Detecting Denial-of-Service Attacks against Wireless Sensor Networks*

Steven Cheung Bruno Dutertre Ulf Lindqvist
Computer Science Laboratory
SRI International
Menlo Park, CA 94025
{steven.cheung,bruno.dutertre,ulf.lindqvist}@sri.com

April 21, 2005

Abstract

Because of resource and cost constraints, and of deployment scenarios in which a number of sensor nodes are spread in a large geographic region and are unattended, wireless sensor networks are especially vulnerable to denial-of-service (DoS) attacks. Moreover, sensor nodes are also used as routers to forward packets from other nodes. Compromised sensor nodes may be able to cause network-wide DoS by dropping, corrupting, and misrouting packets. This paper presents novel, lightweight approaches for detecting DoS caused by physical attacks and disruptive routers.

Keywords: Sensor networks, denial of service, intrusion detection

1 Introduction

Wireless sensor networks have certain characteristics that make them especially vulnerable to denial-of-service (DoS) attacks:

Resource constraints: Sensor nodes are typically resource constrained. In particular, sensor nodes may run on batteries. When the energy reserve is depleted, the sensor nodes may be rendered useless. An attack vector is to drain the energy reserve of sensor nodes. Also, limited memory, computation power, energy reserve, and network bandwidth of sensor nodes prevent resource-intensive security techniques from being employed in sensor networks.

Exposure: In a wired environment, network infrastructure components such as routers are dedicated nodes and are usually physically inaccessible to attackers. In a sensor network, nodes may be spread across a large geographic region, and may be in remote locations that are more likely to be subjected to physical attacks.

*This research is sponsored by DARPA under contract number F30602-02-C-0212 and National Science Foundation under contract number CNS-0434997. The views herein are those of the author and do not necessarily reflect the views of the supporting agencies.

Collaborative processing: Typically, the radios of sensor nodes are of short range and can only communicate directly with other nodes in the vicinity. Multihop routing is needed for sensor networks of nontrivial size. Sensor nodes are not only used as data sources, but also used as routers to forward messages from other nodes. A compromised sensor node may thus be able to affect communication of the entire sensor network.

Wood's and Stankovic's paper [14] presents an overview of the DoS threats against sensor networks such as signal jamming. This paper examines and presents techniques to detect the following two DoS attacks: physically attacking or capturing sensor nodes to cause denial of service, and compromising sensor nodes to disrupt routing.

Section 2 presents a cooperative network monitoring approach for detecting DoS caused by physical attacks. Because of the exposure of sensor nodes, sensor networks are particularly vulnerable to physical attacks. Tamperproofing sensor nodes may not be feasible in many applications because of cost, size, and weight constraints. Our approach involves mutual monitoring among neighbor nodes to detect these attacks, and is resource efficient because it requires only local communication and computation among neighbor nodes.

Section 3 presents a lightweight scheme for detecting disruptive routers in sensor networks. These routers can prevent packets from reaching their destinations intact. A main challenge for disruptive router detection in sensor networks is to address the severe resource constraints for the sensor nodes. Our scheme is based on probabilistically sending "hints", which contain packet digests and local routing information, via different paths (with respect to the ones used for routing the corresponding packets) to reach the destination. The distinguishing characteristics of our scheme include its low runtime overhead for routers—which make it applicable to sensor networks—and its ability to detect various routing misbehavior such as dropping, misrouting, or corrupting packets.

2 Detecting DoS caused by Physical Attacks

Because sensor nodes may be deployed over a large geographic region, and may be unattended, the threats of physical security are more critical in sensor networks than in conventional wired networks. Moreover, in many applications, because of the size, cost, and weight constraints for sensor nodes, and the need for the sensor nodes to be located close to the monitored phenomena, it is virtually impossible to deploy strong tamperproof measures such as enclosing sensor nodes in a vault with thick steel walls.

Examples of physical attacks include turning off sensor nodes, moving sensor nodes to a distant location, and destroying sensor nodes using physical, electrical, or chemical means. An adversary may also extract sensitive information from sensor nodes such as keying materials, or replace programs or data stored in sensor nodes with incorrect ones.

