and detect more forwarding misbehaviors. In Fig. 11(b), overall detection rates of the EYES and HCD increase with r_{pkt} = 0.66 (pkt/s), because more packets are forwarded to malicious node with larger r_{pkt} and then more packets are dropped by malicious node but its forwarding misbehaviors can be detected by the EYES and HCD. The EYES shows the best performance as t_h increases compared to that of the HCD and Watchdog.

7.2.2. Detection latency

Second, the detection latency is measured by changing t_h, r_{pkt}, and δ in Fig. 11(c) and (d). As t_h increases, more nodes are in harvest state and more vulnerable cases are witnessed as observed in the adversarial scenarios in Fig. 2(c), (e), (f), (g) and (h). In Fig. 11(c), the EYES achieves the lowest detection latency compared to that of the HCD and Watchdog. This is because adjacent nodes of malicious nodes can disguise themselves as an energy harvesting node, counterfeit vulnerable cases, and finally detect more forwarding misbehaviors. With higher δ, the detection latency decreases because nodes can frequently disguise themselves and monitor any forwarding operation. The LazyDog also helps to reduce the detection latency by detecting the uncertain forwarding operation of malicious nodes. The EYES can also quickly isolate malicious nodes from the network. Unlike the EYES, the HCD shows higher detection latency for entire t_h. In the HCD, each packet sender can detect the forwarding misbehavior of suspected node only after receiving a *Mode*[5] packet broadcasted from its adjacent nodes. Upon receiving the *Mode* packet, the sender updates the states of its neighbor nodes and searches whether there was any forwarding operation while any forwardee node was in harvest state. The Watchdog shows the highest detection latency because nodes can only detect the forwarding misbehavior in active state. In Fig. 11(d), overall detection latencies decrease with higher packet injection rate 0.66 (pkt/sec),. However, the EYES achieves the best performance and its detection latency decreases quickly compared to that of the HCD and Watchdog.

7.2.3. Packet delivery ratio

Third, we measure the packet delivery ratio (PDR) by varying t_h, r_{pkt}, and δ in Fig. 12. In this paper, we deploy a no malicious node case under different r_{pkt}, denoted as *0-M*, to see the upper bound of average PDR (about 98% or more). In *0-M*, every node cooperatively forwards the received packet to the sink. The Watchdog is not sensitive to t_h and packet injection rate but to packet dropping rate (i.e., 30%), and its PDR is fluctuating between 68% and 47% with a single (*1-M*) and two (*2-M*) malicious nodes, respectively. This is because the malicious node can stay in active state for an extended period but it only randomly drops with 30% packet drop rate. In Fig. 12(a), the EYES with one (*1-M*) and two (*2-M*) malicious nodes shows higher and lower PDR than that of the HCD with a single malicious node, respectively. This is because two malicious nodes located consecutively can collude together and intentionally drop more packets without being detected. Unlike the Watchdog, the HCD can reduce the number of forwarding misbehaviors by decreasing the probability of malicious node being chosen as a forwarder node. Thus, the HCD shows higher PDR than that of the Watchdog with a single malicious node. The HCD also shows lower PDR than that of the EYES with a single malicious node. This is because the malicious node only performs the undetected forwarding operation in the EYES, while the malicious node in the HCD randomly drops the received packet with 30% packet drop rate. In Fig. 12(b), overall PDRs decrease with higher packet injection rate (i.e., 0.66 (pkt/s)) because more packets are dropped by malicious nodes due to more number of generated packets in the network.

