Chapter 1

Hide-and-Detect: Forwarding Misbehavior, Attack, and Countermeasure in Energy Harvesting Motivated IoT Sensor Networks

Sunho Lim[§1] Cong Pu[2] Jinseok Chae[§3] Manki Min[4] and Yi Liu[5]

Internet-of-Things (IoT) sensor networks (IoTSNs), where a set of multi-scale, heterogeneous, and batter-powered sensors and devices (later in short, nodes) is seamlessly blended for network functions, are increasingly popular and deployed in diverse applications ranging from civil to military. Due to the limited amount of battery, IoTSNs powered by harvesting environmental resources have been rapidly emerging as a major part of ubiquitous computing and communication infrastructure. More importantly, IoTSNs are admittedly vulnerable to a denial-of-service (DoS) attack because of the shared medium, lack of resources and centralized coordination, and limited computing and communicating capabilities. In this chapter, we present a countermeasure, called EYES, to the forwarding misbehavior of single and multiple colluding nodes in the IoTSNs. We analyze a set of adversarial scenarios and identify its potential vulnerable cases. In the EYES, each node actively disguises itself as an energy harvesting node and stealthily monitors the forwarding operations of adjacent nodes. Each node also periodically collects the information of overheard packets from adjacent nodes and validates prior uncertain forwarding operations. We conduct the performance evaluation and comparison with competitive existing schemes through extensive simulation experiments using the OMNeT++. The results show the effectiveness of proposed countermeasure in terms of higher detection rate, lower detection latency, and competitive packet delivery ratio. We also comprehensively compare detection strategies of forwarding misbehavior. In

[1]T[2]WISTOR: TTU Wireless Mobile Networking Laboratory, Dept. of Computer Science, Texas Tech University, Lubbock, TX 79409, Email: sunho.lim@ttu.edu, [§]Co-corresponding author
[2]Dept. of Computer Sciences and Electrical Engineering, College of Engineering and Computer Sciences, Marshall University, Huntington, WV 25755, Email: puc@marshall.edu
[3]Dept. of Computer Science and Engineering, Incheon National University, South Korea, Email: jschae@inu.ac.kr, [§]Co-corresponding author
[4]Computer Science Program, Louisiana Tech University, Ruston, Louisiana 71272, Email: mankimin@latech.edu
[5]Dept. of Computer and Information Science, University of Massachusetts, Dartmouth, MA 02747, Email: yliu11@umassd.edu

addition, we explore several new research directions in energy harvesting and their potential applicabilities with diverse points of view.

1.1 Introduction

Internet-of-Things (IoT) sensor networks (IoTSNs) and their applications are quickly emerging, where a set of multi-scale, heterogeneous, and battery-powered sensors and devices (later in short, nodes) is wirelessly interconnected together for actuation, sensing, and communication. Recent advances of technology have fueled the development of a tiny and low-power node available to expedite fast deployment and improve portability, availability, and accessibility. IoTSN applications have been widely deployed in a variety of areas, such as smarthome, healthcare, infrastructure monitor, transportation & logistics, surveillance, and so on. Since the demand of IoTSN applications is rapidly increasing globally, the IoT market is predicted to reach more than 2 trillion by 2023 which is three times higher compared to that of 2016 [1]. We envision that IoTSNs will not only play an important role in realizing diverse applications ranging from civilian to military, but also become the next generation of ubiquitous computing and communication infrastructure.

Since nodes are primarily powered by batteries, it is ultimately unavoidable to replace or replenish batteries. This could be a critical issue if multiple nodes are deployed in a hard-reach area or a very wide area. It would be hard (if it is not impossible) to manually replace or replenish batteries. In light of this, we investigate an energy harvesting motivated IoTSN to remove batteries or at least reduce the frequency of replacing batteries. In IoTSN, each self-sustainable node equipped with energy harvesting capabilities from an immediate environment (e.g., solar, wind, thermal, etc.) can communicate with others directly or indirectly via multi-hop relays.

Although a great effort has been devoted in energy harvesting, we will focus on the cybersecurity in the sense of forwarding misbehaviors, attacks, and countermeasures in IoTSNs. In this chapter, we first review and analyze prior detection strategies of forwarding misbehavior in Section 1.2. Then we introduce a piezoelectric fiber composite bi-morph (PFCB) W14 based energy harvesting, a charge-and-spend energy harvesting policy, and system and adversary models in Section 1.3. In Section 1.4, we investigate energy harvesting motivated forwarding attacks and identify their vulnerabilities, misbehaviors, and scenarios observed in IoTSNs. We present our approaches to mitigate the forwarding misbehaviors and attacks followed by extended performance evaluation and comparison in Sections 1.5 and 1.6. Finally, we discuss future research directions with interdisciplinary aspects and insights and conclude the chapter in Sections 1.7 and 1.8.

1.2 Background and Related Work

In this section, we categorize and analyze existing detection techniques against forwarding misbehaviors in terms of monitor-, acknowledgment-, inducement-based,

and other approaches. We also comprehensively compare detection strategies of forwarding misbehavior deployed in diverse networks and summarize their properties in Table 1.3.

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 [2], the authors propose a centralized detection system (CDS) to detect packet dropping attacks in clustered IoT networks. The basic idea is to use the uplink packet drop probability of the IoT devices to monitor the behavior of the gateway with which they are associated. In addition, the proposed detection rule is given by the generalized likelihood ratio test, where the attack probabilities are estimated using maximum likelihood estimation. The authors propose a heuristic-based detection (HED) scheme against the suppression attack in multicast protocol for Low Power and Lossy Networks in [3, 4]. In the HED, a malicious node multicasts a series of spoof data messages with continuous sequence numbers to prevent normal nodes from accepting valid data messages and cause them to delete cached data messages. In the hop-by-hop cooperative detection (HCD) [5], 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. A monitor-based approach (CMD) [6] 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.

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 [7] and its extension, called checkpoint-based multi-hop acknowledgment scheme (CHEMAS) [8], 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 wireless sensor networks (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) [9] 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. An acknowledgment-based pun-

ishment and stimulation scheme (APS) [10] is proposed to punish malicious nodes dropping data packets in Mobile Ad Hoc Networks (MANETs). The basic idea of the APS is to combine data and routing reputation to assess the direct reputation of neighbor nodes and share their recommendations about other nodes to identify malicious nodes.

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) [11] 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) [12], 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) [13], 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.

