Denial-of-Service in Wireless Sensor Networks: Attacks and Defenses

This survey of denial-of-service threats and countermeasures considers wireless sensor platforms' resource constraints as well as the denial-of-sleep attack, which targets a battery-powered device's energy supply.

David R. Raymond and Scott F. Midkiff
Virginia Tech

Continued research into using wireless sensor networks (WSNs) for medical monitoring, homeland security, industrial automation, and a variety of military applications highlights the need to better secure these networks. Just as researchers have developed new networking protocols to account for the limited resources available to WSN platforms, we must also tailor security mechanisms to such resource constraints. In particular, we must address the denial-of-service attack, which targets service availability.

Computer and network security aim to provide confidentiality, data integrity, and service availability. Confidentiality prevents untrusted third parties from accessing secure data, and data integrity guarantees that data isn't modified in transit and that replayed packets aren't accepted as the original. Availability ensures that authorized parties can access data, services, or other computer and network resources when requested. DoS attacks target availability by preventing communication between network devices or by preventing a single device from sending traffic.

Anthony Wood and John Stankovic published a survey of WSN DoS attacks and prevention mechanisms in 2002.[1] Here, we update their survey with current threats and countermeasures. In particular, we more thoroughly explore the *denial-of-sleep* attack, which specifically targets the energy-efficient protocols unique to sensor network deployments. We start by exploring such networks' characteristics and then discuss how researchers have adapted general security mechanisms to account for these characteristics.

Wireless-sensor-network characteristics

WSN platforms generally have limited processing capability and memory. The design of WSN devices usually favors decreased cost over increased capabilities, so we can't expect Moore's law to lead to enhanced performance. The basic characteristics of sensor networks make them vulnerable to DoS attacks.

Their primary weakness, shared by all wireless networking devices, is the inability to secure the wireless medium. Any adversary in radio range can overhear traffic, transmit spurious data, or jam the network. Powerful antennas allow remote access, so close physical proximity to the network isn't required.

Sensors are also vulnerable to physical tampering and destruction if deployed in an unsecured area. Another vulnerability is the sensor devices' extremely limited and often nonreplenishable power supplies. Resource-consumption

Published by the IEEE CS ■ 1536-1268/08/$25.00 © 2008 IEEE

attacks target nodes' power supplies by keeping the radio on when there's no legitimate network traffic or by imposing an unnecessary computational load.

Furthermore, attackers aren't always limited by the same constraints as the sensor devices. An adversary might have a virtually unlimited power supply, significant processing capability, and the capacity for high-power radio transmission.

Here, we primarily consider small, inexpensive, resource-constrained sensor platforms such as the Crossbow Mica2 and TMote Mini. Both are configured to run for a year or more on a pair of AA batteries, relying on long periods of sleep to save power. The dominant source of power loss in these platforms is the radio subsystem.[2] Table 1 provides some basic configuration and power consumption information for these devices.

TABLE 1
Sensor-platform power consumption and resource data.

Characteristic	Mica2*	TMote Mini†
RAM (Kbytes)	4	10
Program flash memory (Kbytes)	128	48
Maximum data rate (Kbps)	76.8	250
Power draw: Receive (mW)	36.81	57.0
Power draw: Transmit (mW)	87.90	57.0
Power draw: Sleep (mW)	0.048	0.003

* Data from www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf
† Data from www.sentilla.com/pdf/eol/Tmote_Mini_Datasheet.pdf

General sensor network security mechanisms

Some of the earliest research into encryption and authentication for resource-constrained sensor nodes resulted in a suite of protocols called Security Protocols for Sensor Networks.[3] SPINS provides broadcast authentication, two-party authentication, and data confidentiality using symmetric cryptography. Symmetric cryptography is better suited than public-key cryptography for sensor platforms' limited resources because it generally uses shorter encryption keys and requires less computation.

The SPINS protocol suite also supports data freshness for unicast messages, using packet counters to identify replayed packets. To reduce energy-consumption overhead, SPINS doesn't transmit counters with packets. When a node must send a unicast packet, it increments the antireplay counter associated with the packet's destination and calculates a message authentication code, which it transmits with the packet. When the destination node receives the packet, it increments its local copy of the packet counter for that sender and calculates its own authentication code. Because both communication partners increment the counter, this should successfully authenticate valid packets. If authentication fails, the destination node drops the packet. Packet loss, however, disrupts counter synchronization and requires an expensive recovery process.