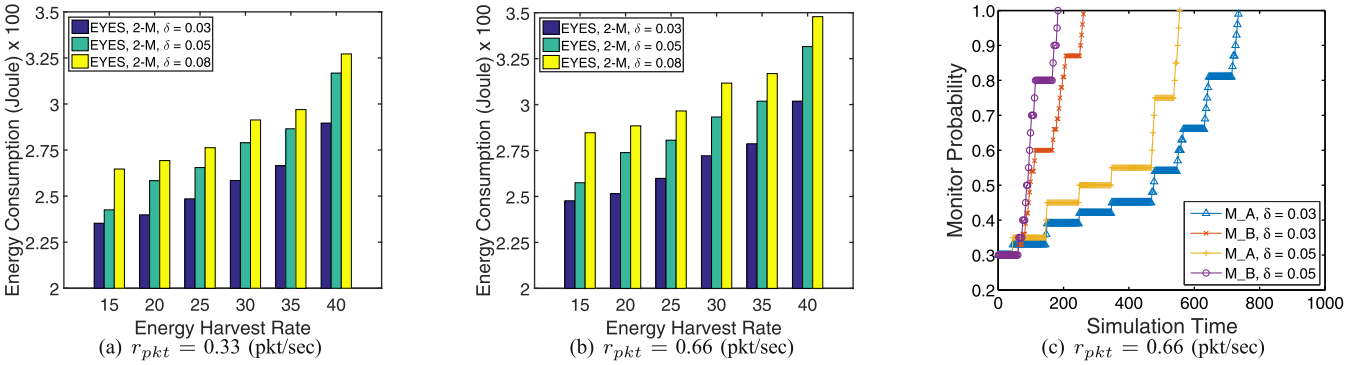
7.2.4. Energy consumption

Fourth, we measure energy consumption in terms of the number of overheard forwarding misbehaviors of malicious nodes [43] in Fig. 13(a) and (b). An overheard forwarding misbehavior occurs when a malicious node forwards a packet to a legitimate node which is in harvest state, resulting in packet loss. As t_h increases, malicious nodes have more chances to forward packets to the nodes in harvest state and reveal forwarding misbehaviors frequently in Fig. 13(a). However, this forwarding misbehavior can be detected by the SlyDog and then the energy consumption of detection increases. With larger δ, nodes have more chances to disguise themselves as an energy harvesting node, monitor any forwarding misbehavior, and consume more energy. In Fig. 13(b), more energy consumption is observed with higher packet injection rate, because more packets are dropped by malicious nodes but these forwarding misbehaviors can be detected, which consumes more energy. Thus, the EYES can efficiently utilize the harvested energy to monitor and detect the forwarding misbehaviors of malicious nodes.

7.2.5. Monitor probability

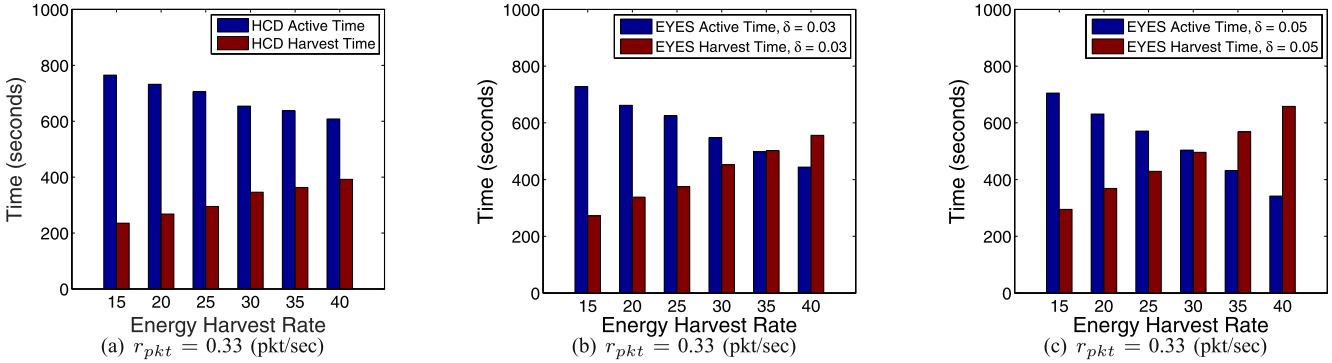
Fifth, we observe the changes of monitor probability (p) in the presence of two malicious nodes with different weights (δ = 0.03 or 0.05) over the simulation period in the EYES as shown in Fig. 13(c).

[5] In [22], a node broadcasts a *Mode* packet whenever it changes its state. This is similar to a *State* packet in this paper.



(a) $r_{pkt} = 0.33$ (pkt/sec) (b) $r_{pkt} = 0.66$ (pkt/sec) (c) $r_{pkt} = 0.66$ (pkt/sec)

Fig. 13. The energy consumption and monitor probability against energy harvest rate, packet injection rate, and δ.



(a) $r_{pkt} = 0.33$ (pkt/sec) (b) $r_{pkt} = 0.33$ (pkt/sec) (c) $r_{pkt} = 0.33$ (pkt/sec)

Fig. 14. The performance of total active and harvest time periods against energy harvest rate and δ.