Other Approach: In [14], a comprehensive attack detection framework is proposed to detect several IoT cyber-attacks using deep learning. The proposed framework implements an attack detector on fog nodes due to its distributed nature, high computational capacity and proximity to edge devices. The authors in [15] propose a dependence estimator-based scheme for DoS attack traffic classification in IoT sensor networks, where the deep analysis of IoT network traffic parameters is conducted to establish a relationship between the network traffic parameters and to identify the key IoT network traffic parameters upon which other parameters would depend. Through the establishment of such a relationship, the analysis of network traffic for malicious activity proves to be more accurate as well as efficient. In [16], an Accurate and Cognitive Intrusion Detection System (ACIDS) has been developed to detect the most vulnerable packet dropping attack known as black hole attack in MANETs. The ACIDS takes the parameters such as destination sequence number and route reply into consideration for detecting the intruders by identifying the deviation of the chosen parameters from the normal behavior.

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.

1.3 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.

System Model: In this chapter, we consider an Internet-of-Things (IoT) sensor network (IoTSN), where each node is assumed to equip with an energy harvesting device to replenish its rechargeable battery [17] 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[6] [20, 21, 22]. It is envisaged that multi-scale piezo devices and integrated self-charging power cells (SCPCs) [23] will enhance the efficiency of energy harvesting.

In IoTSN, 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 chapter, 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 1.6. 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 [5, 13, 24, 25], 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 an one-hop *State* packet to prevent other nodes from mistakenly forwarding a packet to the nodes in harvest state.

When a node detects an event, it initiates to generate and forward a data packet toward a sink. A simple broadcast-based forwarding [26] 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

[6] For instance, the IEEE 802.15.4-compliant Texas Instrument Chipcon CC2420 radio [18] 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 [19] also support four (1, 5, 10, and 30 mW) and six (1, 5, 20, 30, 50, and 100 mW) transmission power levels, respectively.

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.

Adversarial Model: The primary goal of adversary is to attack service availability by 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 [27], a receiver can easily verify the packet and detect any modification. In this chapter, 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 chapter, 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.

1.4 Energy Harvesting Motivated Misbehavior and Selective Forwarding Attack

In this section, we investigate potential forwarding misbehaviors by identifying a set of adversarial scenarios and observing its vulnerable cases in IoTSNs.

Adversarial Scenario: We present eight adversarial scenarios using a snapshot of network consisting of five energy harvesting enabled nodes in Fig. 1.1, where two malicious nodes (n_{m_A} and n_{m_B}) are located along the forwarding path and they could monitor network traffic and collude selective forwarding attacks together. The different states of legitimate and malicious nodes corresponding to the adversarial scenarios are also presented in Table 1.1. Since the presented adversarial scenarios are carefully chosen by observing the interactions between legitimate and malicious nodes in IoTSNs, 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 could 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 [8, 9] to discourage the forwarding misbehavior of interleaved malicious nodes. In addition, we expect that IoTSN 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 on the network performance.

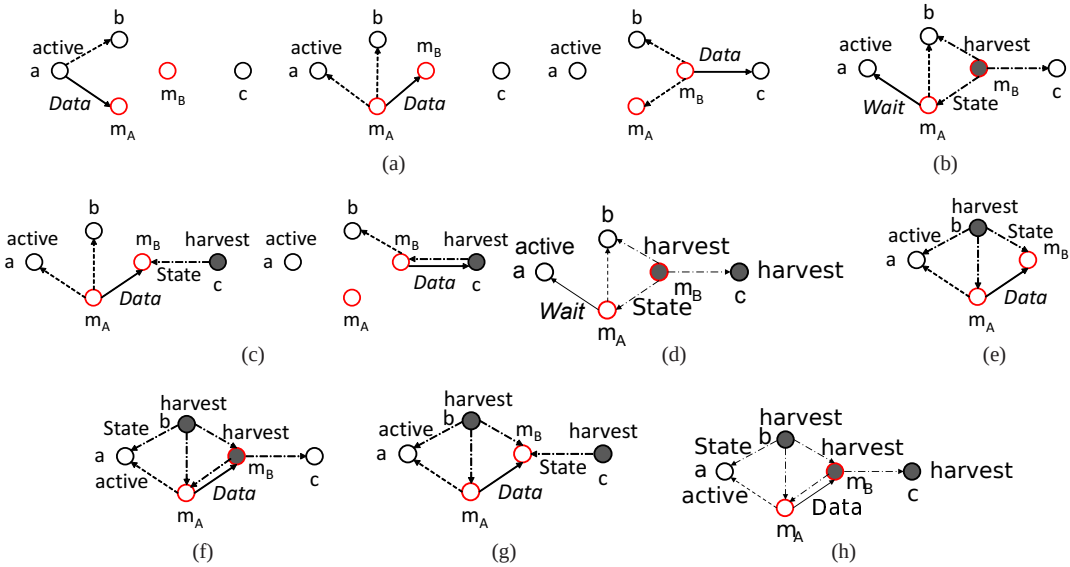


Figure 1.1 A set of adversarial scenarios and its vulnerable cases in the presence of malicious nodes in IoTSNs. Here, a malicious node and a node in harvest state are marked as red and shade, respectively. Solid, dotted, and dash-dotted lines represent forwarding, overhearing, and periodic broadcast operations, respectively.

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. 1.1. 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 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 chapter, 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

Table 1.1 The summary of states for legitimate and malicious nodes in the adversarial scenarios.

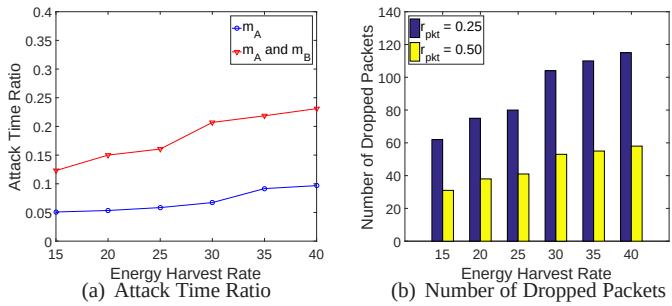
n_a	n_{m_A}	n_b	n_{m_B}	n_c	Scenario
Active	Active	Active	Active	Active	Subfig. 1.1(a)
Active	Active	Active	Harvest	Active	Subfig. 1.1(b)
Active	Active	Active	Active	Harvest	Subfig. 1.1(c)
Active	Active	Active	Harvest	Harvest	Subfig. 1.1(d)
Active	Active	Harvest	Active	Active	Subfig. 1.1(e)
Active	Active	Harvest	Harvest	Active	Subfig. 1.1(f)
Active	Active	Harvest	Active	Harvest	Subfig. 1.1(g)
Active	Active	Harvest	Harvest	Harvest	Subfig. 1.1(h)

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 chapter.