Consider a scenario in which an adversary attempts to disable a sensor network by substituting sensor nodes with nodes that exhibit the same behavior except that they generate incorrect sensor reports. We argue that it is difficult, if not impossible, to prevent this kind of attack. Although one may set up a cryptographic key between a sensor node and the base station for message authentication, the adversary may first compromise the sensor node to obtain its key and disable

it, and then instrument a substitute node with the key to emulate the original sensor node. Because the adversary may perform machining to remove coating materials and use imprinting techniques to retain the content of RAM after powerdown [13], protecting the confidentiality of keying materials stored in sensor nodes for an extended period of time may not be feasible. In the following, we present a network monitoring approach for detecting these attacks.

2.1 A Mutual Monitoring Approach

In this approach, called *NeighborWatch*, sensor nodes cooperatively monitor each other for availability. The basic idea is that every sensor node first establishes neighborhood relationships with other nodes; two nodes are *neighbors* if there exists a high-quality bidirectional communication link between them. After that, sensor nodes exchange authenticated heartbeat messages with their neighbors. If a node fails to receive (fresh) heartbeats from one of its neighbors for a specified period of time, say t_d, it will generate an alert message and send it to the base station (or a designated node), which performs alert correlation and response.[1] This scheme has three variants, depending on the key(s) used for message authentication:

Group keying: All nodes use a shared secret key to authenticate messages from other nodes, a keying scheme used in TinySec [8].

Node-specific keying: Every sensor node generates a secret (symmetric) key and shares it with its neighbors.

Link-specific keying: Every sensor node performs key distribution to set up a secret key to secure communication with each of its neighbors, for example, using [2, 4, 5]. In other words, every link between two neighbor nodes has its own key.

We assume that sensor nodes are equipped with certain tamper-resistant security measures such as coatings to prevent probing attacks [13]. An adversary who wants to extract the keying materials stored on a sensor node will need to destroy or disassemble the node, and it will take at least t_a time units to bring up a substitute node with the keying materials. Moreover, the sensing and other functions pertaining to the sensor node will not be performed during this time interval. We also assume that the detection time interval t_d is less than t_a.

2.2 Discussion

Let us consider a series of physical attacks in the increasing order of sophistication. The first attack, also described in the abovementioned scenario, involves destroying sensor nodes and replacing them with nodes that generate incorrect sensor reports. This attack can be countered by the NeighborWatch scheme because the sensor reports generated by the substitute nodes (without the keys to generate authenticated heartbeat messages) will not be accepted by the other nodes in the sensor network. The adversary needs to attack all nodes in the connected subnet of the sensor network, or the neighbors of the attacked nodes will detect the attack and generate an alert.

[1] An example automated response action is to distribute a new group key to nodes other than the detected missing ones to secure future communication for the sensor network.

In the second attack, the adversary captures a sensor node, extracts the keying data stored in the node, and instruments substitute nodes with the keying data. When a set of sensor nodes is attacked, the attack is detected and reported by the neighbors of these nodes that are intact, if $t_d < t_a$. A strategy to extend the attack time window without triggering an alert is to take out nodes in sequence: After attacking a node, say v, the adversary will have t_d before its neighbors will detect the missing node. If the adversary can take out these neighbors before t_d, these nodes will not be able to generate an alert. In general, the adversary may extend the attack time window to $(k+1)t_d$ if she could take out all nodes that are h hops from v, where $1 \leq h < k$, ht_d time units after v is attacked.

Depending on the topology of the sensor network and the relative values of t_d and t_a, the group keying variant may be able to detect the attack. Consider a linear network topology in which each node v_i has two neighbors, namely v_{i-1} and v_{i+1}, except the first and last nodes. (A similar topology is a ring in which all nodes have two neighbors.) Assume that $t_a < (j+1)t_d$. The adversary can thus prevent the group keying variant of NeighborWatch from generating alerts by attacking v_i to steal the group key at time t, and replacing v_{i-j} and v_{i+j} with substitute nodes equipped with the group key at time $t + t_a$. In a denser sensor network in which every node is watched by a number of other nodes, it may be more difficult to perform this attack. For example, consider a sensor network that has a two-dimensional grid topology in which there is a node at (x, y), where x and y are integers. Moreover, Node $v_{(x,y)}$ can communicate directly with eight neighbors $v_{(r,c)}$, where $r \in \{x-1, x, x+1\}$, $c \in \{y-1, y, y+1\}$, and $(r, c) \neq (x, y)$. In this topology, the number of nodes that the adversary needs to attack is proportional to the square of t_a/t_d. The second attack is effective against the node-specific keying variant if $t_a < 2t_d$, because compromising a node may reveal the keys used by its neighbors, and enables the adversary to deploy substitute nodes to emulate them. The link-specific keying scheme is the most secure among the three variants because compromising a node reveals the keying data only for that node. However, there are additional communication costs associated with the link-specific variant: for the other two variants, a node can generate an authenticate heartbeat message and broadcast it to its neighbors. On the other hand, for the link-specific keying variant, a node needs to generate and send a different heartbeat message to each of its neighbors.