Whenever a node detects a forwarding misbehavior, it increases the monitor probability (p) of suspected node by δ. With larger δ, malicious nodes are monitored more often and thus, there are more chances of their forwarding misbehaviors detected, leading to a quick isolation from the network. For example, the monitor probabilities of malicious nodes n_{m_A} and n_{m_B} (see Fig. 4(a)) reach to 1.0 at 580 and 180 s with $\delta = 0.05$, respectively. This indicates that any forwarding operation of two malicious nodes is suspected and monitored. Note that the monitor probability of n_{m_B} reaches to 1.0 earlier than that of n_{m_A} with different δ. Since the prior packet sender of n_{m_B} is n_{m_A}, n_{m_B} tends to perform more forwarding misbehaviors for possible collusion.

7.2.6. Impact of harvest time and δ

Finally, we measure the total time periods of nodes staying in active and harvest states in the HCD and EYES by changing t_h and δ over the simulation period as shown in Fig. 14(a), (b), and (c). In the HCD, since each node does not perform any monitoring operation during the harvest state, total harvest time period increases linearly as t_h increases in Fig. 14(a). In the EYES, however, total harvest time period in Fig. 14(b) increases quickly compared to that of the HCD in Fig. 14(a). Since nodes actively disguise themselves as an energy harvesting node and pretend not to overhear, longer harvest time period is observed in the EYES. In Fig. 14(b) and (c), more harvest time period is observed with larger δ (i.e., 0.05). This is because larger δ increases the monitor probability quickly and thus, more nodes frequently disguise themselves as an energy harvesting node.

8. Discussion

In this section, we first investigate the proposed approach by considering its features, constraints, and explore possible extensions. Then we also investigate the applicability of the proposed approach to other attacks in EHNets.

8.1. Features, constraints, and potential enhancements

We discuss the EYES in terms of its features, constraints, and possible extensions for improvement. The EYES is designed with three desirable features. First, each node can actively disguise itself as an energy harvesting node to stealthily monitor the forwarding operation of its adjacent nodes. This active detection technique can detect more forwarding misbehaviors and thus, the malicious node can quickly be excluded from participating the forwarding operation in the network. Second, the monitor probability indicates how actively a node monitors the forwarding operation of suspected node, and it increases when a forwarding misbehavior is detected. This incremental monitor probability can significantly increase the detection rate and reduce the detection latency simultaneously. This is because nodes have more chances to disguise themselves as an energy harvesting node and detect more forwarding misbehaviors. Third, since the node cannot make sure whether the forwarded packet from its adjacent nodes has been successfully received by its two-hop neighbor nodes, each node periodically requests its adjacent nodes to broadcast the number of forwarded packets to its two-hop neighbor nodes. Thereby, any prior uncertain forwarding operation can be verified.

In the EYES, there are a few constraints that need to be further considered. First, the major detection operation of the SlyDog is based on an implicit monitoring technique. In case of a sparse network, if n_a only has one forwarding node n_{m_A} which colludes with n_{m_B}, it would be hard to detect any forwarding misbehavior of n_{m_A} in Fig. 4. This is because there are no other neighbor node except for n_a to observe the forwarding misbehavior of n_{m_A}. Second, due to the nature of charge-and-spend energy harvesting policy, malicious nodes still have a chance to perform forwarding misbehaviors without being detected, if their neighbor nodes have to switch to harvest state.

To see the full potential of the EYE, we plan to investigate the followings for extensions.

Table 3
The comparison[a] of detection strategies of forwarding misbehavior.

Approach	Collusive attack	Computation overhead	Communication overhead	Detection latency	Punishment	Architecture
Watchdog [8]	N	Low	N	N	N	Stand-alone
CHEMAS [12]	N	Medium	High	Low	N	Centralized
CAD [14]	N	Medium	Medium	Medium	N	Centralized
SCM [15]	N	Low	N	Medium	N	Stand-alone
EAACK [16]	N	Medium	High	Medium	N	Centralized
FADE [17]	Y	Medium	High	Low	N	Centralized
CRS [18]	Y	High	Medium	Medium	Y	Distributed
CBDS [19]	Y	Medium	Medium	High	N	Distributed
SNBDS [20]	Y	Medium	Medium	High	N	Distributed
SCAD [21]	Y	Medium	Medium	Low	N	Centralized
HCD [22]	N	Medium	Low	High	Y	Distributed
CAM [27]	N	Low	N	N	Y	Stand-alone
SlyDog	Y	Low	N	N	Y	Stand-alone
LazyDog	N	Low	Low	Medium	Y	Distributed

[a] In this paper, we compare the proposed countermeasure with prior detection strategies of forwarding misbehavior in terms of six aspects: (i) *Collusive attack*: Against to two or more cooperative malicious nodes; (ii) *Computation overhead*: Extra computation required for detection; (iii) *Communication overhead*: Extra packets generated for detection; (iv) *Detection latency*: Time delay to identify forwarding misbehaviors; (v) *Punishment*: Penalty of forwarding misbehaviors; and (vi) *Architecture* [21]: *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.

