

# Review of Hardware Architectures for Advanced Encryption Standard Implementations Considering Wireless Sensor Networks

Panu Hämäläinen<sup>1</sup>, Marko Hännikäinen<sup>2</sup>, and Timo D. Hämäläinen<sup>2</sup>

<sup>1</sup> Nokia Technology Platforms, WiWLAN SF  
Visiokatu 3, FI-33720 Tampere, Finland  
panu.hamalainen@nokia.com

<sup>2</sup> Tampere University of Technology, Institute of Digital and Computer Systems  
P.O.Box 553, FI-33101 Tampere, Finland  
marko.hannikainen@tut.fi, timo.d.hamalainen@tut.fi  
<http://www.tkt.cs.tut.fi/research/daci>

**Abstract.** Wireless Sensor Networks (WSN) are seen as attractive solutions for various monitoring and controlling applications, a large part of which require cryptographic protection. Due to the strict cost and power consumption requirements, their cryptographic implementations should be compact and energy-efficient. In this paper, we survey hardware architectures proposed for Advanced Encryption Standard (AES) implementations in low-cost and low-power devices. The survey considers both dedicated hardware and specialized processor designs. According to our review, currently 8-bit dedicated hardware designs seem to be the most feasible solutions for embedded, low-power WSN nodes. Alternatively, compact special functional units can be used for extending the instruction sets of WSN node processors for efficient AES execution.

## 1 Introduction

Cryptographic algorithms are utilized for security services in various environments in which low cost and low power consumption are key requirements. Wireless Sensor Networks (WSN) [1] constructed of embedded, low-cost, and low-power wireless nodes fall into the class of such technologies [2], ZigBee [3] and TUTWSN [4] as examples. Nodes themselves are independent of each other but they collaborate to serve the application tasks of WSNs by sensing, processing, and exchanging data as well as acting according to the data content [1]. WSNs are envisioned as cost-effective and intelligent solutions for various applications in automation, health care, environmental monitoring, safety, and security. A large part of the applications require protection for the data transfer as well as for the WSN nodes themselves [5]. Even though WSNs can contain devices with varying capabilities, in this paper the term *node* refers to an embedded, highly resource-constrained, low-cost, and low-power WSN device.

Compared to software, significantly higher performance and lower power consumption can be achieved with dedicated hardware and specialized processor architectures

tuned for the execution of security procedures in WSN nodes. A software implementation on a general-purpose processor always contains overhead due to instruction fetch and decode, memory access, and possibly due to an unsuitable instruction set and word size. As Advanced Encryption Standard (AES) [6] is a standardized encryption algorithm and considered secure, it has become the default choice in numerous applications, including the standard WSN technologies IEEE 802.15.4 [7] and ZigBee [3].

In this paper, we review and compare hardware architectures that are potentially suitable for AES implementations in WSN nodes. We have selected the architectures from more than 150 examined research papers, including both dedicated hardware as well as specialized cryptographic processor designs. We believe that the paper is comprehensive as well as valuable for designers evaluating and developing AES implementations for embedded, low-cost, and low-power WSN nodes. The survey focuses on academic research papers as publicly available information on commercial implementations is typically very limited. However, we believe that the reviewed designs comprehensively cover the utilized design choices and trade-offs suited for WSN node implementations.

The paper is organized as follows. Section 2 presents an overview of the AES algorithm, discusses high-level architectural alternatives for its hardware implementation, and argues their suitability for WSN nodes. In Section 3, we survey existing low-cost and potentially low-power AES hardware designs. Section 4 reviews specialized processor architectures proposed for efficient AES implementations in low-cost wireless devices. In this paper, a specialized processor architecture refers to a design that includes support for AES but the design can be capable of executing other tasks as well. A dedicated hardware implementation can only be used for executing AES.

## 2 Overview of AES Algorithm

AES [6] is a symmetric cipher that processes data in 128-bit blocks. It supports key sizes of 128, 192, and 256 bits and consists of 10, 12, or 14 iteration rounds, respectively. Each round mixes the data with a *roundkey*, which is generated from the encryption key.

The encryption round operations are presented in Fig. 1. The cipher maintains an internal, 4-by-4 matrix of bytes, called *State*, on which the operations are performed. Initially *State* is filled with the input data block and XOR-ed with the encryption key. Regular rounds consist of operations called *SubBytes*, *ShiftRows*, *MixColumns*, and *AddRoundKey*. The last round bypasses *MixColumns*. Decryption requires inverting these operations.