Furthermore, SPINS requires each node to store a secret key and an antireplay counter for every node with which it might communicate. The memory requirements for storing this information make it unrealistic in memory-constrained sensor nodes, even in a moderately sized network of 25 nodes.[4]

TinySec is another security mechanism designed specifically for sensor networks.[4] It supports both packet authentication and encryption using symmetric cryptography. More important, it's included with the current release of TinyOS version 1.1, a widely used sensor-network operating system (www.tinyos.net). TinySec supports network-wide, cluster-wide, and pairwise encryption keys, and overhead is relatively low. Authentication increases per-packet power consumption by only 3 percent; authenticated encryption increases it by 10 percent. However, TinySec doesn't protect against message replay or provide specific protection against resource consumption attacks.

Some newer sensor platforms, such as the Tmote Sky, use transceivers that meet the IEEE 802.15.4, or ZigBee, specification.[5] ZigBee details physical and medium-access-control (MAC) layer requirements for wireless radios designed for personal-area-network devices and wireless sensor nodes. It provides hardware support for data confidentiality and integrity in compliant devices, mandating the use of Advanced Encryption Standard encryption. AES is a state-of-the-art symmetric-cryptography protocol that ZigBee uses to provide access control, data encryption, and packet authentication. Antireplay counters support data freshness, but they're optional (according to the standard).

As Naveen Sastry and David Wagner point out, designers must take care to avoid insecure implementations when designing devices based on this specification.[6] The ZigBee standard uses the antireplay-counter value transmitted with each packet as the encryption nonce (replay counter) for that packet. Many applications that rely on broadcast communication use cluster-wide or network-wide encryption keys to encrypt data. The ZigBee specification has nodes maintain security data, including encryption keys and antireplay counters, in an *access control list*, usually with one entry per communication partner. If a node places the same encryption key in multiple ACL entries,

TABLE 2
Denial-of-service attacks and defenses by protocol layer.

Protocol layer	Attacks	Defenses
Physical	Jamming	Detect and sleep Route around jammed regions
	Node tampering or destruction	Hide or camouflage nodes Tamper-proof packaging
Link/MAC (medium access control)	Interrogation	Authentication and antireplay protection
	Denial of sleep	Authentication and antireplay protection Detect and sleep Broadcast attack protection
Network	Spoofing, replaying, or altering routing-control traffic or clustering messages	Authentication and antireplay protection Secure cluster formation
	Hello floods	Pairwise authentication Geographic routing
	Homing	Header encryption Dummy packets
Transport	SYN (synchronize) flood	SYN cookies
	Desynchronization attack	Packet authentication
Application	Overwhelming sensors	Sensor tuning Data aggregation
	Path-based DoS	Authentication and antireplay protection
	Deluge (reprogramming) attack	Authentication and antireplay protection Authentication streams

that node might transmit multiple packets with the same nonce and encryption key. This is because each ACL entry maintains its own nonce state. If this happens, an attacker can `xor` the two ciphertexts to determine the `xor` of the plaintexts, potentially breaking confidentiality.[6] Sastry and Wagner detail how to avoid this *same-nonce* attack—and present other security weaknesses that should be avoided in future revisions to the IEEE 802.15.4 standard.[6]

The security primitives that SPINS, TinySec, and ZigBee provide, such as encryption, authentication, and, in some cases, antireplay, are the building blocks of many of the DoS prevention techniques we discuss next.

DoS attacks and defenses

For this discussion, we reduce the Open System Interconnect model's traditional seven layers to five layers: physical, link, network, transport, and application. Although sensor networks don't generally adhere as closely to the OSI model as other network devices for efficiency reasons, the layered model is still useful for categorizing various DoS attacks and defenses (see table 2). Some DoS attacks focus on physical aspects of sensor systems, such as covering a node with an acoustic barrier to reduce sensitivity. We focus primarily on attacks that exploit weaknesses in network protocols and applications, although we also mention techniques for preventing physical tampering and for mitigating sensor overstimulation.