3 Detecting Disruptive Routers in Sensor Networks

To detect disruptive nodes that exhibit various routing misbehaviors, such as maliciously dropping packets and unauthorized packet modification, which may cause denial of service, we present a "hint-based" approach. In this approach, as a packet traverses the network from its source to the base station (typically, a significantly more resourceful node than the sensors), the source and every intermediate node probabilistically send a hint of the packet to the destination. Moreover, the hint specifies a set of nodes that should not be used for routing the hint. The basic idea is that if the next-hop sensor for the sensor packet is N (as determined by the underlying routing protocol), the hint should not be routed through N. If N misbehaves, it may not affect both the sensor packet and the hint. Based on the packets and hints received, the base station may detect misbehaving nodes in the network. Our approach is lightweight for routing nodes, and thus it is particularly

applicable to sensor networks, whose nodes are typically resource constrained.

3.1 Assumptions

1. *Communication pattern:* The communication pattern involves sensors (sources) sending packets to a base station (sink).

2. *Powerful base station:* We assume that the base station is a resource-rich node. For example, a base station may be a regular notebook computer.

3. *Key distribution:* Every node shares a (secret) symmetric key with the base station.

4. *Alternate path:* There is no node in the network whose removal can disconnect the network (i.e., no cut vertex).

5. *Known distance from base station:* The base station knows the distance between any sensor and itself. (Depending on the routing protocol used, this information may be computed from the routing updates.)

6. *Bounded message delay:* The total delay (including propagation, queueing, and processing delays) for a packet to travel from its source to the sink is bounded by α. (Thus, when the sink receives a sensor or hint packet, it can compute the time that it can safely discard the packet.)

3.2 Hint-based Detection

Figure 1 depicts an example of our approach: Sensor A forwards a packet p to the base station S, and N is the next-hop sensor (as determined by the underlying routing protocol). With a specified probability, A will also send a hint h, which will not be routed through N. Thus, if the node N misbehaves, it cannot affect both the packet and the hint. If S receives h but not p, it may increase the suspicion that N is a disruptive router.

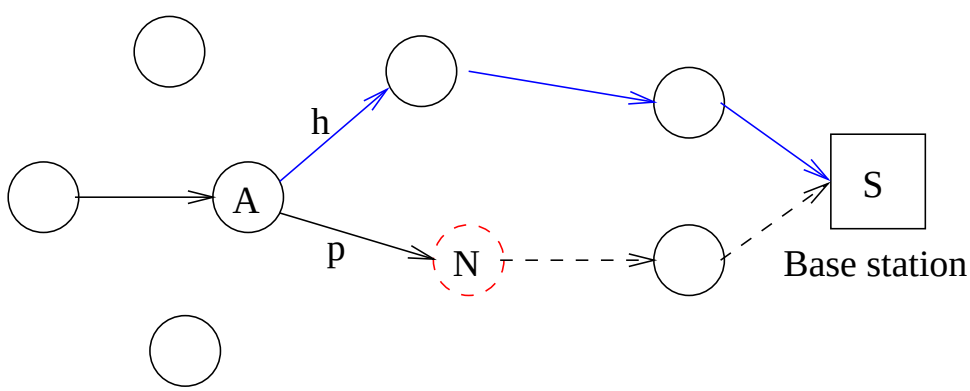

Figure 1: Hint-based Detection Approach: An Example

When a packet traverses the network from its source to the base station, the source and every intermediate sensor may probabilistically send a *hint* to the base station. The hint contains (1) a digest of the packet, used by the destination to verify whether the corresponding packet reaches its destination and to verify its integrity, (2) its local routing information such as the next-hop router, and (3) a set of nodes that should not be used for routing the hint (in the protocol we will describe in this section, the set contains just the next-hop neighbor). The base station may increase the suspicion that a node X is misbehaving if (1) the base station receives a hint but not the corresponding packet, and (2) X is in the exclusion list of the hint. Based on the packets and the hints received, the base station detects and locates misbehaving routing nodes.