Potential Forwarding Behaviors: We show two adversarial scenarios, where malicious nodes can potentially conduct 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 their local cache as shown in Subfig. 1.1(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 Subfig. 1.1(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 Subfig. 1.1(d), n_{m_A} still need to perform the same operations as mentioned above, otherwise, the forwarding misbehavior can be easily detected.

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 Subfig. 1.1(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



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

Figure 1.2 The changes of attack time ratio and number of dropped packets against energy harvest rate and packet injection rate.

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 Subfig. 1.1(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 Subfig. 1.1(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 Subfig. 1.1(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 Subfig. 1.1(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.

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. 1.2. Here, t_{at} and t_{tot} are total attack time of undetected forwarding misbehaviors and total observation time (e.g., 1,000 (sec)), 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 Subfigs. 1.1(c), (e), (f), (h), and (g). In Subfig. 1.2(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 Subfig. 1.2(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/sec). This is because two colluding malicious nodes receive more packets with smaller r_{pkt}, resulting in more packets dropped due to undetected forwarding misbehaviors.

1.5 Energy Harvesting Motivated DoS Attacks and Their Countermeasures

We propose a countermeasure, called *EYES*, to efficiently detect forwarding misbehaviors of colluding malicious nodes in the IoTSNs. The proposed countermeasure consists of two schemes: *SlyDog* and *LazyDog*. The EYES is different from the prior approaches, [28, 29, 30, 7, 8, 31, 32, 33, 34, 35, 36, 9], where each node only *passively* monitors any forwarding misbehavior witnessed in an adversarial case for detection in the battery-powered networks.

1.5.1 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 [13], to detect a collusion of malicious nodes.

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 an 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.

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.

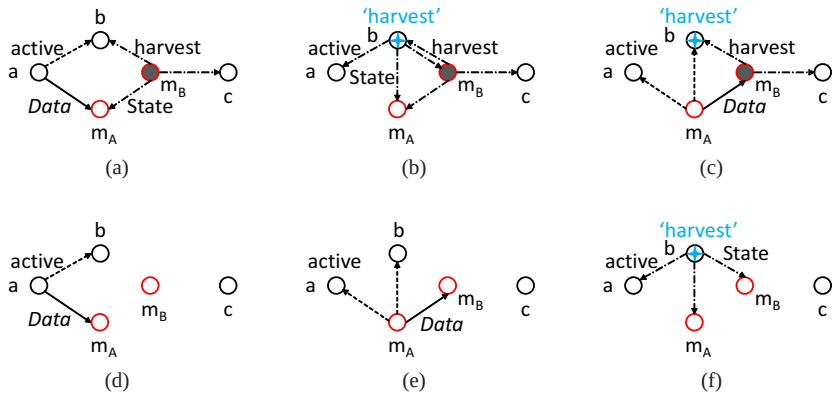


Figure 1.3 The proposed SlyDog detection operations.

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 chapter, 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 1.6. 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.

Detection Operations: Under the basic operations, we present detection operations of SlyDog with a set of snapshots of networks in Fig. 1.3. 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 Subfig. 1.3(a). Then n_b checks the state of its one-hop neighbor nodes based on the state table, ST_b. In Subfig. 1.3(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 peri-

odically broadcasting a *State* packet piggybacked with its harvest state. In Subfig. 1.3(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 Subfig. 1.1(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 piggybacked 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 Subfig. 1.3(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 Subfig. 1.3(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 Subfig. 1.3(f), when n_{m_B} overhears a harvest *State* packet from n_b, it is the aforementioned vulnerable case (see Subfigs. 1.1(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 detection operations of the SlyDog are summarized in Fig. 1.4.

1.5.2 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 received by n_c because n_b is not aware of the state of n_c as shown in Subfig. 1.1(e).

Detection Operations: Under the basic operation of SlyDog, we present detection operations of LazyDog with a set of snapshots of networks in Fig. 1.5. 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 Subfig. 1.5(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 Subfig. 1.5(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 Subfig. 1.5(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

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 $n_x \in S_i$ **and** $n_y \in G_i^*$
    for $n_z \in C_{i,y}$ **and** $n_z \in F_i$
      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** $n_{vim} \in C_{i,tget_{slyA}}$
      $MT_i[tget_{slyA}].p \mathrel{+}= \delta$; $MT_i[tget_{slyA}].c_{mis} \mathrel{+}= 1$;
  ▷: n_i overhears a *Wait* packet $pkt[Wait, tget_{slyA}, src, seq]$.
    if $ST_i[vim].s == s_h$ **and** $n_{vim} \in C_{i,tget_{slyA}}$
      Stop the *SlyDog* on $n_{tget_{slyA}}$;
    else
      $MT_i[tget_{slyA}].p \mathrel{+}= \delta$; $MT_i[tget_{slyA}].c_{mis} \mathrel{+}= 1$;
  ▷: n_i does not overhear any packet within timeout period.
    $MT_i[tget_{slyA}].p \mathrel{+}= \delta$; $MT_i[tget_{slyA}].c_{mis} \mathrel{+}= 1$;

◇ n_i overhears a *Data* packet $pkt[Data, tget_{slyA}, rec, seq]$.
  if $rec \in F_i$ **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 $pkt[Data, tget_{slyB}, nrec, seq]$.
    $AT_i[tget_{slyB}, nrec].op \mathrel{+}= 1$;
    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.
    $MT_i[tget_{slyB}].p \mathrel{+}= \delta$; $MT_i[tget_{slyB}].c_{mis} \mathrel{+}= 1$;

Figure 1.4 A pseudo code of detection operations in the SlyDog.

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 Subfig. 1.5(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 detection operations of the LazyDog are summarized in Fig. 1.6.

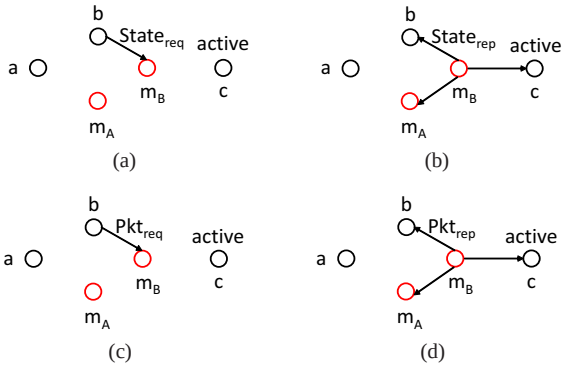


Figure 1.5 The proposed LazyDog detection operations.

