Utilizing Hardware AES Encryption for WSNs

by Felix Büsching, Andreas Figur, Dominik Schürmann, and Lars Wolf
Technische Universität Braunschweig | Institute of Operating Systems and Computer Networks
Felix Büsching | buesching@ibr.cs.tu-bs.de | Phone +49 (0) 531 391-3289

Motivation

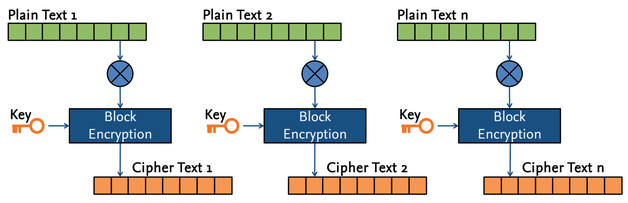
Encryption is essential in many Wireless Sensor Network (WSN) applications. Several encryption frameworks exist which are mostly based on software algorithms. However, nearly every up-to-date radio transceiver chip is equipped with an integrated hardware encryption engine. Here we show the benefits of utilizing an integrated hardware encryption engine in comparison to pure software-based solutions.

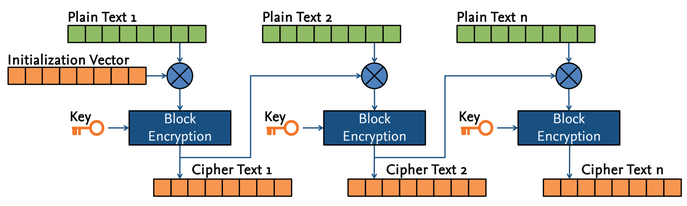
Advanced Encryption Standard - AES

- ... is a variant of Rijndael with a fixed block size of 128 bits.
- ... is based on a substitution-permutation network, which shall operate fast in both software and hardware.
- ... is a block cipher with various modes of operation.

Operation Modes of Block Ciphers

Different operation exist for different purposes. Electronic Codebook (ECB) and Cipher-block chaining (CBC) are the most common modes. In ECB each block is encrypted separately.

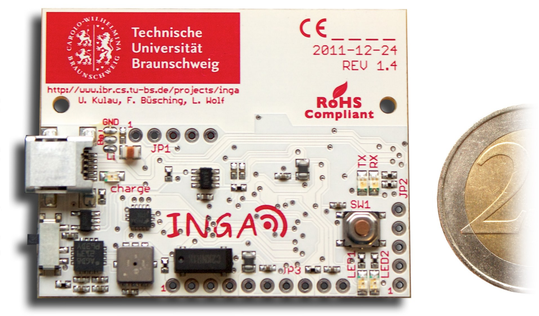




In CBC each plain-text block is XOR-ed with the previous cipher-text and afterwards being encrypted.

Implementations

Four different AES algorithms were implemented for **INGA** Wireless Sensor Nodes. While the first algorithm (1) **SW-AES-1** was realized for Contiki in C in a straight forward way, the (2) **SW-AES-2** implementation was improved by a static lookup table.
To compare our C implementations with a software reference, we utilized (3) **RijndaelFast**, an optimized **assembler implementation** for the Atmel ATmega family. We see this external implementation as a theoretical limit, knowing that this performance could never be reached when using an operating system like TinyOS or Contiki.
Most of the currently available radio transmitters and have an integrated hardware AES unit. These units can usually be addressed by special registers via SPI bus. When using an operating system like Contiki or TinyOS, the corresponding hardware drivers have to be implemented – we did this for Contiki running on INGA and by that created the (4) **HW-AES** implementation.