*SubBytes* is an invertible, nonlinear transformation. It uses 16 identical 256-byte substitution tables (*S-box*) for independently mapping each byte of *State* into another byte. *S-box* entries are generated by computing multiplicative inverses in *Galois Field*  $GF(2^8)$  and applying an affine transformation. *SubBytes* can be implemented either by computing the substitution [8,9,10,11,12] or using table lookups [10,13,14]. *ShiftRows* is a cyclic left shift of the second, third, and fourth row of *State* by one, two, and three bytes, respectively. *MixColumns* performs a modular polynomial multiplication in  $GF(2^8)$  on each column. Instead of computing separately, *SubBytes* and *MixColumns* can also be combined into large Look-Up-Tables (LUT), called *T-boxes* [9,15]. During each round,

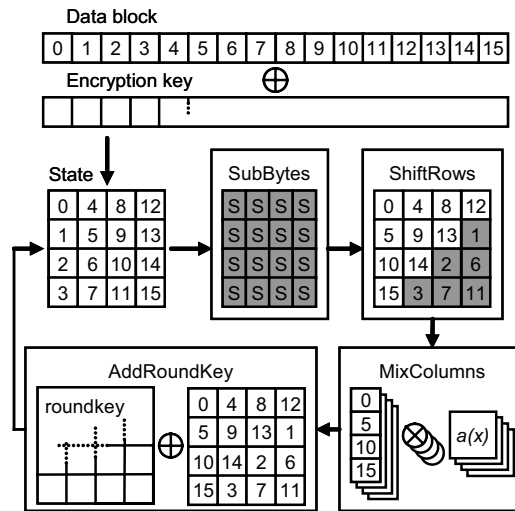


Fig. 1. Round operations of AES encryption

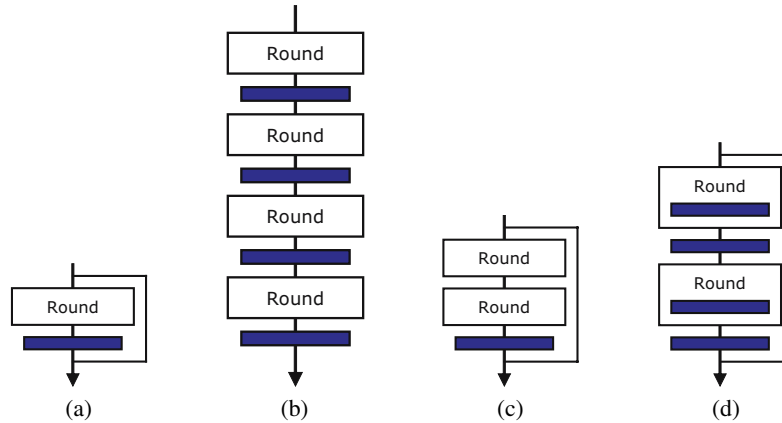
AddRoundKey performs XOR with State and the roundkey. Roundkey generation (*key expansion*) includes S-box substitutions, word rotations, and XOR operations performed on the encryption key. For more details on the AES algorithm and its inversion, we refer to [6].

### 2.1 Design Choices for AES Support in Hardware

The basic techniques for implementing a block cipher with rounds, such as AES, are iterated, pipelined, and loop-unrolled architectures [16]. The more advanced structures include partial pipelining and sub-pipelining combined with these basic techniques. The architectures are illustrated in Fig. 2.

The iterated architecture leads to the smallest implementations as it consists of one round component which is fed with its own output until the required number of rounds has been performed. The pipelined architecture contains all the rounds as separate components with registers in between. As a result, it is the fastest (in terms of throughput) and the largest of the basic structures. The loop-unrolled architectures perform two or more rounds per clock cycle and the execution of the cipher is iterated. In a pipelined architecture, unrolling can only decrease the latency of outputting the first block. In sub-pipelining, registers are placed inside the round component in order to increase the maximum clock frequency. In the partial pipelining scheme, the pipeline contains e.g. the half of the rounds with registers in between.