1.6 Performance Evaluation and Analysis

1.6.1 Simulation Testbed

We conduct extensive simulation experiments using the OMNeT++ [37] 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 [38], and the channel error rate is set to 5%. A single node generates data traffic with injection rate 0.33 or 0.66 (pkt/sec) and the data packet size is 1 KByte. The inter-arrival time of traffic is assumed to be exponentially distributed. The total simulation time is 1,000 seconds. The periods of active and harvest states vary between 50 to 80 (secs) and 15 to 40 (secs), 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. A set of interactions and its corresponding operations are visually presented in Figs. 1.1, 1.3, and 1.5.

For performance comparison, we compare our proposed schemes, *EYES*, with a hop-by-hop cooperative detection scheme, called *HCD* [5], which is the first countermeasure to selective forwarding attack in EHNets. The EYES is also compared with the well-known scheme, *Watchdog* [28]. 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 1.2.

1.6.2 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

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
    $MT_x[t].c_{mis}$ += $(|RP_x[t] - FP_t[x]|)$;

Figure 1.6 A pseudo code of detection operations in the LazyDog.

simulation parameters, including energy harvest time (t_h), the number of malicious node, and monitor probability (δ).

Detection Rate: We first measure the detection rate by changing harvest time (t_h), packet injection rate (r_{pkt}) and δ in Subfigs. 1.7(a) and (b). In Subfig. 1.7(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

Table 1.2 Simulation Parameters

Parameter	Value
Network size	$200\times200\ m^2$
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/sec
Packet size	1 KByte
Packet drop rate of HCD and Watchdog	30%
Radio range	12.3 m
Radio model	CC2420
Simulation time	1000 secs
Active time period	50 to 80 secs
Harvest time period	15 to 40 secs

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 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 Subfig. 1.7(b), overall detection rates of the EYES and HCD increase with $r_{pkt} = 0.66$ (pkt/sec), 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.

Detection Latency: Second, the detection latency is measured by changing t_h, r_{pkt}, and δ in Subfigs. 1.7(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 Subfigs. 1.1(c), (e), (f), (g), and (h). In Subfig. 1.7(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 net-

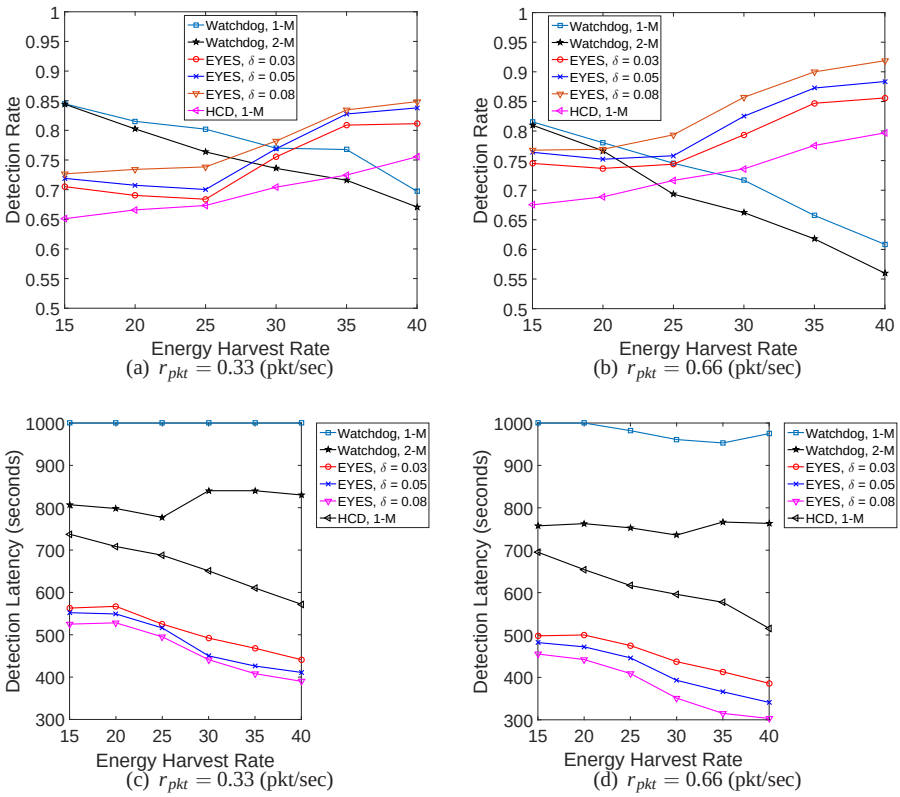
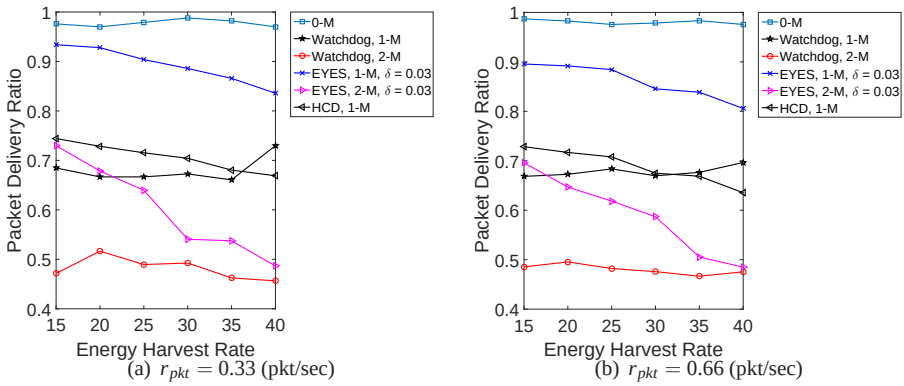


Figure 1.7 The performance impact against energy harvest rate, number of malicious nodes, packet injection rate, and δ.

work. 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*[7] 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 Subfig. 1.7(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.