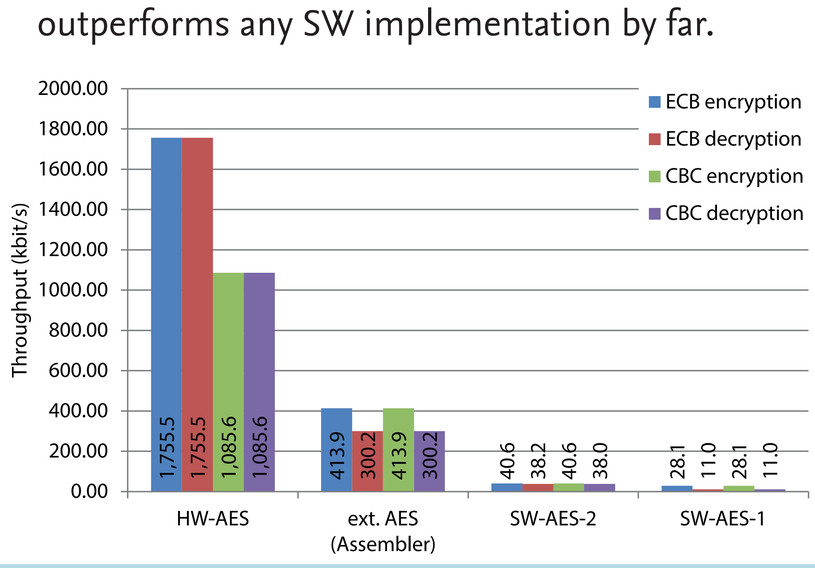


Resources

Our AES implementations, the AES hardware drivers, and the INGA Wireless Sensor Node are open source:
http://www.ibr.cs.tu-bs.de/projects/inga

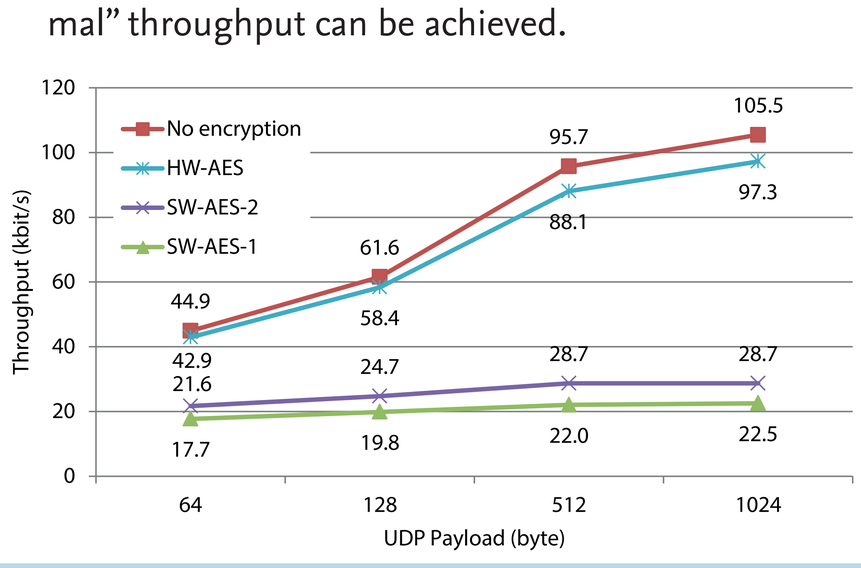
AES Throughput

The optimized software solution is nearly twice as fast as SW-AES-1. The assembler implementation outperforms both Contiki implementations, but, it is not working with any other software. The hardware utilization, which again runs in Contiki, outperforms any SW implementation by far.



UDP Throughput

We measured the UDP throughput between two nodes: without encryption, with hardware support and with our two software AES implementations. While software AES significantly cuts down the throughput, with hardware AES nearly the “normal” throughput can be achieved.



Code Size

The memory utilization [bytes] of our implementations can be divided in RAM and ROM (for data and functions).

	RAM	ROM	
		Data	Funct.
SW-AES-1	32	522	2514
SW-AES-2	32	2058	2462
HW-AES	0	0	518

The utilization of **hardware AES** only consumes **518 bytes of ROM** for the implementation of the drivers.

Conclusion: Use hardware AES, wherever possible!

Technische Universität Braunschweig

Institute of Operating Systems and Computer Networks