S-MAC protocol vulnerabilities to DoS attacks

Quentin Monnet

***Abstract*—S-MAC is a MAC layer protocol designed for WSNs, based on the MAC layer of the IEEE 802.11$^{\text{TM}}$ stack. The specifications of S-MAC enable an attacker to launch a variety of denial of service attacks in a network running this protocol; this work is an attempt to list those attacks.**

***Index Terms*—S-MAC; WSN; Wireless protocol; Energy; Denial of service; Security**

I. Front matter

A. About this work

This document is a technical report about vulnerabilities of S-MAC protocol to denial of service attacks. This work was realized in 2012, but written (more accurately, translated to English, completed with "countermeasures", and pushed to the web) in 2015. It is released under the Creative Commons Zero 1.0 Universal License (Public Domain Dedication).

Last edit: July 29, 2015.

B. Acronyms

DoS:	Denial of Service (a generic kind of attack aiming at disrupting the normal behavior of the network)
MAC:	Media Access Control, second layer of the TCP/IP model
S-MAC:	Sensor-MAC, a MAC layer protocol designed for WSNs
WSN:	Wireless Sensor Network, self-organized network made of sensors embedding cheap hardware (thus with low resources) and communicating through wireless protocols

II. S-MAC protocol

If you found this paper, there is a great chance that either you know S-MAC already, or the remaining of this report will probably not be of much interest to you. In both cases, there is limited interest in describing S-MAC in details here. Plus this is not a full article but a technical note only, so I'll just skip the description of this protocol. If you do want to learn more about S-MAC, see its original specification [1], or this survey for a summary [2]. You're welcome.

OK, just as a short reminder nevertheless: S-MAC organizes the nodes of the network into virtual clusters (different clusters may overlap). Inside a cluster, nodes synchronize their activity and sleep periods; each activity period is split into two parts: the first one is used to resynchronize the sensors in order to prevent clock drift, the second one is dedicated to message passing.

It is somewhat assumed that the reader has some understanding of the functioning of the IEEE 802.11$^{\text{TM}}$ MAC protocol (having read the specifications of S-MAC to see the difference between the two protocols should be enough).

III. Potential S-MAC vulnerabilities to DoS

This is a list of potential issues that I noticed with S-MAC in regard to DoS attacks. It is (most probably) not exhaustive. All attacks are described in three steps: first the process of the attack, then the expected implications on the network functioning, and third potential countermeasures or mitigation methods.

Laboratoire d'Algorithmique, Complexité et Logique
Université Paris-Est Créteil, France
Contact: quentin.monnet@lacl.fr

A. Attacks raised on nodes synchronizing

Attacks in this subsection targets the synchronization step of the nodes of a same virtual cluster.

1) Declaring nonexistant clusters

- **Process** In the event of overlapping clusters, the nodes that belong to several clusters may come to adopt several schedules. Thus an attacker can simulate the existence of several distinct surrounding clusters, so that its neighbors get attached to all of them. In this scenario the attacker acts as a *synchronizer* and announce multiple sleep delays during the initial synchronizing step. If needed, it can resort to MAC spoofing.
- **Consequences** Nodes under attack react as if they were in an area with multiple overlapping clusters, and try to adopt all schedules so as to be able to communicate with all their (actually mostly fictive) neighbors. Hence they will stay awake on longer duty cycles and consume more energy than needed.
- **Countermeasures** Authentication mechanisms should prevent the attacker to impersonate distinct nodes and to declare distinct clusters. Checking that announced virtual clusters contain more than one node (but several nodes instead, with distinct signal power for instance) can also be a solution.

2) Extended awake period

- **Process** For each duty cycle, during the synchronizing step, an attacker can send to all its neighbors a sync packet announcing that it will sleep in $T_{sync_{max}}$, where $T_{sync_{max}}$ is the maximal delay value permitted by the protocol. After that, the attacker immediately goes to sleep.
- **Consequences** For each cycle, nodes under attack have to wait $T_{sync_{max}}$ before starting their sleep period, while the attacker saves its resources. Again, targeted nodes uselessly consume their energy.
- **Countermeasures** Systematic announcements of this $T_{sync_{max}}$ value should be considered suspicious, and reported. Nodes could also check that the suspicious node is actually still awake and responding after such announcements.