8.1.1. Dummy packets

In the SlyDog, each node actively pretends not to overhear the on-going communications of its adjacent nodes, but in fact monitors the forwarding activities to detect a lurking deep malicious node. The monitor probability of the suspected node increases if the forwarding misbehavior is detected, and the suspected node has more chances to be monitored later. Thus, we plan to extend the SlyDog for the packet sender to intentionally distribute dummy packets [44] to the suspected node when there is no on-going communication. Then we can lure the suspected node to reveal its forwarding misbehavior.

8.1.2. Bypass retransmission

We also plan to deploy a bypass retransmission technique in the SlyDog to quickly recover unexpected packet losses caused by the forwarding misbehavior. For example, if a node detects a forwarding misbehavior or packet drop from a suspected node, it retransmits its cached data packet by selecting an alternative forwarding path [45,46] to prevent the suspected node from participating the forwarding operation.

8.2. Applicability to other attacks

We also investigate the proposed approach whether it can be applicable to two other well-known attacks: (i) limited transmission power attack; and (ii) receiver collisions attack [8].

8.2.1. Limited transmission power attack

A malicious node may drop a packet on purpose by transmitting it with reduced transmission power to exclude a legitimate next-hop node from its communication range. This attack is similar to a selective forwarding attack and it can be detected by the EYES. For example, suppose n_{m_A} forwards a data packet to n_{m_B} in Fig. 4(e). n_b overhears this packet transmission, chooses not to perform the SlyDog on n_{m_B}, and stays in active state. Then n_{m_B} may forward the packet by carefully reducing the communication range that does not reach to n_c but the packet can be overheard by n_b. In the LazyDog, since n_b periodically requests its adjacent node (i.e., n_{m_B}) to advertise the number of packets forwarded to its two-hop neighbor nodes (i.e., n_c), this forwarding misbehavior can be detected by either n_b or n_c.

8.2.2. Receiver collisions attack

A malicious node may create a packet collision at the receiver on purpose by simultaneously sending any packet with the packet sender. It is not trivial to avoid this receiver collisions attack but this attack can be detected by the EYES. For example, suppose n_a sends a *Data* packet to n_{m_A} and n_{m_B} also simultaneously sends any packet to n_{m_A} in Fig. 4(d). Then n_{m_A} fails to receive the *Data* packet due to the collision. In the EYES, after n_b overhears the packet transmission from n_a to n_{m_A}, n_b will monitor the following forwarding operation of n_{m_A} no matter whether it performs the SlyDog on n_{m_A}. Since the *Data* packet is lost at n_{m_A}, n_b cannot overhear it forwarded from n_{m_A} before its timer expires. Thus, n_b will prosecute the forwarding misbehavior of n_{m_A}.

In summary, we compare the proposed approach with recent detection approaches and summarize their properties in Table 3, extended from [21].

9. Concluding remarks

In this paper, we investigated the forwarding misbehavior and its countermeasure in the realm of EHNets. Under the charge-and-spend harvesting policy, a set of adversarial scenarios is created and analyzed, and its potential vulnerabilities are also identified. We proposed a countermeasure, called *EYES*, to efficiently detect the forwarding misbehaviors of multiple malicious nodes in the EHNets. The EYES is the combination of inducement- and monitor-based approaches to quickly identify the lurking deep malicious nodes and isolate them from the network. Extensive simulation results show that the EYES provides 70–92% detection rate and achieves 23–60% lower detection latency compared to the HCD and Watchdog. The EYES also shows a competitive performance in PDR.

Acknowledgment

This research was supported by Startup grant in the Weisberg Division of Computer Science at Marshall University.