Packet Delivery Ratio: Third, we measure the packet delivery ratio (PDR) by varying t_h, r_{pkt}, and δ in Fig. 1.8. 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

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



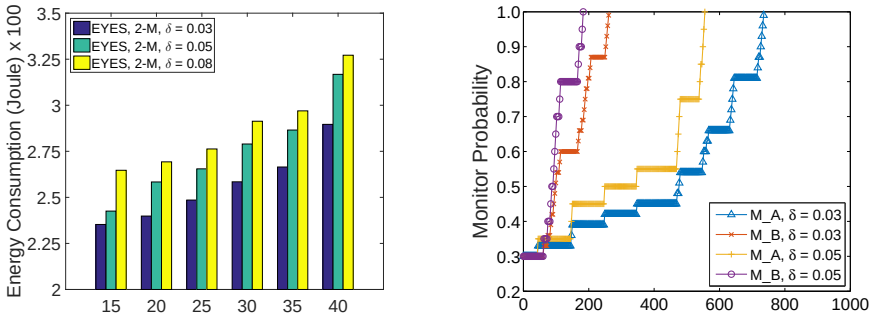
(a) $r_{pkt} = 0.33$ (pkt/sec) (b) $r_{pkt} = 0.66$ (pkt/sec)

Figure 1.8 The packet delivery ratio against energy harvest rate, packet injection rate, and δ.

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 Subfig. 1.8(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 Subfig. 1.8(b), overall PDRs decrease with higher packet injection rate (i.e., 0.66 (pkt/sec)) because more packets are dropped by malicious nodes due to more number of generated packets in the network.

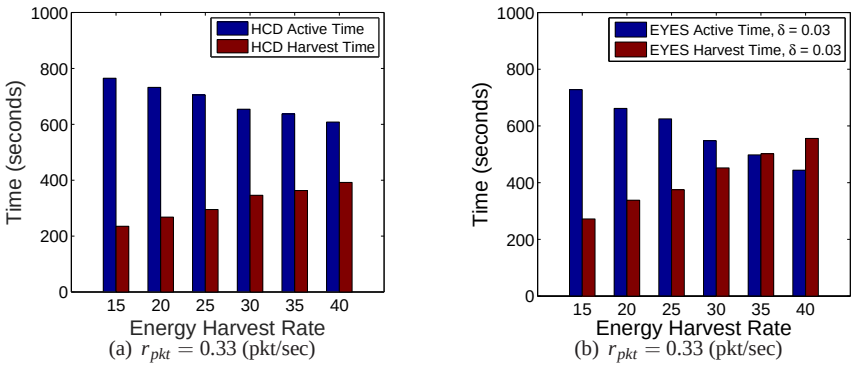
Energy Consumption: Fourth, we measure energy consumption in terms of the number of overheard forwarding misbehaviors of malicious nodes [39] in Subfigs. 1.9(a). An overheard forwarding misbehavior occurs when a malicious node forwards a packet to a legitimate node which is in harvest state, resulting 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. 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. Thus, the EYES can efficiently utilize the harvested energy to monitor and detect the forwarding misbehaviors of malicious nodes.

Monitor Probability: Fifth, we observe the changes of monitor probability (p) in the presence of two malicious nodes with different weights ($\delta = 0.03$ or 0.05) over



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

Figure 1.9 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)

Figure 1.10 The performance of total active and harvest time periods against energy harvest rate and δ.

the simulation period in the EYES as shown in Subfig. 1.9(b). 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 Subfig. 1.3(a)) reach to 1.0 at 580 and 180 seconds 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.

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 over the simulation period as shown in Subfigs. 1.10(a) and (b). 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 Subfig. 1.10(a). In the EYES, however, total harvest time period in Subfig. 1.10(b) increases quickly compared to

that of the HCD in Subfig. 1.10(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 addition, we comprehensively compare the proposed approach with detection schemes deployed in diverse networks and summarize their properties in terms of six aspects in Table 1.3 extended from [40]: (i) collusive attack, (ii) computational overhead; (iii) communication overhead; (iv) detection delay; (v) punishment; and (vi) architecture.

Table 1.3 The comparison† of detection strategies of forwarding misbehavior.

Approach	Collusive Attack	Computation Overhead	Communication Overhead	Detection Latency	Punishment	Architecture
Watchdog [28]	N	Low	N	N	N	Stand-alone
CHEMAS [8]	N	Medium	High	Low	N	Centralized
CAD [41]	N	Medium	Medium	Medium	N	Centralized
SCM [33]	N	Low	N	Medium	N	Stand-alone
EAACK [34]	N	Medium	High	Medium	N	Centralized
FADE [35]	Y	Medium	High	Low	N	Centralized
CRS [36]	Y	High	Medium	Medium	Y	Distributed
CBDS [11]	Y	Medium	Medium	High	N	Distributed
SNBDS [12]	Y	Medium	Medium	High	N	Distributed
SCAD [42]	Y	Medium	Medium	Low	N	Centralized
HCD [5]	N	Medium	Low	High	Y	Distributed
CAM [13]	N	Low	N	N	Y	Stand-alone
CDS [2]	N	High	Low	Medium	N	Centralized
HED [4]	Y	Low	Low	Medium	Y	Distributed
APS [10]	N	Medium	High	Medium	Y	Distributed
ACIDS [16]	N	Medium	N	Medium	Y	Centralized
SlyDog	Y	Low	N	N	Y	Stand-alone
LazyDog	N	Low	Low	Medium	Y	Distributed

† In this chapter, 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* [9]: *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.

picture(0,0)(-42,0)(1,0)30 (0,-42)(0,1)30 picture picture(0,0)(42,0)(-1,0)30 (0,42)(0,-1)30 picture
picture(0,0)(-42,0)(1,0)30 (0,-42)(0,1)30 picture picture(0,0)(42,0)(-1,0)30 (0,42)(0,-1)30 picture

1.7 Discussion and Future Research Directions

In this section, we investigate the applicability of the proposed approach to other attacks and discuss a wide range of future research directions in IoTSNs.

1.7.1 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 [28].

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 Subfig. 1.3(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.

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 Subfig. 1.3(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}.

1.7.2 Future Research Directions and Applicability

We envision that energy harvesting-based computing would be essential in future IoT and explore its potential applicability with diverse research directions.

Vibration Sensitive Medium Access Control: In vibration motivated energy harvesting, a disturbance event initiates the direct piezoelectric effect actively or passively. A passive event can be caused by surrounding environmental resources (e.g., ground disturbance or wind) in a static IoTSN, where the nodes located nearby the event sense and transform it into mechanical vibration energy for communication. Since multiple nodes can respond to the same event, they may initiate the transmission simultaneously that may result in packet contention, collision, and retransmission. On the other hand, an active event can be caused by immediate environmental resources (e.g., kinetic motion of walking or running) in a mobile IoTSN, where

each individual node responds to the event. Note that each node must maximize the utilization of harvested energy for communication.

The prior energy harvesting aware MAC protocols have concentrated on solar- [24, 43] or thermal-based [44] energy harvesting. However, there is a plenty of space to extend by deploying vibration motivated energy harvesting from intermittent kinetic movements and its integration with the IEEE 802.11 MAC protocol. This research approach newly considers underlying properties of ambient vibrations and practical obstacles in terms of medium access technique that will significantly affect the design of algorithms and communication protocols embedded in upper layers, such as the network and application layers.