3) Virtual cluster table overflow

- **Process** An attacker can announce more virtual clusters than surrounding nodes are able to handle and to store in their internal table.
- **Consequences** may vary, depending on the implementation of the protocol. What happens in case of an overflow on this table? Do the first (and presumably legit) registered schedules get deleted? In this case, targeted nodes could be tricked into considering only corrupted schedules? In this situation communication could become impossible between some neighbor nodes not sharing any common schedule. Besides, if the operating system has been poorly written and that such an overflow can crash a process, things become even better for the attacker.
- **Countermeasures** Obviously crashing the node should not happen if the software is correctly designed. An overflow on the cluster table could be handled by temporarily refusing to accept *new* clusters (instead of overwriting the first clusters that were detected), but it could fail if the attacker declares all fake clusters first. Other algorithms would need a deeper study; at least, clusters "working well" (with a high number of neighbors, with no significant error rates on transmissions) should be conserved.

4) Forced desynchronization

- **Process** The attacker spoofs some legit MAC address and announces a new sync delay to some neighbor nodes, such that the new schedule is not compatible with the previous one.

- **Consequences** Synchronization between targeted nodes and the legit node whose MAC address has been spoofed is lost. They can no more communicate in a direct fashion.
- **Countermeasures** Authentication of the nodes in the network are the key here.

5) Jamming attack on synchronizing

- **Process** The attacker jams the sync packets of its neighbors.
- **Consequences** Targeted nodes cannot fix a common schedule, thus they fail to establish a communication.
- **Countermeasures** Traditional mechanisms used to mitigate jamming (*spread spectrum* for example) should be used in this case.

B. Attacks raised on message sharing

Attacks in this subsection target the message passing step between the nodes of a virtual cluster. Many of them are attacks that could be launched against the vanilla MAC protocol of IEEE 802.11^{TM} stack, since S-MAC is based upon it.

1) Denied acknowledgement

- **Process** On receiving a frame, the attacker does not send back any ACK packet. The ACK packet can even be denied only for the last frame of the packet (as a reminder: contrary to the vanilla MAC protocol in use with IEEE 802.11^{TM}, S-MAC specifies that if no acknowledgement is received, the DATA fragment is sent again until a ACK is sent back by the recipient, or until the upper limit of attempts (whose value is fixed during the implementation) is reached; we note this limit R_{max}).
- **Consequences** Instead of sending data once and for all, targeted node sends each fragment to the attacker up to R_{max} times. Thus it uselessly occupy the medium and wastes its energy.
- **Countermeasures** A neighbor node with a good signal power often failing to acknowledge for packets (or systematically reaching R_{max}) should be considered as suspicious, and reported (to the cluster head or to the base station).

2) Simple packet forgery

- **Process** The attacker forges RTS, CTS, DATA or ACK packets and sets the network allocation vector field (NAV) to the duration of the transmission, the upper limit set by the implementation (and noted D_{max}). But in the end, no data follows these announcements.
- **Consequences** All neighbors of the attacker, on reception of such packets, enter energy-saving mode and stop listening to the medium during D_{max}, since it is supposed to be busy (this is the *virtual carrier sense* mechanism). Thus they cannot communicate, and even potentially miss legit transmissions during this period.
- **Countermeasures** Nodes could periodically check whether announced transmissions are actually processed. If they detect that a given node regularly gives up the time slot it has asked for, it should be considered as suspicious.

3) Packet forgery with MAC spoofing (see Figure 1)

- **Process** The attacker X overhear a RTS packet sent from node A to B. The attacker X can send a CTS packet containing the MAC address of B node as the source address. Thus A will start sending data to B. In the event that B is already receiving data from a further node C whose transmission has not been detected by A (this is the *hidden terminal problem*), then data sent from A to B will collide with fragments sent from C to B, resulting in corrupted transmissions.
- **Consequences** Node B will fail to receive correctly data sent by both C and A, and those two nodes may try to send again their DATA fragments several times causing useless wastes of energy (and occupation of the medium).
- **Countermeasures** Authentication mechanisms.