The physical layer

Jamming is the primary physical-layer attack against WSNs. Spread-spectrum communication is a common defense against physical-layer jamming in wireless networks. Unfortunately, low-power, low-cost sensor nodes are usually limited to simple radios that can't use these techniques. If WSN nodes can identify a jamming attack, a logical defense is to put sensors into a long-term sleep mode and have them wake periodically to test the channel for continued jamming. Although this won't prevent a DoS attack, it could significantly increase the life of sensor nodes by reducing power consumption. An attacker would then have to jam for a considerably longer period, possibly running out of power before the targeted nodes do.

Wenyuan Xu and his colleagues provide a mechanism for identifying jamming attacks in WSNs, classifying them as *constant*, *deceptive*, *random*, or *reactive*.[7] A constant jamming attack corrupts packets as they are transmitted between WSN nodes. However, this attack requires a significant amount of energy and thus might not be feasible if the attacker is under similar power constraints as the target network.

Instead of transmitting a random signal, a deceptive jammer sends a constant stream of bytes into the network to make it look like legitimate traffic. For example, in TinyOS, if the device receives a constant stream of preamble bytes, all nodes within transmission range will remain in receive mode, never transitioning to send mode.

A random jammer randomly alternates between sleep and jamming to save energy.

Finally, a reactive jammer only transmits a jam signal when it senses traffic. Identifying reactive jamming can be difficult, because it might seem like routine packet collisions.

Techniques for identifying jamming attacks include statistically analyzing the received signal strength indicator (RSSI) values, the average time required to sense an idle channel (carrier sense time), and the packet delivery ratio (PDR).[7] All three techniques require taking baseline measurements, so the network can't be jammed upon deployment. None of these techniques alone is sufficient to identify jamming. However, algorithms that combine these techniques can reliably identify all four types of jamming. One such algorithm first identifies poor link utility through PDR analysis, then uses RSSI analysis as a consistency check to determine whether jamming is causing the poor network performance.

Another strategy for defending against jamming is to have nodes collaboratively identify the jammed region and then route traffic around it.[1] Such a mechanism would be redundant in the face of constant jamming in a multihop network, because you would expect the routing protocol to automatically route around jammed regions. In the case of intermittent jamming, routes that pass through jammed portions of the network would be unreliable. Routing protocols such as TinyOS Destination-Sequenced Distance-Vector Routing,[8] which associates a link quality estimator with each link to form paths using high-quality bidirectional links, would route around these portions of the network.

Other physical-layer attacks include node tampering or destruction. Although you can't prevent destruction of nodes deployed in an unsecured area, redundant nodes and camouflaging can mitigate this threat. Defenses against tampering include hiding or camouflaging nodes, tamper-proofing packages, or implementing tamper reaction such as erasing all program or cryptographic memory.[1]

The link/MAC layer

MAC protocols operate at the link layer, and most require cooperation between nodes to arbitrate channel use, making them particularly vulnerable to DoS attacks. Link-layer threats include collisions, interrogation, and packet replay. A collision attack is synonymous with the reactive-jamming attack we just described. You can mitigate some collisions by using error-correcting codes. However, ECCs add transmission overhead, consuming additional energy.

An interrogation attack exploits the two-way request-to-send/clear-to-send (RTS/CTS) handshake that many MAC protocols use to mitigate the hidden-node problem. An attacker can exhaust a node's resources by repeatedly sending RTS messages to elicit CTS responses from a targeted neighbor node. Antireplay protection and strong link-layer authentication can mitigate these attacks. However, a targeted node receiving the bogus RTS messages still consumes energy and network bandwidth.

Another link-layer threat to WSNs is the denial-of-sleep attack, which prevents the radio from going into sleep mode.[9] Frank Stejano and Ross Anderson first introduced the notion of this attack, calling it *sleep deprivation torture* and investigating its impact on battery-powered mobile devices.[10] An attacker might choose to execute a denial-of-sleep attack over a simple jamming-based DoS attack on a WSN to limit the attack's duration. To permanently disable a sensor network, a jamming attack might take months to deplete the targeted device's batteries. On the other hand, a clever denial-of-sleep attack that keeps the sensor nodes' radios on would drain the batteries in only a few days (at least for the class of devices considered here). Also, many denial-of-sleep attacks don't require a constant signal, making it more difficult to identify the traffic as malicious and to locate the attacking node via its emitted transmissions.