References

[1] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (IoT): a vision, architectural elements, and future directions, Futur. Gener. Comput. Syst. 29 (2013) 1645–1660.
[2] R. Kravets, P. Krishnan, Power management techniques for mobile communication, Proceedings of ACM MOBICOM, (1998), pp. 157–168.
[3] N.A. Pantazis, S.A. Nikolidakis, D.D. Vergados, Energy-efficient routing protocols in wireless sensor networks: A Survey, IEEE Commun. Surv. Tutor. 15 (2) (2013) 551–591.

[4] M. Gorlatova, J. Sarik, G. Grebla, M. Cong, I. Kymissis, G. Zussman, Movers and shakers: kinetic energy harvesting for the internet of things, IEEE J. Sel. Areas Commun. 33 (8) (2015) 1624–1639.
[5] P. Kamalinejad, C. Mahapatra, Z. Sheng, S. Mirabbasi, V. Leung, Y.L. Guan, Wireless energy harvesting for the internet of things, IEEE Commun. Mag. 53 (6) (2015) 102–108.
[6] Y. Wang, Y. Liu, C. Wang, Z. Li, X. Sheng, H. Lee, N. Chang, H. Yang, Storage-less and converter-less photovoltaic energy harvesting with maximum power point tracking for internet of things, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 35 (2) (2015) 173–186.
[7] A.D. Wood, J.A. Stankovic, Denial of service in sensor networks, IEEE Comput. 35 (10) (2002) 54–62.
[8] S. Marti, T.J. Giuli, K. Lai, M. Baker, Mitigating routing misbehavior in mobile ad hoc networks, Proceedings of ACM MOBICOM, (2000), pp. 255–265.
[9] D.R. Raymond, S.F. Midkiff, Denial-of-service in wireless sensor networks: attacks and defense, IEEE Pervasive Comput. 7 (1) (2008) 74–81.
[10] T.H. Hai, E. Huh, Detecting selective forwarding attacks in wireless sensor networks using two-hops neighbor knowledge, Proceedings of IEEE NCA, (2008), pp. 325–331.
[11] B. Yu, B. Xiao, Detecting selective forwarding attacks in wireless sensor networks, IEEE IPDPS, (2006), pp. 1–8.
[12] B. Xiao, B. Yu, C. Gao, CHEMAS: identify suspect nodes in selective forwarding attacks, J. Parallel Distrib. Comput. 67 (11) (2007) 1218–1230.
[13] K. Liu, J. Deng, P.K. Varshney, K. Balakrishnan, An acknowledgment-based approach for the detection of routing misbehavior in MANETs, IEEE Trans. Mob. Comput. 6 (5) (2007) 536–550.
[14] D.M. Shila, C. Yu, T. Anjali, Mitigating selective forwarding attacks with a channel-Aware approach in WMNs, IEEE Trans. Wirel. Commun. 9 (5) (2010) 1661–1675.
[15] X. Li, R. Lu, X. Liang, X. Shen, Side channel monitoring: packet drop attack detection in wireless ad hoc networks, Proceedings of IEEE ICC, (2011), pp. 1–5.
[16] E.M. Shakshuki, N. Kang, T.R. Sheltami, EAACK: a secure intrusion-detection system for MANETs, IEEE Trans. Ind. Electron. 60 (3) (2013) 1089–1098.
[17] Q. Liu, J. Yin, V. Leung, Z. Cai, FADE: forwarding assessment based detection of collaborative grey hole attacks in WMNs, IEEE Trans. Wirel. Commun. 12 (10) (2013) 5124–5137.
[18] J. Ren, Y. Zhang, K. Zhang, X.S. Shen, Exploiting channel-aware reputation system against selective forwarding attacks in WSNs, Proceedings of IEEE GLOBECOM, (2014), pp. 330–335.
[19] J.-M. Chang, P.-C. Tsou, I. Woungang, H.-C. Chao, C.-F. Lai, Defending against collaborative attacks by malicious nodes in MANETs: a cooperative bait detection approach, IEEE Syst. J. 9 (1) (2015) 65–75.
[20] R.H. Jhaveri, N.M. Patel, A sequence number based bait detection scheme to Thwart grayhole attack in mobile ad hoc networks, Wirel. Netw. 21 (8) (2015) 2781–2798.
[21] C. Pu, S. Lim, A light-weight countermeasure to forwarding misbehavior in wireless sensor networks: design, analysis, and evaluation, IEEE Syst. J. (2016).