Although pipelined and loop-unrolled architectures enable very high-speed AES implementations, they also imply large area and high power consumption, which makes them unattractive for WSN nodes. Furthermore, they cannot be fully exploited in feedback modes of operation [9,14]. Feedback modes are often used for security reasons in encryption and for Message Authentication Code (MAC) generation, e.g. as in the



**Fig. 2.** Hardware architectures for round-based block cipher implementations: (a) iterated, (b) pipelined, (c) loop-unrolled, and (d) the combination of partial pipelining and sub-pipelining. The exemplar full cipher consists of four rounds.

security schemes of the standard WSN technologies [7,3]. Iterative architectures enable low-resource implementations with full-speed utilization also in feedback modes. The width of the AES data path can be further reduced to decrease logic area and power [8,9,10,11,12,13,14]. Hence, the review of this paper focuses on AES designs utilizing iterated structures.

In addition to the architectural choices, the design of AES enables a large number of algorithm-specific hardware trade-offs. The trade-offs consist of choosing between memory-based LUTs and combinatorial logic, decreasing the amount of parallelism, transferring the GF computations into another arithmetic domain, choosing between precomputed and on-the-fly key expansion, and sharing resources between encryption, decryption, and key expansion data paths. These aspects and their effects are discussed in the review of the following sections.

### 3 Hardware Implementations of AES

Since the ratification of AES in 2001, a large number of its hardware implementations has appeared. We surveyed more than 100 papers for the review of this section. According to the survey, most AES designs have been targeted at and implemented in Field Programmable Gate Array (FPGA) technologies. Whereas earlier AES designs mainly focused on intensively pipelined, high-speed implementations, the more recent work has concentrated on compactness and lower power consumption. Of all the designs, Table 1 lists the proposals which we have considered to have achieved the most significant results and which are possibly suitable for highly resource-constrained WSN nodes. The table is organized according to the time of publication of the designs, in order to reflect also the evolution in the field. A more comprehensive table, containing the

**Table 1.** Compact hardware implementations of AES

Design	Tech.	Data width	E/D (1)	Mode (2)	Keys (3)	S-box (4)	Cells (5)	Mem (6)	Lat (7)	Clk [MHz]	Tp [Mbit/s]
Ref. [16]	Xilinx	128	E	ECB	n/a	logic	3528	0	11	25	294
	Xilinx	128	E	ECB	n/a	logic	3061	0	21	40	492
Ref. [18]	Altera	16	E	ECB	n/a	ROM	1693	3	80	n/a	32
	Altera	16	ED	ECB	n/a	ROM	3324	3	80	n/a	24
Ref. [19]	Altera	32	E	ECB	n/a	ROM	824	10	44	n/a	115
	Altera	32	ED	ECB	n/a	ROM	1213	10	44	n/a	115
Ref. [8]	.11 $\mu$ m	32	ED	ECB	128	logic	5400	0	54	131	311
	.11 $\mu$ m	32	ED	ECB	128	logic	6300	0	44	138	400
	.11 $\mu$ m	64	ED	ECB	128	logic	8000	0	32	137	549
	.11 $\mu$ m	128	ED	ECB	128	logic	12500	0	11	145	1691
Ref. [13]	Xilinx	32	ED	ECB	128	ROM	222	3	46	60	166
Ref. [15]	Xilinx	32	ED	ECB	128	ROM	146	3	46	123	358
Ref. [10]	.60 $\mu$ m	32	ED	CBC	all	logic	8500	0	92	50	70
	Xilinx	mix	ED	CBC	all	ROM	1125	0	n/a	161	215
Ref. [11]	.35 $\mu$ m	8	E	ECB	128	logic	3600	0	1016	100	13
Ref. [14]	Altera	32	E	ECB	128	ROM	512	7	55	116	270
	Altera	32	ED	CCM	128	ROM	1434	11	112	78	90
Ref. [9]	Xilinx	8	ED	ECB	128	logic	124	2	n/a	67	2
Ref. [12]	.35 $\mu$ m	8	ED	ECB	128	logic	3400	0	1032	80	10
Ref. [20]	.13 $\mu$ m	8	E	ECB	128	logic	3100	0	160	152	121

- (1) Encryption (E) or decryption (D) or both (ED) supported for the mode in (2).  
(2) Supported mode of operation by the design.  
(3) 'n/a' means no key expansion included, a value refers to the supported keys sizes.  
(4) Specifies the technique used for the SybBytes implementation. 'ROM' means memory-based table-lookups and 'logic' combinatorial logic.  
(5) Resource consumption of the design. ASICs in gate-equivalents and FPGAs as general-purpose programmable resources: Xilinx *slices* or Altera *Logic Elements* (LE).  
(6) Dedicated memory components used from the specific FPGA of the reference.  
(7) The number of clock cycles for encrypting a block of data. Latencies caused by precomputed key expansion not included.  
'Tp' refers to the encryption throughput in the mode of (2). Latencies caused by precomputed key expansion not included.

highest-speed pipelined implementations as well, can be found in [17]. For the details of Xilinx and Altera FPGA devices, the readers are referred to their specific data sheets.