MAC protocols are a natural focus for denial-of-sleep attacks. This is because they control the functionality of the transceiver, which consumes more energy than any other component on most wireless-sensor platforms.[2] The link layer coordinates access to the physical medium linking a network's nodes. In a WSN, the link layer dictates when the radio should transmit frames, listen to the channel to receive data, and sleep to conserve energy. MAC protocols designed for WSNs use various techniques to save battery power by placing the radio in low-power modes when the radio isn't actively sending or receiving data. The Crossbow Mica2 consumes 36.81 mW in receive mode and 0.048 mW in sleep mode (see table 1). Two standard 3,000 mAh AA batteries will last over

A clever denial-of-sleep attack that keeps the sensor nodes' radios on would drain the batteries in only a few days.

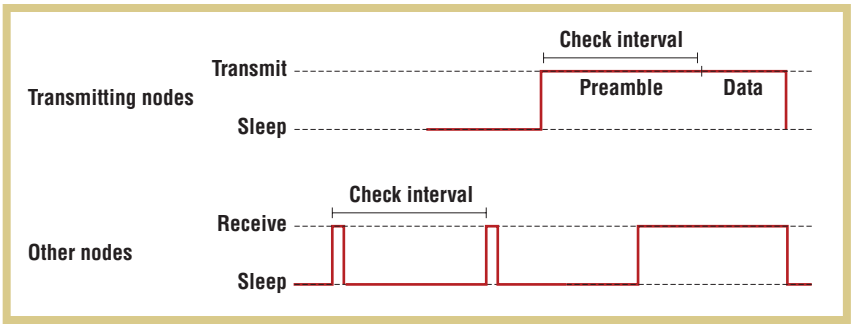


Figure 1. Berkeley medium-access-control transmitter and receiver behavior.

4,000 days for a device in sleep mode but only 10 days for a device in receive mode. This disparity between receive cost and sleep cost leads to an exponential increase in network lifetime as sleep time increases, suggesting that an attack that decreases sleep time by even a few percentage points can dramatically decrease a network's lifetime.

Because differences exist in packet structure and timing between WSN MAC protocols, an attacker can determine which MAC protocol a particular WSN is using by analyzing network traffic. This is enough information to mount an efficient denial-of-sleep attack against most sensor networks employing energy-efficient protocols, such as Sensor MAC (S-MAC),[11] Berkeley MAC (B-MAC),[12] or Timeout MAC (T-MAC).[13]

S-MAC divides time into 1,300-ms frames and simply has nodes sleep for a fixed percentage of each frame to conserve energy. It synchronizes the frames of single-hop neighborhoods of nodes so that the nodes are awake at the same time and thus can communicate. It does this using control packets called SYNC packets, which contain a field that indicates to surrounding nodes when the transmitting node will next enter sleep mode. When a node receives a SYNC packet, it resets its sleep timer to maintain synchronization with the transmitting node. If an attacker can construct counterfeit SYNC packets, he or she can periodically send one containing a sleep delay longer than the frame duration to keep clusters of nodes awake permanently.

Packet authentication can prevent this attack. However, recording and replaying legitimate SYNC packets at a rapid rate causes nodes to continually reset their sleep timers according to the received packet's value, also preventing them from sleeping. Link-layer authentication and antireplay support can protect against these attacks.

T-MAC uses a similar synchronization mechanism, but it improves on S-MAC's energy efficiency by using an *adaptive time-out* that lets nodes enter sleep mode when there's no more traffic in the network.[13] Each node counts down its time-to-sleep, which is reset to the adaptive time-out value each time it transmits or receives a packet. When the time-to-sleep counter expires, the nodes go to sleep until the next frame begins. An attacker can keep T-MAC nodes awake permanently by broadcasting or replaying a constant stream of small packets at an interval slightly shorter than the network's adaptive time-out duration.

B-MAC uses low-power listening for energy efficiency.[12] Using LPL, nodes in a B-MAC network don't synchronize schedules but instead periodically poll the wireless channel at a set check interval and spend the rest of the time in low-power sleep mode. Transmitting nodes send a preamble that's slightly longer than the check interval, followed by the data packet (see figure 1). This ensures that all nodes in the transmitter's one-hop neighborhood have polled the channel during the preamble. When a node overhears a preamble during a poll, it remains awake to receive the subsequent data packet. If an attacker sends a constant stream of unauthenticated or replayed broadcast packets, B-MAC nodes will overhear, on average, one-half of each preamble, plus the packet's data portion. This attack keeps B-MAC nodes awake over half the time, on average.