Energy Harvesting Motivated Lower Power and Lossy Networks: The vision of Internet of Things (IoT) foresees a future communication paradigm in which information systems will be seamlessly integrated with heterogeneous smart sensors and objects that are capable of communicating with each other without human intervention [45]. In the realm of IoT, IPv6-based Lower Power and Lossy Networks (LLNs) consisting of a myriad of resource-constrained devices endowed with the capabilities of sensing, computing, and wireless communicating represent a key enabler for IoT applications. Due to the limited battery power, it is ultimately unavoidable to replace or replenish batteries. Thus, energy harvesting has emerged as a promising technology to extend the lifetime of devices by continuously harvesting environmental resources, such as sunlight, wind, vibration, etc [46]. We envision that Energy Harvesting Motivated Lower Power and Lossy Networks (EH-LLNs) will be a major part of ubiquitous computing and communication infrastructure in IoT, where a set of self-sustainable devices equipped with energy harvesting capabilities communicate using RPL routing protocol [47], which is a novel routing protocol specifically designed for LLNs. However, RPL was not originally designed with the consideration of energy harvesting feature. To investigate the effects of energy harvesting on extending LLNs' lifetime, we plan to develop an energy harvesting module to seamlessly integrate with RPL routing protocol and conduct different simulation scenarios by using Contiki Cooja network simulation [48]. In addition, we plan to investigate the dissipation of harvested energy and traffic load, and design a traffic load and energy balancing RPL routing protocol to further extend the network lifetime and improve the network performance.

Authentication with Lightweighted Cryptography: We can find the presented scheme to be applicable to IoT device authentication, especially collaborative authentication on a group of IoT nodes using threshold cryptography [49, 50, 51]. The proposed countermeasure will allow us to further filter out the honest behaving nodes or generating the honestness weights, so when it is combined with some other security measures we can further strengthen the threshold and improve the security level of the IoT network nodes. We can imagine that this level of work may happen at a much more powerful node such as a base station which possibly possess the entire (or at least majority of) the IoT node topology and the honestness of each node can aid the more accurate computation of group security measures.

Another research direction related to the proposed scheme is the authentication itself of each IoT device using very lighweight cryptographic functions such as cryp-

tographic hash functions. Traditionally hash chain was found to be useful to balance off the computational overhead and the security level [52, 53, 54, 55], but it does not work well to handle more complicated hierarhical structure. Hierarchical structured authentication has extensively been studied [56, 57, 58, 59], and we can use more complicated hash structure such as hash Merkle tree, multidimensional hash chains, and hash vine. By designing a lightweight hash function to sacrifice the collision attack security, we can make it to work with low computing powered IoT devices for lower security but shorter lifetime protection of broadcast communications. This is a plausible direction in the sense that the lifetime of each broadcast is very short, so each hash computation needs to be safe against collision attack for short period time that can be achieved with the lightweighted hash design.

IoT Software Architecture: IoT, as a rapidly growing field, has applications in various domains such as healthcare, automated home services, smart energy and smart grid, food and water tracking, and transportation, and etc. [60, 61]. *"Software engineering for the IoT poses challenges in light of new applications, devices, and services."* [62]. The IoT adds additional complexity to software development as its nature of distribution and inclusion of heterogeneous devices, such as sensors and actuators [62]. One of the areas of research in the IoT from the Software Engineering perspective is software architecture.

Several reference architectures have been proposed in order to standardize the design of IoT systems, in which some reference architectures are more generic on industry scale implementation [62] while some are more specific [63] to the resources or environment, such as cloud computing. Some research targets on specific software architecture for the IoT applications in different domains, for example, [64] presents a service-oriented software architecture for a data-driven smart city utilities application and [65] did a mapping study on using microservice architecture as the building blocks for IoT systems and cloud computing solutions. The research work [66, 67] have done mapping studies on exploiting software architecture models to develop IoT systems.

Although reference architectures give the software developers a general guide and the specific software architectures proposed for difference domains and resources allow the developers to adapt the methodologies while developing the IoT systems similar settings, there are still scenarios in which these architectures are not applicable. The energy harvesting-based wireless sensor networks systems [68] involve the special requirements that need to be addressed in the software architectural design. The existing proposed IoT software architectures may need to be extended and expanded with the unique aspects of wireless sensor networks and energy harvesting-based computing involved. Thus, as one of the future works, we plan to exploit the software architectural styles that work as the best practice in IoTSN powered by harvesting environmental resources.

1.8 Concluding Remarks

In this chapter, we investigated the forwarding misbehavior and its countermeasure in the realm of IoTSNs. Under the charge-and-spend harvesting policy, a set of ad-

versarial scenarios is investigated and its potential vulnerabilities are also identified and analyzed. We proposed a countermeasure, called *EYES*, to efficiently detect the forwarding misbehaviors of multiple malicious nodes in the IoTSNs. The EYES is a combination of inducement- and monitor-based approaches to quickly identify the lurking deep malicious nodes and isolate them from the network. We conducted extensive simulation experiments and their results show that the EYES provides 70 to 92% detection rate and achieves 23 to 60% lower detection latency compared to the HCD and Watchdog. The EYES also shows a competitive performance in PDR.

In addition, we compared the detection strategies of forwarding misbehavior comprehensively and presented a potential applicability of the EYES to other well-known attacks. More importantly, we provided a set of future research directions with diverse aspects for IoT and its variants, including energy harvesting aware medium access control and low power and lossy networks, authentication, and software architecture.

Appendix

In Table 1.4, we summarize all the key acronyms and schemes used in this chapter.

References

[1] Internet of Things (IoT) Market -Share, industry Trends, Development, Revenue, Demand and Forecast, to 2023;. Https://www.marketwatch.com/.

[2] Abhishek NV, Tandon A, Lim T, et al. Detecting forwarding misbehavior in clustered IoT networks. In: Proc. ACM International Symposium on QoS and Security for Wireless and Mobile Networks; 2018. p. 1–6.

[3] Pu C, Zhou X, Lim S. Mitigating Suppression Attack in Multicast Protocol for Low Power and Lossy Networks. In: Proc. IEEE LCN; 2018. p. 251–254.

[4] Pu C, Zhou X. Suppression Attack Against Multicast Protocol in Low Power and Lossy Networks: Analysis and Defenses. Sensors. 2018;18(10):3236.