[22] S. Lim, H. Lauren, Hop-by-hop cooperative detection of selective forwarding attacks in energy harvesting wireless sensor networks, Proceedings of ICNC, (2015), pp. 315–319.
[23] A. Varga, OMNeT++, 2014. http://www.omnetpp.org/.
[24] Y. Chae, L.C. DiPippo, Y.L. Sun, Trust management for defending on-off attacks, IEEE Trans. Parallel Distrib. Syst. 26 (4) (2015) 1178–1191.
[25] J. Ren, Y. Zhang, K. Zhang, X. Shen, Adaptive and channel-aware detection of selective forwarding attacks in wireless sensor networks, IEEE Trans. Wirel. Commun. 15 (5) (2016) 3718–3731.
[26] C. Pu, S. Hajjar, Mitigating forwarding misbehaviors in RPL-basedlow power and lossy networks, Proceedingsof IEEE CCNC, (Jan 2018).
[27] C. Pu, S. Lim, Spy vs. spy: camouflage-based active detection in energy harvesting motivated networks, Proceedings of MILCOM, (2015), pp. 903–908.
[28] D. Midi, E. Bertino, Node or link? Fine-grained analysis of packet-loss attacks in wireless sensor networks, ACM Trans. Sensor Netw. 12 (2) (2016).
[29] Y. Zhang, L. Lazos, W. Kozma, AMD: audit-based misbehavior detection in wireless ad hoc networks, IEEE Trans. Mob. Compu. 15 (8) (2016) 1893–1907.
[30] M. Stehlik, V. Matyas, A. Stetsko, Towards better selective forwarding and delay attacks detection in wireless sensor networks, Proceedings of IEEE ICNSC, (2016), pp. 1–6.
[31] S. Lim, J. Kimn, H. Kim, Analysis of energy harvesting for vibration-motivated wireless sensor networks, Proceedings of ICWN, (2010), pp. 391–397.
[32] 2.4 GHz IEEE 802.15.4/ZigBee-ready RF transceiver, http://www.ti.com/lit/ds/symlink/cc2420.pdf (Last accessed at Feb 2018).
[33] Cisco Aironet 802.11a/b/g wireless CardBus adapter, https://www.cisco.com/c/en/us/products/collateral/wireless/aironet-802-11a-b-g-cardbus-wireless-lan-client-adapter-cb21ag/product_data_sheet09186a00801ebc29.html (Last accessed at Feb 2018).
[34] T. Starner, Human-powered wearable computing, IBM Syst. J. 35 (3 & 4) (1996) 618–629.
[35] T. Starner, J.A. Paradiso, Human Generated Power for Mobile Electronics, in: CRC Press, 2004, pp. 1–35.
[36] Z.L. Wang, Nanogenerators for Self-powered Devices and Systems, Georgia Institute of Technology, Atlanta, USA, 2011.
[37] X. Xue, S. Wang, W. Guo, Y. Zhang, Z.L. Wang, Hybridizing energy conversion and storage in a mechanical-to-electrochemical process for self-charging power cell, Nano Lett. 12 (9) (2012) 5048–5054.
[38] Z.A. Eu, H. Tan, W.K.G. Seah, Design and performance analysis of MAC schemes for wireless sensor networks powered by ambient energy harvesting, Ad Hoc Netw. 9 (3) (2011) 300–323.
[39] C. Fujii, W.K.G. Seah, Multi-tier probabilistic polling for wireless sensor networks powered by energy harvesting, Proceedings of IEEE ISSNIP, (2011), pp. 383–388.
[40] C. Pu, T. Gade, S. Lim, M. Min, W. Wang, Light-weight forwarding protocols in energy harvesting wireless sensor networks, Proceedings of MILCOM, (2014), pp. 1053–1059.
[41] W. Stallings, Cryptography and Network Security - Principles and Practices, 6th Edition, Prentice Hall, 2013.
[42] A. Boulis, Castalia, 2014. http://castalia.forge.nicta.com.au.
[43] X. Tang, J. Xu, Extending network lifetime for precision-constrained data aggregation in wireless sensor networks, INFOCOM, (2006), pp. 1–12.
[44] S. Jiang, Efficient Network Camouflaging in Wireless Networks, Ph.D. thesis. Texas A&M University, 2006.
[45] J. Deng, R. Han, S. Mishra, INSENS: intrusion-tolerant routing for wireless sensor networks, Comput. Commun. 29 (2) (2006) 216–230.
[46] Q. Fang, J. Gao, L.J. Guibas, Locating and bypassing holes in sensor networks, Mob. Netw. Appl. 11 (2) (2006) 187–200.