3.3 Hint Generation

Suppose the next hop for sending a packet from a node A to the base station (sink) is N. When node A receives a packet p, A will forward it to N. Moreover, if N is not the base station, then A, with probability δ, will send an authenticated *hint* to a neighbor other than N. The purpose of the hint is to test whether N is a misbehaving node. The hint contains the identities of A and N, and a packet identifier sid. The structure of the hint is

$$\langle A, N, sid_p, G_{k_{as}}(A, N, sid_p)\rangle,$$

where G is a keyed one-way hash function, and k_{as} is a shared symmetric key between A and the base station S. Note that a node does not generate a hint for a hint that it receives.

The packet identifier is used to associate the hint with the packet p. In the simplest case, one may combine the source node id and a sequence number as the packet identifier. In this paper, we do not assume that packets have explicit fields for the sender's id and the sequence number, which may not be present in the message format of existing systems. Instead, we use a shared cryptographic hash function H to generate packet identifiers. Specifically, A uses the concatenation of seq_{AN} and $H(p)$ as the packet identifier for packet p, where seq_{AN} is a sequence number denoting the number of packets forwarded from A to N. Note that instead of using seq_{AN}, one may use a random number to serve as a nonce to counter replay attacks.

To prevent a malicious node from acting only on packets that do not have an associated hint to avoid detection, we use two techniques to obfuscate the association: encryption and random time delay. When a node sends a hint, it encrypts the packet identifier using a secret key it shares with the base station. Thus, the only nodes that can link the hint and the packet using the content of the hint are the hint-generating node and the base station. Moreover, a random delay is introduced between the sending times of a packet and the associated hint. This is used to prevent a malicious node from inferring the relationship between the packet and the hint by using its send time.

3.4 Hint Routing

If a misbehaving node can affect both a packet and the corresponding hints, one may not be able to detect it. In our protocol, hints are sent using a special communication primitive $sendx(D, L, p)$, where D is the destination, L is a set of intermediate nodes to avoid in the route, and p is the packet to send. For our hint-based detection protocol, A sends the hint h to the base station S using

$sendx(S, \{N\}, h)$. For certain routing protocols, realizing $sendx()$ requires no or minor adaptation of the underlying routing protocol. We will describe several such algorithms and the adaptation needed, if any.

For constraint-based routing [11], one may specify node exclusion using a *route constraint*. When route discovery is performed to find a path from Node A to the base station, the route constraint for excluding node N will be used to guide the search process.

In [6], multipaths are used to increase the robustness of routing. In particular, the authors present heuristics for constructing disjoint paths and braided paths (which are partially disjoint paths) for directed diffusion-type routing schemes. These multipaths may be used to realize $sendx()$ by routing p through N and routing h through an alternate path.

For routing protocols such as [7], a data sink broadcasts a beacon to its direct neighbors, which will in turn broadcast it to their neighbors, and so on. Each node will compute its "levels" (measured by the number of hops from the sink). A level-(k+1) node may send a packet to the sink by forwarding it to a level-k node, or by forwarding it to its sibling (i.e., another level-(k+1) neighbor node). The node that receives the packet will then repeat the same procedure until the packet reaches its destination. These routing protocols can be adapted to implement hint routing. Assume that A is in level-(k+1), and its next-hop neighbor N is in level k. If A has two level-k neighbors, A can forward p to N and forward h to the other level-k node with an attached constraint that it should not route h to N. Otherwise, A may probabilistically forward h to a level-(k+1) node with an attached constraint that it should not forward the packet to N. That node will then either forward h to a level-k node that is not N or probabilistically forward it to another level-(k+1) neighbor node. A "time-to-live" counter may be attached to the hint packet to ensure that it will either reach its destination or be dropped after a specified number of hops.

3.5 Diagnosis

Based on the packets and hints received, the base station attempts to detect and to locate misbehaving routers. Here, we present a diagnosis procedure to identify misbehaving routers in sensor networks.