[5] Lim S, Huie L. Hop-by-Hop Cooperative Detection of Selective Forwarding Attacks in Energy Harvesting Wireless Sensor Networks. In: Proc. Int'l Conf. on Computing, Networking and Communications (ICNC); 2015. p. 315–319.

[6] Pu C, Hajjar S. Mitigating Forwarding Misbehaviors in RPL-based Low Power and Lossy Networks. In: Proc. IEEE CCNC; Jan 2018. .

[7] Yu B, Xiao B. Detecting Selective Forwarding Attacks in Wireless Sensor Networks. In: IEEE IPDPS; 2006. p. 1–8.

[8] Xiao B, Yu B, Gao C. CHEMAS: Identify Suspect Nodes in Selective Forwarding Attacks. Journal of Parallel and Distributed Computing. 2007;67(11):1218–1230.

[9] Pu C, Lim S. A Light-Weight Countermeasure to Forwarding Misbehavior in Wireless Sensor Networks: Design, Analysis, and Evaluation. IEEE Systems Journal. 2018;12(1):834–842.

Table 1.4 Summary of acronyms and schemes

Acronym	Description
ACIDS	Accurate and cognitive intrusion detection system
AODV	Ad hoc on-demand distance vector routing
APS	Acknowledgment-based punishment and stimulation scheme
CAD	Channel aware detection
CAM	Camouflage-based active detection scheme
CBDS	Cooperative bait detection scheme
CDS	Centralized detection system
CHEMAS	Checkpoint-based multi-hop acknowledgment scheme
CMD	Monitor-based approach
CRS	Channel aware reputation system
DSR	Dynamic source routing
EAACK	Enhanced adaptive acknowledgment
EHNet	Energy harvesting motivated network
EH-LLN	Energy harvesting motivated lower power and lossy network
EYES	Proposed forwarding misbehavior detection scheme
FADE	Forwarding assessment based detection
HCD	Hop-by-hop cooperative detection
HED	Heuristic-based detection
IoTSN	Internet-of-things sensor network
LazyDog	Proposed monitor-based detection scheme
LLN	Low power and lossy network
MAC	Medium Access Control
MANET	Mobile ad hoc network
PFCB	Fiber composite bi-morph
SCAD	Single checkpoint-based countermeasure
SCM	Side channel monitoring
SCPC	Integrated self-charging power cell
SlyDog	Proposed Inducement-based detection scheme
SNBDS	Sequence number based bait detection scheme
WatchDog	Observation-based detection scheme
WSN	Wireless sensor network

[10] Bounouni M, Bouallouche-Medjkoune L. Acknowledgment-based punishment and stimulation scheme for mobile ad hoc network. The Journal of Supercomputing. 2018;74(10):5373–5398.

[11] Chang J, Tsou P, Woungang I, et al. Defending against collaborative attacks by malicious nodes in MANETs: A cooperative bait detection approach. IEEE Systems Journal. 2014;9(1):65–75.

[12] Jhaveri R, Patel N. A sequence number based bait detection scheme to thwart grayhole attack in mobile ad hoc networks. Wireless Networks. 2015;21(8):2781–2798.

[13] Pu C, Lim S. Spy vs. Spy: Camouflage-based Active Detection in Energy Harvesting Motivated Networks. In: Proc. Military Communications Conference (MILCOM) – Track 3. Cyber Security and Trusted Computing; 2015. .

[14] Samy A, Yu H, Zhang H. Fog-Based Attack Detection Framework for Internet of Things Using Deep Learning. IEEE Access. 2020;8:74571–74585.

[15] Baig Z, et al. Averaged dependence estimators for DoS attack detection in IoT networks. Future Generation Computer Systems. 2020;102:198–209.

[16] Sivanesh S, Dhulipala V. Accurate and cognitive intrusion detection system (ACIDS): a novel black hole detection mechanism in mobile ad hoc networks. Mobile Networks and Applications. 2020;p. 1–9.

[17] Lim S, Kimn J, Kim H. Analysis of Energy Harvesting for Vibration-Motivated Wireless Sensor Networks. In: Proc. International Conference on Wireless Networks; 2010. p. 391–397.

[18] 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).

[19] 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).

[20] Starner T. Human-powered Wearable Computing. IBM Systems Journal. 1996;35(3 & 4):618–629.

[21] Starner T, Paradiso JA. Human Generated Power for Mobile Electronics. In: CRC Press; 2004. p. 1–35.

[22] Wang ZL. Nanogenerators for Self-powered Devices and Systems. Georgia Institute of Technology, Atlanta, USA; 2011.

[23] Xue X, Wang S, Guo W, et al. Hybridizing Energy Conversion and Storage in a Mechanical-to-Electrochemical Process for Self-Charging Power Cell. Nano Letter. 2012;12(9):5048–5054.

[24] Eu ZA, Tan H, Seah WKG. Design and Performance Analysis of MAC Schemes for Wireless Sensor Networks Powered by Ambient Energy Harvesting. Ad Hoc Networks. 2011;9(3):300–323.

[25] Fujii C, Seah WKG. Multi-tier Probabilistic Polling for Wireless Sensor Networks Powered by Energy Harvesting. In: Proc. IEEE ISSNIP; 2011. p. 383–388.

[26] Pu C, Gade T, Lim S, et al. Light-Weight Forwarding Protocols in Energy Harvesting Wireless Sensor Networks. In: Proc. MILCOM; 2014. p. 1053–1059.

[27] Stallings W. Cryptography and Network Security - Principles and Practices, 6th Edition. Prentice Hall; 2013.

[28] Marti S, Giuli TJ, Lai K, et al. Mitigating Routing Misbehavior in Mobile Ad hoc networks. In: Proc. ACM MOBICOM; 2000. p. 255–265.

[29] Raymond DR, Midkiff SF. Denial-of-Service in Wireless Sensor Networks: Attacks and Defense. IEEE Pervasive Computing. 2008;7(1):74–81.

[30] Hai TH, Huh E. Detecting Selective Forwarding Attacks in Wireless Sensor Networks Using Two-hops Neighbor Knowledge. In: Proc. IEEE NCA; 2008. p. 325–331.

[31] Liu K, Deng J, Varshney PK, et al. An Acknowledgment-Based Approach for the Detection of Routing Misbehavior in MANETs. IEEE Trans on Mobile Computing. 2007;6(5):536–550.

[32] Shila DM, Yu C, Anjali T. Mitigating Selective Forwarding Attacks with a Channel-Aware Approach in WMNs. IEEE Trans on Wireless Communications. 2010;9(5):1661–1675.