We modeled several attacks in a separate study of WSN MAC protocol vulnerabilities.[9] The results showed that if an attacker knows the protocol, he or she can mount a denial-of-sleep attack even without penetrating link-layer encryption. If the attacker can penetrate encryption, more effective attacks are possible, and the attacker can reduce a network's lifetime from several months to only a few days. We propose a framework for mitigating these denial-of-sleep threats that includes strong link-layer authentication, antireplay protection, jamming identification and mitigation, broadcast attack protection, and tamper resistance.[9] Although the basic primitives for WSN security detailed earlier provide mechanisms for authentication and some support for antireplay, no other current research thoroughly investigates prevention or mitigation of link-layer denial-of-sleep attacks.

The network layer

Routing-disruption attacks can lead to DoS attacks in multihop sensor networks. Chris Karlof and David Wagner thoroughly discuss sensor network routing vulnerabilities and attack countermeasures.[14] General attacks on routing protocols include spoofing, replaying, or altering routing traffic. Link-layer authentication and antireplay can effectively prevent these attacks.

A malicious node that subverts the network's routing protocol can mount a DoS attack by making itself part of many routes and then dropping all packets (in a *black hole* attack). Or, it can selectively forward packets to reduce the probability of detection.

One way to combat black holes

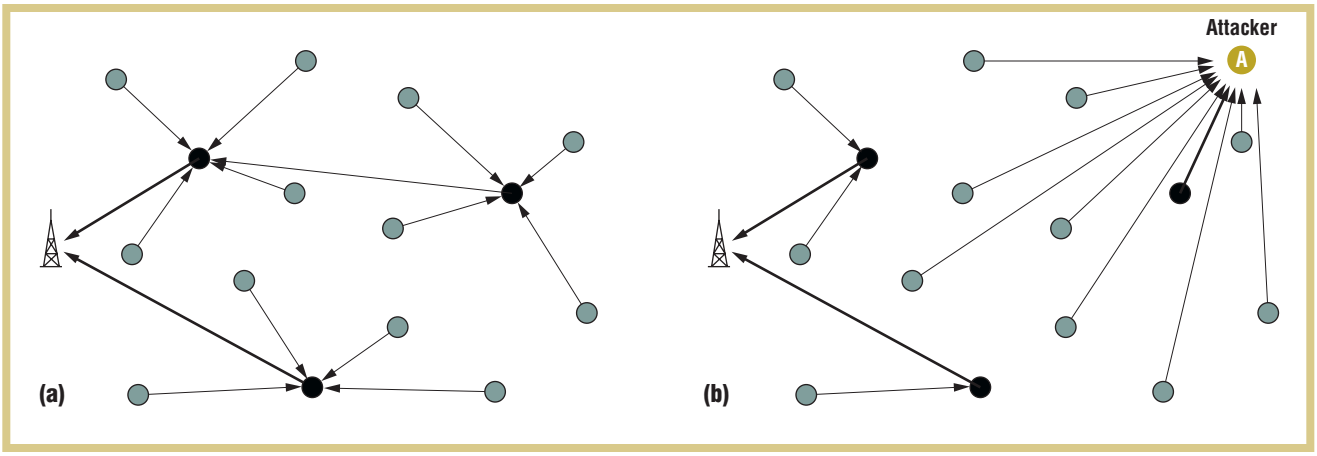


Figure 2. How an attack can take over large portions of a network: (a) a properly clustered network and (b) a network subverted by a bogus cluster-head volunteer message from an attacker.

and selective forwarding is implicit acknowledgments, which ensure that packets are forwarded as they were sent. Another technique is multipath routing, which sends the same data over multiple paths to give it a higher probability of reaching its destination. However, neither solution is attractive for sensor networks. Implicit acknowledgments require that the sensor node's radio be active (thus increasing power consumption), and they're unreliable when bidirectional links aren't guaranteed. Multipath routing wastes power on redundant paths and consumes additional network bandwidth. Also, it might not be feasible in sparse networks owing to the lack of routing options.