The base station should never rely on single packets or single sources to draw a conclusion. First, packets may be lost because of a benign cause such as collisions and traffic congestion. Second, it is possible that a malicious node may attempt to "frame" a good node X by just sending a hint with X in the exclusion list. Third, a packet may be discarded by any nodes on the path. If the base station receives a hint sent by A (for testing N) but not the corresponding packet, it may be caused by any node on the path from N to the base station. As a result, the base station should depend on multiple tests/events and multiple sources to detect and to locate misbehaving nodes.

The base station constructs a directed graph, called *accusation graph*, $G = (V, E)$, where V corresponds to the set of sensors in the network, and an edge $(P, Q) \in E$ corresponds to an *accusation* from P to Q. The base station computes these accusations based on the packets it received during the last period.

Let P be a sensor node in the network. Suppose the base station received n_2 distinct hints sent by P for testing Q (i.e., P forwarded the corresponding packets to Q), and received n_1 distinct

packets that match those hints. Two thresholds, t_1 and t_2, define the edges of an accusation graph. The base station creates an edge (P, Q) in G if the following conditions are met:

- $(n_2 - n_1) > t_1$
- $n_1/n_2 < t_2$

Threshold t_1 ensures that the difference between the number of hints and the number of corresponding packets is considered "significant" enough for disruptive node detection. Threshold t_2 concerns what the "acceptable" ratio on the number of hints and the corresponding packets pertaining to a node that can reach the sink is.

Based on the accusation graph, the diagnosis procedure computes a set of suspicious nodes V_s. Specifically, V_s is the set of nodes that have an in-degree $> t_{acc}$ in G, where t_{acc} is a threshold number of accusations needed to infer that a node is a disruptive node. We choose t_{acc} as the maximum number of disruptive routing nodes in the network plus one to prevent the disruptive nodes from conspiring to cause a good node to be misclassified as bad. A good node may appear in V_s because there exists a disruptive node downstream (i.e., closer to the sink). Our protocol avoids that problem by finding a node in V_s that is closest to the base station, and outputs that node as a disruptive routing node.

3.6 Simulation

To study the detection and false positive rates of the hint-based detection scheme, we have conducted simulations using JProwler [12], a sensor network simulation package.

The simulation consists of the following steps:

1. Every node exchanges messages with other nodes to identify its neighbors.
2. The base station initiates a flooding procedure to compute the distances between the base station and every node in the network. In this step, the base station broadcasts a message advertising to its neighbors that it is zero hop from the base station. When its neighbors receive this message, they increment the hop count and advertise to their neighbors that they are one hop from the base station, and so on.
3. A set of c source nodes is selected randomly from the network. Each source sends a report to the base station at a fixed period p. After r reports are sent, the source stops.
4. When a normal node receives a sensor report, it selects a next-hop neighbor probabilistically and with probability δ sends a hint to the sink. Because communication links in sensor networks tend to be lossy, an "implicit acknowledgement" scheme is used to increase routing robustness. When a node A forwards a packet to a node B, A monitors B's activities to see if B forwards the packet; if not, A retransmits the packet to B (up to a retransmission threshold). We call the disruptive nodes in the network "bad" nodes. When a bad node receives a packet, the node drops it instead of forwarding it to the sink.

t_1	TP	FN	FP	TN
25	0	0	4	96
50	0	0	2	98
75	0	0	0	100
100	0	0	0	100
125	0	0	0	100

t_1	TP	FN	FP	TN
25	53	0	1	46
50	49	3	2	46
75	47	8	2	43
100	32	25	3	40
125	25	34	0	41

Table 1: Varying t_1: No bad node (left) and one bad node (right)

5. Based on the sensor reports and hints received, the sink performs the diagnosis procedure described in Section 3.5 to identify a bad node.

If the protocol incorrectly (correctly respectively) identifies a node as bad, we call this a false positive (FP) (true positive (TP) respectively). If the bad node(s) removes more than a fraction f of the sensor reports, and the protocol cannot identify a bad node, we call this a false negative (FN). If there is no bad node in the network, or the fraction of the sensor reports removed by the bad nodes is less than f, and the hint-based protocol does not identify any node as bad, we call this a true negative (TN).

We have conducted a series of experiments to study the effects of varying t_1, p, and the number of bad nodes in the network. In the experiments, we use a sensor network of size 100. The network lives in a 100×100 grid, and consists of the sink, 10 nodes that generate sensor reports and act as routers, 0 to 4 bad nodes, and 95 to 99 nodes that act as routers only. The sink is located at coordinate (10,10). Moreover, the bad nodes, if they exist, are placed at the following coordinates: the first bad node at (25,25), the second at (50,50), the third at (25,50), and the fourth at (50,25). The rest of the nodes are randomly distributed in the grid.