[33] Li X, Lu R, Liang X, et al. Side Channel Monitoring: Packet Drop Attack Detection in Wireless Ad Hoc Networks. In: Proc. IEEE ICC; 2011. p. 1–5.

[34] Shakshuki EM, Kang N, Sheltami TR. EAACK: A Secure Intrusion-Detection System for MANETs. IEEE Trans on Industrial Electronicsg. 2013;60(3):1089–1098.

[35] Liu Q, Yin J, Leung V, et al. FADE: Forwarding Assessment Based Detection of Collaborative Grey Hole Attacks in WMNs. IEEE Trans on Wireless Communications. 2013;12(10):5124–5137.

[36] Ren J, Zhang Y, Zhang K, et al. Exploiting channel-aware reputation system against selective forwarding attacks in WSNs. In: Proc. IEEE Global Communications Conference; 2014. p. 330–335.

[37] OMNeT++ Documentation and Tutorials;. Http://www.omnetpp.org/documentation/.

[38] Boulis A. Castalia; 2014. Http://castalia.forge.nicta.com.au.

[39] Tang X, Xu J. Extending Network Lifetime for Precision-Constrained Data Aggregation in Wireless Sensor Networks. In: INFOCOM; 2006. p. 1–12.

[40] Pu C, Lim S, Jung B, et al. EYES: Mitigating Forwarding Misbehavior in Energy Harvesting Motivated Networks. Computer Communications. 2018;124(2018):17–30.

[41] Shila D, Cheng Y, Anjali T. Mitigating selective forwarding attacks with a channel-aware approach in WMNs. IEEE transactions on wireless communications. 2010;9(5):1661–1675.

[42] Pu C, Lim S. A Light-Weight Countermeasure to Forwarding Misbehavior in Wireless Sensor Networks: Design, Analysis, and Evaluation. IEEE Systems Journal. 2018;12(1):834–842.

[43] Fafoutis X, Dragoni N. ODMAC: An On-Demand MAC Protocol for Energy Harvesting – Wireless Sensor Networks. In: Proc. PE-WASUN; 2011. p. 49–56.

[44] Vithanage MD, Fafoutis X, Andersen CB, et al. Medium Access Control for Thermal Energy Harvesting in Advanced Metering Infrastructures. In: Proc. EuroCon - Internet Services and Applications; 2013. p. 291–298.

[45] Pu C. Sybil Attack in RPL-Based Internet of Things: Analysis and Defenses. IEEE Internet of Things Journal. 2020;7(6):4937–4949.

[46] Pu C, Lim S, Jung B, et al. Mitigating Stealthy Collision Attack in Energy Harvesting Motivated Networks. In: Proc. Military Communications Conference (MILCOM) - Track 3. Cyber Security and Trusted Computing; 2017. .

[47] Winter T, Thubert P. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks; 2012. RFC Standard 6550.

[48] Romdhani I, Qasem M, Al-Dubai A, et al.. Cooja Simulator Manual; 2016. Edinburgh Napier University.

[49] Abidin A, Aly A, Mustafa MA. Collaborative Authentication Using Threshold Cryptography. In: International Workshop on Emerging Technologies for Authorization and Authentication. Springer; 2019. p. 122–137.

[50] Feng Q, He D, Wang H, et al. Lightweight Collaborative Authentication With Key Protection for Smart Electronic Health Record System. IEEE Sensors Journal. 2019;20(4):2181–2196.

[51] Rimmer V, Preuveneers D, Joosen W, et al. Frictionless authentication systems: emerging trends, research challenges and opportunities. arXiv preprint arXiv:180207233. 2018;.

[52] Bailey DV, Duane WM, Katz A. Protected resource access control utilizing credentials based on message authentication codes and hash chain values. Google Patents; 2015. US Patent 8,984,602.

[53] Bailey DV, Duane WM, Young E. Protected resource access control utilizing intermediate values of a hash chain. Google Patents; 2015. US Patent 8,990,905.

[54] Alshahrani M, Traore I. Secure mutual authentication and automated access control for IoT smart home using cumulative keyed-hash chain. Journal of information security and applications. 2019;45:156–175.

[55] Pinto A, Costa RF. Hash-chain-based authentication for IoT. 2016;.

[56] He M, Fan P, Kaderali F, et al. Access key distribution scheme for level-based hierarchy. In: Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies; 2003. p. 942–945.

[57] Shehab M, Bertino E, Ghafoor A. Efficient hierarchical key generation and key diffusion for sensor networks. In: 2005 Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2005. IEEE SECON 2005.; 2005. p. 76–84.

[58] Castiglione A, Santis AD, Masucci B, et al. Hierarchical and Shared Access Control. IEEE Transactions on Information Forensics and Security. 2016 April;11(4):850–865.

[59] Zaman MU, Shen T, Min M. Hash Vine: A New Hash Structure for Scalable Generation of Hierarchical Hash Codes. In: 2019 IEEE International Systems Conference (SysCon); 2019. p. 1–6.

[60] Borgia E. The Internet of things vision: key features, applications and open issues. Journal of Computer Communications. 2014;.

[61] Al-Fuqaha A, Guizani M, Mohammadi M, et al. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Communications Surveys & Tutorials. 2015;(17):2347–2376.

[62] Weyrich M, Ebert C. Reference Architectures for the Internet of Things. IEEE Software. 2016;33(1):112–116.

[63] Breivold H. A Survey and Analysis of Reference Architectures for the Internet-of-Things. In: Proc. Software Engineering Advances (ICSEA); 2017. .

[64] Simmhan Y, Ravindra P, Chaturvedi S, et al. Towards a Data-driven IoT Software Architecture for Smart City Utilities. Software: Practice and Experience. 2018;48(7):1390–1416.

[65] Campeanu G. A Mapping Study on Microservice Architectures of Internet of Things and Cloud Computing Solutions. In: Proc. Mediterranean Conf. on Embedded Computing (MECO); 2018. .

[66] Alreshidi A, Ahmad A. Architecting Software for the Internet of Thing Based Systems. Future Internet. 2019;11(7).

[67] Mucchini H, Moghaddam MT. IoT Architectural Styles: A Systematic Mapping Study. In European Conference on Software Architecture. In: Proc. European Conf. on Software Architecture; 2018. .

[68] Gaglione A, Rodenas-Herraiz D, Jia Y, et al. A. Energy Neutral Operation of Vibration Energy-Harvesting Sensor Networks for Bridge Applications. In: Proc. Embedded Wireless Systems and Networks (EWSN); 2018. .