Utilizing Hardware AES Encryption for WSNs

Felix Büsching, Andreas Figur, Dominik Schürmann, and Lars Wolf

Technische Universität Braunschweig,
Institute of Operating Systems and Computer Networks (IBR)
Email: {buesching | figur | schuerm | wolf}@ibr.cs.tu-bs.de

Abstract. Encryption is essential in many 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. With this Poster we show the benefits of utilizing an integrated hardware encryption engine in comparison to pure software-based solutions.

1 Introduction

In most Wireless Sensor Network (WSN) applications data is recorded by wireless sensor nodes and then transmitted via a radio link (e.g. to a sink). While in wildlife monitoring or similar scenarios the data may be transferred in plain text and unencrypted, in other scenarios there is the demand for a reliable and secure transmission of confidential data. Especially in Body Area Networks (BAN), where vital parameters are recorded and transmitted, which are by definition personal, it is obvious that this data should not be easily accessible by a third party. Encryption – in general, and if implemented correctly – can ensure a certain degree of security and therefore may help to conserve privacy and keep confidental data secret.

One of the first implementations of link layer security in wireless sensor networks was TinySec [1] for TinyOS. It utilized the Skipjack algorithm which is vulnerable to several cryptanalysis attacks [2]. In a footnote, which was added later to their paper, the authors had to admit that AES would also be a viable choice with similar performance like Skipjack. Even TinySecs successor MiniSec [3] and another implementation called TinyKey [4] are based on Skipjack instead of using widely accepted and standardized block ciphers like AES.

ContikiSec [5] provides security for the operating system Contiki, utilizing a software AES implementation.

2 AES Implementations for WSNs

One of the major challenges for encryption in WSNs is the limited computational power of common wireless sensor nodes. Common sensor nodes are based on 8 or 16 bit microcontrollers, which are running at 4 to 8 MHz. Also the available memory (RAM and ROM) and (in most scenarios) the energy is limited. One

major design goal of AES was the opportunity to run it even on weak processors, therefore, it is surely possible to implement AES on such microcontrollers.

To get a deeper understanding of AES performance and cost issues, we implemented the mentioned algorithms and hardware drivers for INGA [6], a low-cost node consisting of an Atmel ATmega microcontroller and an Atmel AT86RF231 radio transceiver.

Simple Software Implementation. We implemented the original AES algorithm in a straight forward way. The block size and the key length were fixed to 128bit. Only ECB and CBC Mode were implemented.

Advanced Software Implementation. The authors of [7] have shown that there is still potential for improvements, thus, we improved our simple implementation by a lookup table, which needs another 1563 bytes of program memory.

Reference Assembler Implementation. To compare our C implementations for Contiki with a software reference, we utilized RijndaelFast[1], an optimized assembler implementation for the Atmel ATmega family (without integration in Contiki). 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.

Hardware Implementation. Most of the current available radio transmitters 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.

3 Evaluation

The software algorithms as well as the hardware-based approach were evaluated on INGA, running Contiki. We measured encryption throughput, application layer throughput and code size of the considered solutions.

Encryption Throughput. In Figure 1 the average achievable pure encryption throughput is shown. The optimized software solution (SW-AES-2) uses an additional lookup table in comparison to SW-AES-1 and, hence, is nearly twice as fast in all categories. The optimized assembler implementation outperforms both of the Contiki implementations, but, it is not working with any other software. A huge difference between the hard- and software implementation can be seen. The hardware utilization, which again runs under Contiki, outperforms any software implementation by far.

[1] `http://point-at-infinity.org/avraes/`

Application Layer Throughput. To see the impact of the usage of a hardware encryption engine, we measured the UDP throughput between two nodes: without encryption, with hardware support and with our two software AES implementations. As can be seen in Figure 2 while software AES significantly cuts down the throughput, with hardware AES nearly the "normal" throughput can be achieved.

Code Size. Table 1 gives the code size of all implementations. Functions and data add to the overall needed Flash memory. It is easy to see that the utilization of the hardware AES module on the radio also saves an substantial amount of RAM and ROM.

