

Poster Abstract: AMSecure—Secure Link-Layer Communication in TinyOS for IEEE 802.15.4-based Wireless Sensor Networks

Anthony D. Wood, John A. Stankovic
 Department of Computer Science
 University of Virginia
 {wood, stankovic}@cs.virginia.edu

Categories and Subject Descriptors

D.4.6 [Operating Systems]: Security and Protection—Cryptographic Controls

General Terms: Security, Design, Performance

Keywords: Wireless Sensor Network Security, AES

1 Introduction

Many wireless sensor networks (WSNs) for medical, military, and control applications require strong security protection of messages. Yet, the algorithms and protocols used must be efficient in space and time due to the constrained resources of sensor devices.

Existing link-layer security solutions for WSNs, such as SPINS [3] and TinySec [2]), rely on software-level encryption and authentication routines. Due to memory and computation limitations, these have used algorithms that are considered less secure than the Advanced Encryption Standard (AES). Also, they do not provide explicit support for concurrent use of multiple keys and security modes.

We briefly present AMSecure, a new link-layer security component we developed for MICAz and Telos motes, which both use the Chipcon CC2420 radio transceiver. AMSecure uses its built-in support for inline AES cryptographic operations specified in the IEEE 802.15.4 standard [1]. Message confidentiality, authentication, integrity, replay protection, and semantic security are provided.

AMSecure uses hardware-accelerated cryptography in a backward-compatible manner with insecure Active Messages (the messaging layer in TinyOS). It supports multiple keys per source, allowing operation with the many key management schemes in the literature. It also allows TinyOS components to specify, on a per-message basis, the security mode to use: none, authentication-only, encryption-only, or authentication and encryption. AMSecure is therefore efficient, composable, and more flexible than previous solutions.

We fully implemented AMSecure on MicaZ motes, and evaluated it by collecting real performance measurements.

2 AMSecure Description

AES security modes supported by AMSecure are those specified in IEEE 802.15.4 [1] and which have hardware

support: none, CBC-MAC authentication-only, CTR mode encryption-only, and CCM, which combines authentication with encryption. These modes are selectable on a per-message basis, allowing application-level decisions about security policy based on message semantics or context.

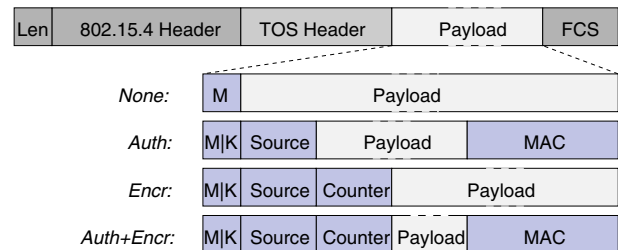


Figure 1. Message format for the AMSecure modes. $M|K$ is a bitfield consisting of the mode and key id.

Each AMSecure message contains only the header fields necessary for the desired mode of operation. Message formats are shown in Figure 1. At a minimum, an additional byte specifying a mode of *NONE* is necessary. For *authentication-only* mode, a key id and two-byte source are added, as well as the four-byte message authentication code (MAC). *Encryption* or *encryption-and-authentication* modes also include a two-byte non-repeating counter.

Keys are identified by $\langle \text{source, key-id} \rangle$ tuples, stored in the KeyStore module. The KeyStore also manages monotonically increasing transmit and receive counters, which is important for the security of CTR and CCM modes [4].

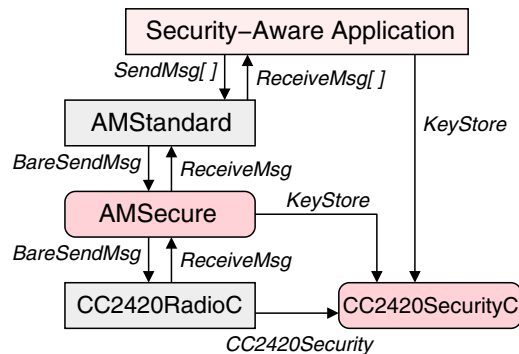


Figure 2. AMSecure component wiring in TinyOS.