Hello flooding is an attack that doesn't require the attacker to compromise encryption.[14] Many routing protocols have nodes broadcast hello messages to inform one-hop neighbors of their presence. An attacker mounts a hello flood by recording hello packets, sending them from a laptop-class node with high transmit power. These replayed hello packets reach nodes that the originating node can't communicate with directly. Any node that uses the originating node as the next hop in a route but that isn't within that node's radio range won't be able to reliably forward traffic.

Pairwise authentication, which lets nodes verify bidirectional links before constructing routes, can combat this attack. Geographic routing protocols such as Geographic and Energy-Aware Routing[15] that let nodes discount hello messages from nodes not within communication range can also prevent this attack. Geographic protocols require each node to know its location and be able to communicate that location to other nodes.

Large-scale sensor deployments use clustering to route traffic in an energy-efficient way via data aggregation at cluster heads. By organizing into clusters, nodes can also reduce their transmit power levels since they need only reach the nodes in their cluster. This reduces energy consumption for transmitters and improves spatial reuse in the network. Most clustering protocols further manage energy consumption by reclustering often and rotating the cluster-head burden throughout the network's nodes.

The exact clustering process differs by protocol, but the basic steps are as follows: A certain proportion of nodes volunteer to be cluster heads on the basis of energy levels, desired cluster size, or some other metric. These nodes advertise their status as a cluster head. Other nodes join clusters by selecting a cluster head, usually selecting the one with the strongest signal, and then they broadcast a message, indicating membership in the cluster.

An attacker can subvert this process in several ways. By transmitting bogus cluster-head volunteer messages using a very strong radio signal, a network intruder might trick numerous nodes into joining a nonexistent cluster. Recording and later replaying these cluster volunteer messages can have the same effect. Figure 2 shows how this attack can take control of large portions of a network.

The first steps in mitigating such attacks are traffic authentication and antireplay support, which will cause nodes to ignore counterfeit cluster-head volunteer messages. Kun Sun and his colleagues propose a secure distributed-clustering protocol, based on *cliques* in which all nodes use public-key encryption to establish trust relationships with their neighbors.[16] This mechanism relies on asymmetric cryptography, which sensor networks usually avoid, because such protocols have high computational complexity and memory requirements.

Instead of having nodes volunteer to become cluster heads, some clustering protocols use cluster-head elections based on each nodes' stated resources, such as current energy supply. A network intruder might lie when providing resource information to ensure it's elected

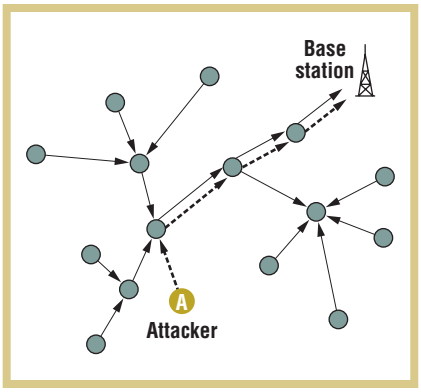


Figure 3. A path-based DoS attack in a WSN. An attacker injects network traffic, which consumes bandwidth on the path to the base station and causes the DoS.

as a cluster head. Garth Crosby, Niki Pissinou, and James Gadze introduce a trust-based framework for secure cluster-head election in ad hoc networks.[17] Their technique, although promising, relies on a combination of network-wide, cluster-wide, and pairwise encryption keys, which makes it impractical for large-scale sensor deployments.

Homing is a network layer attack that uses traffic pattern analysis to identify and target nodes that have special responsibilities, such as cluster heads or cryptographic-key managers. An attacker then achieves DoS by jamming or destroying these key network nodes. Header encryption is a common prevention technique, but it doesn't completely prevent traffic analysis. Simply analyzing the volume of traffic in various portions of the network might be enough to identify the location of cluster-head nodes or base stations. Jing Deng, Richard Han, and Shivakant Mishra suggest using "dummy packets" throughout the network to equalize traffic volume and thus prevent traffic analysis.[18] Unfortunately, this wastes significant sensor node energy, so use it only when preventing traffic analysis is of utmost importance.