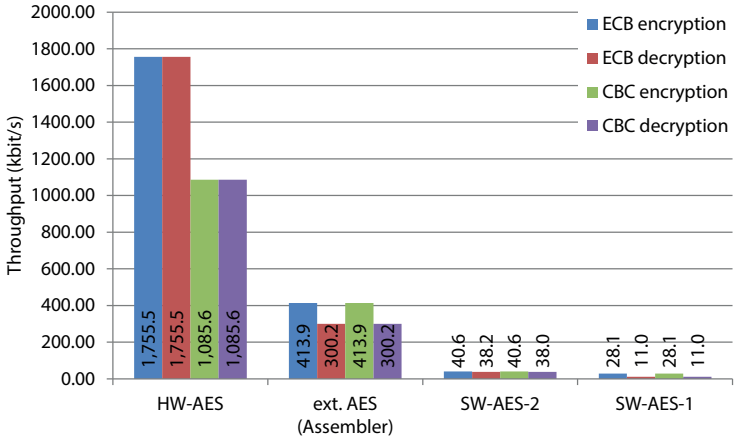

Fig. 1: Encryption throughput of INGA

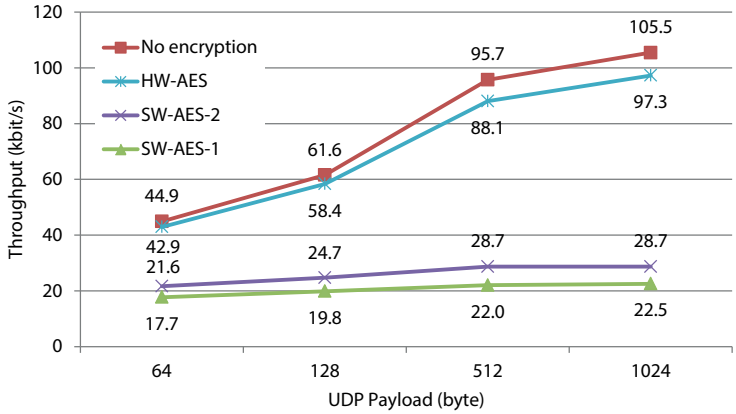

Fig. 2: UPD throughput

Table 1: Code size (bytes) of all three implementations

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

4 Summary and Conclusion

We have shown that utilizing the hardware for AES operations outperforms any software implementation. Although this result was expected, our evaluation demonstrates the significant gains which can be achieved, thus, if a hardware AES unit is present it should be utilized in any case. The implementation for Contiki and INGA is available for download at: `http://www.ibr.cs.tu-bs.de/projects/inga/`.

References

1. C. Karlof, N. Sastry, and D. Wagner, "Tinysec: a link layer security architecture for wireless sensor networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, ser. SenSys '04. New York, NY, USA: ACM, 2004, pp. 162–175.
2. E. Biham, A. Biryukov, O. Dunkelman, E. Richardson, and A. Shamir, "Initial observations on skipjack: Cryptanalysis of skipjack-3xor," in *Selected Areas in Cryptography*, ser. Lecture Notes in Computer Science, S. Tavares and H. Meijer, Eds. Springer Berlin Heidelberg, 1999.
3. M. Luk, G. Mezzour, A. Perrig, and V. Gligor, "Minisec: a secure sensor network communication architecture," in *Proceedings of the 6th international conference on Information processing in sensor networks*, ser. IPSN '07. New York, NY, USA: ACM, 2007, pp. 479–488.
4. R. Doriguzzi Corin, G. Russello, and E. Salvadori, "Tinykey: A light-weight architecture for wireless sensor networks securing real-world applications," in *Wireless On-Demand Network Systems and Services (WONS), 2011 Eighth International Conference on*, jan. 2011.
5. L. Casado and P. Tsigas, "Contikisec: A secure network layer for wireless sensor networks under the contiki operating system," in *Proceedings of the 14th Nordic Conference on Secure IT Systems: Identity and Privacy in the Internet Age*, ser. NordSec '09, Berlin, Heidelberg, 2009.
6. F. Büsching, U. Kulau, and L. Wolf, "Architecture and Evaluation of INGA - An Inexpensive Node for General Applications," in *Sensors, 2012 IEEE*. Taipei, Taiwan: IEEE, oct. 2012, pp. 842–845.
7. S. Didla, A. Ault, and S. Bagchi, "Optimizing AES for embedded devices and wireless sensor networks," in *Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities*, ser. TridentCom '08, ICST, Brussels, Belgium, Belgium, 2008.