3.6.1 Varying t_1

This experiment examines the effects of changing the threshold t_1. Recall that t_1 corresponds to a threshold for the difference between the number of hints sent by a node for diagnosing a particular neighbor that reach the sink and the number of corresponding packets reaching the sink intact. In this experiment, we use the following parameters[2]: r=1000, p=2500 ms, c=10, t_2=0.5, f=10%, δ=5%. For each value of t_1, ranging from 25 to 125, we performed the experiment 100 times for the cases in which there is no bad node and one bad node, respectively. The simulation results are summarized in Table 1. The results agree with the intuition that one may achieve a higher true positive rate by selecting a small value for t_1. But, choosing a small value for t_1 might cause more false positives.

[2]Recall that r denotes the number of reports sent by each report-generating sensor node, p the period for report generation, c the number of sensors that generate reports, t_2 a diagnosis threshold for the ratio of hints and corresponding packets received by the sink for diagnosing a node, f the fraction the reports removed by bad nodes considered to be significant for the purpose of disruptive router detection, and δ the probability for sending a hint when a sensor node receives a (non-hint) packet.

$p(ms)$	TP	FN	FP	TN
250	0	0	0	100
500	0	0	0	100
1000	0	0	1	99
2000	0	0	1	99
5000	0	0	0	100

$p(ms)$	TP	FN	FP	TN
250	24	15	0	61
500	22	6	2	70
1000	36	7	2	55
2000	41	2	3	54
5000	47	4	0	49

Table 2: Varying p: No bad node (left) and one bad node (right)

Num bad nodes	TP	FN	FP	TN
0	0	0	1	99
1	36	7	2	55
2	42	16	0	42
3	41	42	3	14
4	47	46	1	6

Table 3: Varying the number of bad nodes

3.6.2 Varying p

This experiment studies the effect of changing the period of sensor report generation. The following values are used for this experiment: r=1000, c=10, t_1=50, t_2=0.5, f=10%, δ=5%. For each value of p, ranging from 250 ms to 5000 ms, we performed the experiment 100 times for the cases in which there is no bad node and one bad node, respectively. The results are summarized in Table 2. A higher report generation rate may increase the probability of packet collision and network congestion. Thus, the smaller the value of p, the more difficult it is to correctly detect bad nodes.

3.6.3 Varying the number of bad nodes

This experiment studies the more difficult problem of multiple bad nodes in the network. The following parameters are used: r=1000, p=1000 ms, c=10, t_1=50, t_2=0.5, f=10%, δ=5%. The results are summarized in Table 3. Having more bad nodes may increase the probability that the fraction of packets dropped by them is greater than f. Thus, as the number of bad nodes increases, the number of true negatives decreases. Also, the number of packets removed by an individual bad node may decrease with the increase in the number of bad nodes, making it more difficult to detect them.

3.6.4 Remarks

The simulation results show that the hint-based approach can be quite effective in identifying nodes that maliciously drop packets and has a low false positive rate, under the conditions of modest network traffic volume and a small number of bad nodes. As for future work, there are several directions we plan to pursue, including developing scalable disruptive router detection techniques

and protocols for large sensor networks, improving the performance of the detection protocol for high-volume traffic environments, and implementing and evaluating the detection protocol on an existing sensor network platform such as Mica motes running TinyOS.

3.7 Related Work

Existing techniques for detecting disruptive routers in a network are either not suitable for sensor networks or have limitations. In [9], the authors propose a protocol for detecting routers that fail to forward packets in wireless networks. By exploiting the broadcast nature of wireless networks, a node may detect whether a neighbor fails to forward packets by monitoring the packets it receives and the packets it forwards. This technique has weaknesses. When node A forwards a packet to node B, the packet may not be received by B (e.g., because of packet collision or weak signal strength). Thus, the technique may fail in some cases. Our approach does not suffer from these problems because the detection and diagnosis are performed at the end point.