The transport layer

At the transport layer, which manages end-to-end connections, flooding attacks exploit protocols that maintain connection information at either end.[1] For example, in a TCP SYN (*syn*chronize) flood attack, an adversary sends multiple connection requests without ever completing the connection, thus overwhelming the target's half-open connection buffer. Connectionless transport protocols are immune to this type of attack, but they might not provide the necessary transport-layer functionality to applications. The primary defense against this is SYN cookies, which encode information from the client's TCP SYN message and return it to the client to avoid maintaining state at the server (see http://cr.yp.to/syncookies.html). Yet these techniques' computational and message overhead makes them undesirable for WSNs.

In a desynchronization attack, an attacker interrupts an active connection between two nodes by transmitting forged packets with bogus sequence numbers or control flags that desynchronize endpoints so that they'll retransmit data.[1] Header or full-packet authentication can defeat such an attack.

The application layer

At the application layer, an attacker might attempt to overwhelm network nodes with sensor stimuli, causing the network to forward large volumes of traffic to a base station. This attack consumes network bandwidth and drains node energy. However, it's effective only when particular sensor readings (such as motion detection or heat signatures) trigger communications—not when sensor readings are sent at fixed intervals.

You can mitigate this attack by carefully tuning sensors so that only the specifically desired stimulus, such as vehicular movement, as opposed to any movement, triggers them. Rate-limiting and efficient data-aggregation algorithms can also reduce these attacks' effects.

Another application-layer attack involves injecting spurious or replayed packets into the network at leaf nodes in a *path-based DoS attack*.[19] As the packet is forwarded to its destination, nodes along the path to the base station waste bandwidth and energy transmitting the traffic. This attack can starve the network of legitimate traffic, because it consumes resources on the path to the base station, thus preventing other nodes from sending data to the base station (see figure 3). Combining packet authentication and antireplay protection prevents these attacks.

Protocols such as TinyOS's Deluge network-programming system let you remotely reprogram nodes in deployed networks.[20] Most of these systems, including Deluge, are designed for use in a trustworthy environment. If the reprogramming process isn't secure, an intruder can hijack this process and take control of large portions of a network.

One security technique uses authentication streams to secure the reprogramming process.[21] This divides a program binary into a series of messages, each of which contains a hash of the next message. This mechanism ensures that an intruder can't hijack an ongoing program transmission, even if he or she knows the hashing mechanism. This is because it would be almost impossible to construct a message that matches the hash contained in the previous message. A digitally signed advertisement, which contains the program name, version number, and hash of the first message, ensures that the process is securely initiated.

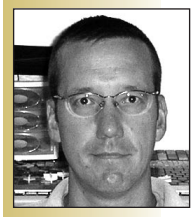
We can defeat many threats using existing encryption and authentication mechanisms, and other techniques (such as identifying jamming attacks)[7] can alert network administrators of ongoing attacks or trigger techniques to conserve energy on affected devices. However, we need additional research in low-overhead antireplay protocols. Such protocols

would complement current authentication techniques and would help prevent many of the attacks we've described. Defending against denial-of-sleep attacks is also crucial to the viability of sensor network deployments. Providing such security is critical if sensor networks are to realize the promise of widespread deployment.