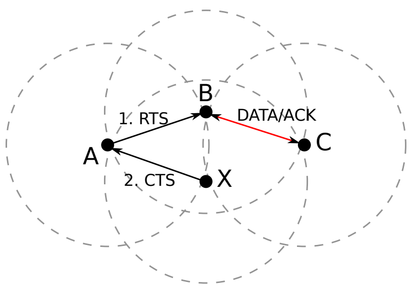

Figure 1. Scheme illustrating the packet forgery attack with MAC spoofing

4) Classical jamming attack: data corruption

- **Process** The attacker corrupts data sent between two nodes.
- **Consequences** vary depending on targeted packet. Corrupted RTS or CTS will prevent the whole transmission to happen. A corrupted DATA packet may result in several scenarios: it can alter data gathered by the base station if there is no error detection or integrity check performed on the messaged. But the most probable case is that the recipient node will detect the collision and will send no ACK packet. As for a corrupted ACK packet, the DATA fragment will need to be sent again (until R_{max} attempts have been made), resulting in energy depletion.
- **Countermeasures** Checksums or even error codes could be use to at least detect corrupted data, and potentially make the recipient able to correct it. Classical detection measures for jamming apply here.

5) Jamming, the greedy way: monopolizing the channel

- **Process** The attacker regularly emits some bits on the channel.
- **Consequences** Neighbor nodes have to pass through a *physical carrier sense* step before sending their RTS packet. If they detect some transmission they consider the medium as busy and cannot send their own frames. Thus this attack may have to objectives: preventing useful data to be sent over the channel, and monopolizing the medium for the sole use of the attacker so as to obtain a better transmission rate.
- **Countermeasures** If the emitter of these junk bits can be identified, a new channel could be established without the attacker. Identification is probably easier to perform with greedy attacks (since the attacker seeks a better transmission rate, thus emitting with a MAC address; if it is the only one able to emit, it becomes suspicious) than for the jamming part only (if the attacker does not seek to earn a better rate, it will probably not indicate its identity in the junk frames). Some methods have been proposed to detect such attacks [3].

6) Greedy behavior: neglecting the DIFS time intervals

- **Process** The attacker does not wait for a DIFS interval during RTS and CTS, CTS and DATA, or DATA and ACK packets. This attack also exists on MAC protocol used in IEEE 802.11^{TM}.
- **Consequences** The attacker can obtain a better transmission rate by cheating during the contention for the access to the medium.
- **Countermeasures** A node regularly neglecting the DIFS interval should be considered as suspicious.

7) Greedy behavior: neglecting *physical carrier sense* step

- **Process** The attacker does not listen to the medium prior to send a RTS packet. This attack could be led in IEEE 802.11^{TM}, but with a limited efficiency (high collision risk); S-MAC makes it more likely to succeed, because at the beginning of the message passing step, all synchronized nodes of the virtual cluster are supposed to be listening the channel for at least a minimum value. So there is necessarily a short interval of time (except in case of large clock drift) when the channel should be free (all nodes wishing to transmit are busy listening to the channel).

- **Consequences** Again, the attacker can obtain a better transmission rate by cheating during the contention for the access to the medium.
- **Countermeasures** A node regularly neglecting the physical carrier sense step should be considered as suspicious.

IV. Remark

The article describing the S-MAC protocol specifies that for a better clarity, the authors have considered that all nodes had sleep periods of equal length (and similarly, duty cycle of equal length). Other attacks might be discovered if this condition is not verified.

References

[1] W. Ye, J. Heidemann, and D. Estrin, “An energy-efficient MAC protocol for wireless sensor networks,” in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM’02)*, vol. 3, New-York City, NY, USA, Jun. 2002, pp. 1567–1576.

[2] B. Yahya and J. Ben-Othman, “Towards a classification of energy aware MAC protocols for wireless sensor networks,” *Wireless Communications and Mobile Computing*, vol. 9, no. 12, pp. 1572–1607, Dec. 2009.

[3] K. Pelechrinis and M. Iliofotou, “Denial of service attacks in wireless networks: The case of jammers,” *IEEE Communications Surveys and Tutorials*, vol. 13, no. 2, pp. 245–257, 2011.