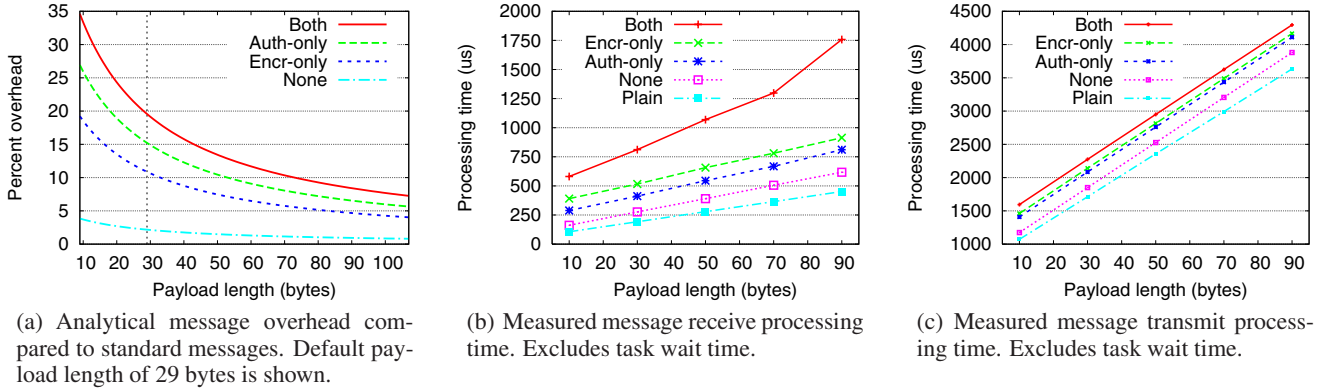


Figure 3. AMSecure overhead and performance for varying payload lengths up to 802.15.4 maximum.

AMSecure is a link-layer security suite, and the IEEE 802.15.4 and TinyOS headers are included in MAC computations. Therefore, neighbors share keys for authentication and/or encryption for messages shared among them. Messages from an attacker (that claim to be secure) are discarded when received by a legitimate node in the network, since the MAC will not be computed correctly.

Figure 2 shows how the TinyOS components are wired together for an application that uses AMSecure. Prior to sending or receiving a secure message, the application must store a key in the `CC2420SecurityC` module using the `KeyStore` interface. When sending a message, the application specifies the security mode and key-id in the message structure. The AMSecure component is interposed between AMStandard and the radio configuration, where it adds and removes the secure message headers/footers shown in Figure 1 as needed. When receiving or transmitting, `CC2420RadioC` consults `CC2420SecurityC` using the `CC2420Security` interface to setup parameters for inline security operations.

Backward compatibility is preserved by segmenting the TinyOS message type field. A compile-time constant sets the threshold separating standard Active Messages from AMSecure messages. AMSecure processing is only invoked when receiving message types above the specified threshold. This allows components unaware of security to be re-used in the network unchanged (after verifying their message types are below the threshold), albeit without security guarantees.

3 Evaluation

To evaluate the performance of AMSecure, we measured the message and processing overheads. Messages are always longer than without security, and it takes longer to perform cryptographic operations, hence security comes at a cost.

Figure 3(a) shows calculated percentage overhead due to message expansion. The overhead due to the AMSecure header and footer (MAC) naturally decreases as the payload length increases. A vertical line shows the default TinyOS payload length of 29 bytes. We believe the strong guarantees and flexibility of AMSecure are worth the overhead cost.

When receiving a message, AMSecure reads enough of the RXFIFO to determine whether this message should be processed as a secure frame, and in what mode. The in-

line security operation is configured on the CC2420, and the remainder of the frame is read out of the RXFIFO when it is done. As the message propagates through the AMSecure component, the header and footer are stripped out before the payload is signaled to the higher layers.

The processing time for all these steps is shown in Figure 3(b). The time required for receiving an insecure (“plain”) message is also shown. Multiple asynchronous task postings make simple measurement of the processing time difficult. The data presented here are the sums of the synchronous code blocks, so that task execution dynamics do not obscure the measurements. Times were measured using the Tektronix TDS 2022 scope’s rise-fall measure function, and do not include one-time costs like initializing the key in the radio registers. The largest incremental cost comes when using both encryption and authentication together.

Figure 3(c) shows similar measurements for the transmit path. These times include shifting the payload to make room for the AMSecure header, setting up transmit security, writing the frame to the TXFIFO, and cleaning up. In contrast to the receive measurements, these include the radio transmission time, and so are uniformly higher than Figure 3(b).

We see that the overhead of AMSecure is a predictable function of message length, and that processing times are low due to the aid of hardware-implemented cryptography.

4 Conclusion

We briefly presented AMSecure, a link-layer security component for TinyOS in IEEE 802.15.4-based WSNs. It is backward-compatible with insecure Active Messages, composable with many key distribution schemes by explicitly supporting multiple keys, flexible by allowing applications to choose security modes, and efficient by providing hardware-accelerated AES cryptography on constrained devices.

5 References

- [1] IEEE 802.15.4-2003. *Wireless MAC and PHY Spec. for Low Rate Wireless Personal Area Networks (LR-WPANs)*, 2003.
- [2] C. Karlof, N. Sastry, and D. Wagner. TinySec: a link layer security architecture for wireless sensor networks. In *Proc. of SenSys*, pages 162–175, 2004.
- [3] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. In *Proc. of MOBICOM*, pages 189–199, 2001.
- [4] N. Sastry and D. Wagner. Security considerations for IEEE 802.15.4 networks. In *Proc. of WiSe*, pages 32–42, 2004.