REFERENCES

1. A.D. Wood and J.A. Stankovic, "Denial of Service in Sensor Networks," *Computer*, vol. 35, no. 10, 2002, pp. 54–62.
2. K. Sohrabi et al., "Protocols for Self-Organization of a Wireless Sensor Network," *IEEE Personal Comm.*, vol. 7, no. 5, 2000, pp. 16–27.
3. A. Perrig et al., "SPINS: Security Protocols for Sensor Networks," *Wireless Networks*, vol. 8, no. 55, 2002, pp. 521–534.
4. C. Karlof, N. Sastry, and D. Wagner, "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks," *Proc. 2nd Int'l Conf. Embedded Networked Sensor Systems*, ACM Press, 2004, pp. 162–175.
5. *IEEE Std. 802.15.4, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification for Low-Rate Wireless Personal Area Networks (LR-WPANs)*, IEEE, 2003.
6. N. Sastry and D. Wagner, "Security Considerations for IEEE 802.15.4 Networks," *Proc. ACM Workshop Wireless Security*, ACM Press, 2004, pp. 32–42.
7. W. Xu et al., "The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks," *Proc. 11th Ann. Int'l Conf. Mobile Computing and Networking*, ACM Press, 2005, pp. 46–57.
8. A. Woo, T. Tong, and D. Culler, "Taming the Underlying Challenges for Reliable Multihop Routing in Sensor Networks," *Proc. 1st ACM Int'l Conf. Embedded Networked Sensor Systems*, ACM Press, 2003, pp. 14–27.
9. D. Raymond et al., "Effects of Denial of Sleep Attacks on Wireless Sensor Network MAC Protocols," *Proc. 7th Ann. IEEE Systems, Man, and Cybernetics (SMC) Information Assurance Workshop* (IAW), IEEE Press, 2006, pp. 297–304.
10. F. Stajano and R. Anderson, "The Resurrecting Duckling: Security Issues for Ad-Hoc Wireless Networks," *Proc. 7th Int'l Workshop Security Protocols*, Springer, 1999, pp. 172–194.
11. W. Ye, J. Heidemann, and D. Estrin, "Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks," *IEEE/ACM Trans. Networking*, vol. 12, no. 3, 2004, pp. 493–506.
12. J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," *Proc. 2nd ACM Int'l Conf. Embedded Networked Sensor Systems*, ACM Press, 2004, pp. 95–107.
13. T. VanDam and K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," *Proc. 1st ACM Int'l Conf. Embedded Networked Sensor Systems*, ACM Press, 2003, pp. 171–180.
14. C. Karlof and D. Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures," *Proc. 1st IEEE Int'l Workshop Sensor Network Protocols and Applications*, IEEE Press, 2003, pp. 113–127.
15. Y. Yu, R. Govindan, and D. Estrin, *Geographical and Energy Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks*, tech. report UCLA/CSD-tr-01-0023, Computer Science Dept., Univ. of California, Los Angeles, 2001.
16. K. Sun et al., "Secure Distributed Cluster Formation in Wireless Sensor Networks," *Proc. 22nd Ann. Computer Security Applications Conf.*, IEEE CS Press, 2006, pp. 131–140.
17. G.V. Crosby, N. Pissinou, and J. Gadze, "A Framework for Trust-Based Cluster Head Election in Wireless Sensor Networks," *Proc. 2nd IEEE Workshop Dependability and Security in Sensor Networks and Systems*, IEEE Press, 2006, pp. 13–22.
18. J. Deng, R. Han, and S. Mishra, "Intrusion Tolerance and Anti-Traffic Analysis Strategies for Wireless Sensor Networks," *Proc. Int'l Conf. Dependable Systems and Networks*, IEEE CS Press, 2004, pp. 637–656.
19. J. Deng, R. Han, and S. Mishra, "Defending against Path-Based DoS Attacks in Wireless Sensor Networks," *Proc. 3rd ACM Workshop Security of Ad Hoc and Sensor Networks*, ACM Press, 2005, pp. 89–96.
20. J.W. Hui and D. Culler, "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale," *Proc. 2nd ACM Conf. Embedded Networked Sensor Systems*, ACM Press, 2004, pp. 81–94.
21. P.K. Dutta et al., "Securing the Deluge Network Programming System," *Proc. 5th Int'l Conf. Information Processing in Sensor Networks*, ACM Press, 2006, pp. 326–333.

the AUTHORS



David R. Raymond is a third-year PhD student in Virginia Tech's Bradley Department of Electrical and Computer Engineering. His research interests include energy-efficient medium-access-control protocols for wireless sensor networks, mobile and ad hoc networking, and network security. He received his MS in computer science from Duke University. He's a student member of the IEEE and the ACM. Contact him at the Bradley Dept. of Electrical and Computer Eng., Virginia Tech, 302 Whittemore Hall (0111), Blacksburg, VA 24061; raymondd@vt.edu.

Scott F. Midkiff is a professor in Virginia Tech's Bradley Department of Electrical and Computer Engineering. He has been on a temporary assignment as a program director at the US National Science Foundation since September 2006. His research interests include system issues in wireless and ad hoc networks, network services for pervasive computing, and performance modeling of mobile ad hoc networks. He received his PhD in electrical engineering from Duke University. He's a senior member of the IEEE. Contact him at the Bradley Dept. of Electrical and Computer Eng., Virginia Tech, 302 Whittemore Hall (0111), Blacksburg, VA 24061; midkiff@vt.edu.