In [1, 3], a technique based on conservation of flow is used for detecting routers that drop packets to cause DoS. Basically, the neighbors of a node maintain counters to keep track of the amount of traffic going into and going out of a node, and they exchange information by flooding to determine whether the node drops many packets. A possible attack not addressed by the flow conservation technique involves a malicious node modifying packets in transit. Because the amount of traffic flow is not changed by this attack, the malicious node will not be detected by this technique. On the other hand, our approach can detect unauthorized packet modification by means of cryptographic checksums.

In [10], a secure traceroute-based approach has been proposed to detect and locate the source of routing misbehavior. The scheme assumes that any node can establish a secret key with any other nodes in the network (e.g., using a public-key infrastructure or a PGP-style web-of-trust technique). This assumption may not hold for sensor networks because of the resource constraints. Furthermore, our approach is lightweight for sensors; most of the detection and diagnosis work is performed by the resourceful base station.

4 Conclusion

Denial-of-service threats are more pronounced for wireless sensor networks because sensor nodes are resource constrained, the nodes and the communication medium are exposed and unattended, and these networks rely on the sensor nodes to route packets. We have presented techniques for detecting two important types of DoS attack against sensor networks—physical attacks against sensor nodes and compromising nodes to disrupt routing.

Our approaches are particularly suitable for sensor networks because they are lightweight. In particular, the NeighborWatch scheme requires only local communication and computation among neighbors for monitoring the availability of sensor nodes. In the hint-based detection scheme, we exploit the asymmetry in resources between sensor nodes and the base station, and shift most of the detection burden to the more resource-rich base station. The scheme is lightweight for sensor nodes, and the more computation- and memory-intensive operations are performed by the base station.

Finally, our techniques may also be applicable to other attacks. The NeighborWatch scheme may be used to detect resource exhaustion attacks against sensor nodes—for example, sending many requests to a sensor node to drain its energy reserve—symptomatically.

References

[1] K. A. Bradley, S. Cheung, N. Puketza, B. Mukherjee, and R. A. Olsson. Detecting disruptive routers: A distributed network monitoring approach. *IEEE Network*, Sept./Oct. 1998.

[2] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 197–213, Berkeley, CA, May 11-14, 2003.

[3] S. Cheung and K. N. Levitt. Protecting routing infrastructures from denial of service using cooperative intrusion detection. In *Proceedings of the New Security Paradigms Workshop*, Sept. 23-26, 1997.

[4] B. Dutertre, S. Cheung, and J. Levy. Lightweight key management in wireless sensor networks by leveraging initial trust. Technical Report SRI-SDL-04-02, System Design Laboratory, SRI International, Apr. 2004.

[5] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 41–47, Nov. 2002.

[6] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *ACM Mobile Computing and Communication Review (MC2R)*, 1(2), 2002.

[7] C. Karlof, Y. Li, and J. Polastre. ARRIVE: Algorithm for robust routing in volatile environments. Technical Report UCB//CSD-03-1233, Department of EECS, University of California at Berkeley, Mar. 2003.

[8] C. Karlof, N. Sastry, and D. Wagner. TinySec: A link layer security architecture for wireless sensor networks. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, Nov. 2004.

[9] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the* 6th *Annual International Conference on Mobile Computing and Networking (MOBICOM 2000)*, pages 255–265, Boston, MA, Aug. 2000.

[10] V. N. Padmanabhan and D. R. Simon. Secure traceroute to detect faulty or malicious routing. In *ACM SIGCOMM Workshop on Hop Topic in Networks (HotNets-I)*, Princeton, NJ, Oct. 2002.

[11] Y. Shang, M. P. Fromherz, Y. Zhang, and L. S. Crawford. Constraint-based routing for ad-hoc networks. In *Proceedings of International Conference on Information Technology: Research and Education (ITRE '03)*, pages 306–310, Newark, NJ, Aug. 10-13, 2003.

[12] G. Simon, P. Völgyesi, M. Maróti, and Á. Lédeczi. Simulation-based optimization of communication protocols for large-scale wireless sensor networks. In *2003 IEEE Aerospace Conference*, Big Sky, MT, Mar. 8, 2003.

[13] S. H. Weingart. Physical security devices for computer subsystems: A survey of attacks and defenses. In Ç. K. Koç and C. Paar, editors, *Proceedings of the Second International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2000)*, volume 1965 of *LNCS*, pages 302–317, Worcester, MA, Aug. 17–18 2000. Springer-Verlag.

[14] A. D. Wood and J. A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, Oct. 2002.