The references [16,18,19] are included in the table mainly for historical reasons as AES implementations that are better suited for WSNs have appeared later. However, those references were the first most comprehensive implementation studies that proposed compact AES designs as well. Ref. [16] presents a thorough study of AES encryption data path implementations with the different architectural choices described in Section 2.1 but lacks decryption and key expansion logic. The functionalities are also lacking from [18,19]. Nevertheless, [18,19] have been the first to propose *folded* AES

designs [13], in which the data path width has been decreased from its native width (128 bits). Later on, folding has successfully been utilized in the most compact and low-power AES implementations discussed below. Direct comparison between [16] and [18,19] is not possible as different FPGA and SubBytes implementation technologies have been used (logic vs. ROM). A Xilinx slice roughly equals to two Altera LEs. As [16] uses the native data width, its latency is lower and throughput higher than in [18,19]. Since [19] utilizes the T-box method, its LE count is lower than that of [18], despite of the wider data path. On the other hand, the method requires larger amount of memory for the LUTs. The folding factor increases the latency as more clock cycles are needed for processing a 128-bit block of data.

A number of iterative Application Specific Integrated Circuit (ASIC) designs with varying data path widths have been reported in [8]. The designs are based on an efficient S-box architecture and include en/decryption for 128-bit keys. Roundkeys are generated on-the-fly, either by sharing S-boxes with the main data path or by dedicating separate S-boxes for key expansion. The smallest version is a 32-bit AES architecture with four shared S-boxes. The results of [8] are still currently relevant: even though the gate counts are not the lowest, according to our knowledge the implementations offer the best area-throughput ratios of existing compact AES implementations.

A 32-bit AES architecture with a precomputed key expansion is developed for FPGAs in [13]. The design takes advantage of the dedicated memory blocks of FPGAs by implementing S-box as a LUT. The paper proposes a method for arranging the bytes of State so that it can efficiently be stored into memory components or shift registers. The arrangement allows performing ShiftRows with addressing logic. The same method is proposed again in [10]. For decreasing the amount of storage space as well as supporting various data path widths, we have developed the idea further in [21] without an implementation. In [14], we removed the decryption functionality of [13] and used the core for implementing the security processing of IEEE 802.15.4 [7] and ZigBee [3] in a low-cost FPGA. Ref. [15] improves the FPGA resource consumption of [13] with the T-box method. The design requires equal amount of memory components in the FPGA but uses them more efficiently.

A resource-efficient ASIC design supporting en/decryption is presented in [10]. The on-the-fly roundkey generation shares S-boxes with the main data path. The design is based on a regular architecture that can be scaled for different speed and area requirements. The smallest ASIC version contains a 32-bit data path. The FPGA design uses varying data widths for different phases of the algorithm. Support for the Cipher Block Chaining (CBC) encryption mode is also included. Compared to the 32-bit implementations of [8], the throughput of the ASIC implementation is lower and area larger. Ref. [10] also uses an older ASIC technology which prevents absolute area comparisons. However, the latencies of [8] are lower, which indicates that its designs are more efficient.

A low-power and compact ASIC core for 128-bit-key AES encryption is reported in [11]. The 8-bit data path is used for the round operations as well as for the on-the-fly key expansion. The data path contains one S-box implemented as combinatorial logic. State and the current roundkey are stored in a  $32 \times 8$ -bit RAM, which has been implemented with registers and multiplexers. The memory is intensively used by cycling each intermediate result through the RAM, increasing the total cycle count of the design. For

MixColumns, the design uses a shift-register-based approach, which is capable of computing the operation in 28 cycles. Decryption functionality is added to the design in [12], which also reports results from a manufactured chip. As stated, an increase in the folding factor increases latency and thus decreases throughput from the designs with wider data paths.

A 8-bit AES processor for FPGAs is designed in [9], capable of 128-bit-key encryption and decryption. The data path consists of an S-box and a GF multiplier/accumulator. The execution is controlled with a program stored in ROM. RAM is used as data memory. The design is fairly inefficient as the cycle count is significantly higher than e.g. in [15] with not much lower FPGA resource consumption.

According to our knowledge, our encryption-only core presented in [20] is the most efficient one of reported 8-bit AES implementations in terms of area-throughput ratio. This is due to the novel data path architecture that is based on the 8-bit permutation structure proposed in [21]. In [9,11,12], the AES round operations as well as the round-key generation operations are performed sequentially. In our design, the operations are performed in parallel (for different 8-bit pieces of data/key), which considerably decreases the total cycle count and increases the throughput. Still, we succeeded in maintaining the hardware area and the power consumption low. The gate area is at the same level with [11,12]. The achieved cycle count of 160 can be seen as the minimum for an iterated 8-bit AES implementation. We have estimated that including the decryption functionality would add about 25% to the total area.

Only [12,20] of the ASIC references include power consumption measures. For [12] the power consumption is 45  $\mu\text{W}/\text{MHz}$  and for the area-optimized implementation of [20] 37  $\mu\text{W}/\text{MHz}$ . In [12], the power has been measured from a manufactured chip. However, the higher throughput of [20] potentially results in considerably lower energy consumption per processed block. For achieving equal throughputs, [20] can be run at considerably lower clock frequency.

### 3.1 WSN Suitability of Dedicated Hardware Implementations

According to the survey, the best suited approaches for the hardware implementation of AES in WSN nodes seem to be the 8-bit designs [12,20]. They result in the lowest hardware area (i.e. cost). Their power consumptions are presumably also among the lowest even though power has not been reported for the other designs. Even though [20] includes only encryption functionality, it is still usable in real WSNs: decryption functionality of the AES core itself is not often required in commonly used security processing schemes. For example, this is the case in the standardized WSN technologies [7,3].

In addition to these two 8-bit designs, the 32-bit implementations of [8] can also be suitable for WSN nodes. The hardware areas are low and the area-throughput ratios high. The 32-bit cores can be combined with the encryption-mode wrapper of [14] for efficient security processing in the standard WSN technologies. Considering FPGAs, the T-box method of [13] seems to be the best approach for resource-efficient implementations. However, FPGA technologies are currently not feasible solutions for WSN nodes due to their high power consumption.

## 4 Specialized Processor Architectures for AES

An effective performance-area trade-off between dedicated hardware implementations and general-purpose processors can be achieved with programmable specialized processors. Such Application Specific Instruction set Processors (ASIP) are typically general-purpose but they have also been tailored to support a specific application domain or task. A large part of the proposals in the cryptographic domain have concentrated on maximizing performance and programmability [17], which often results in high power consumption and cost and thus makes the proposals unsuited for WSN nodes. In this section we review processor architectures proposed for efficient AES execution in low-cost devices, shown in Table 2.

The ASIP implementation reported in [22] uses the Xtensa configurable processor architecture from Tensilica. In the paper, the execution of cryptographic algorithms, including AES, is accelerated by extending the instruction set of the processor with algorithm-specific instructions. As a result, the performance is improved by several ten-folds from the original (however, the original implementations are poor). The achieved throughput for AES is 17 Mbit/s at 188 MHz in a 0.18  $\mu\text{m}$  ASIC technology. Area or power figures have not been reported.

An ASIP architecture based on the 32-bit MIPS processor architecture has been published in [23]. A special unit supporting fast LUT functionality is included for accelerating the RC4 and AES algorithms. The unit consists of two 1024 $\times$ 32-bit RAMs implying large area. For accelerating Data Encryption Standard (DES), [23] proposes a very large configurable permutation unit consisting of 512 32 $\times$ 1-bit multiplexers. The achieved throughput for AES is around 64 Mbit/s at 100 MHz. The size of the processor core is 6.25 mm<sup>2</sup> in a 0.18  $\mu\text{m}$  ASIC technology. The power is approximately 90 mW.

In [26], an instruction set extension has been developed for accelerating AES in 32-bit processors. The custom instruction performs the SubBytes (or its inverse) operation using a special functional unit. The unit has been integrated into a LEON-2 processor prototyped in an FPGA. The resulting encryption speedup is up to 1.43 and the code size reduction 30–40%. The area of the unit is 400 gates in a 0.35  $\mu\text{m}$  ASIC technology. The absolute value for the throughput and the complete processor size in the ASIC technology have not been reported.

We have utilized a processor architecture called Transport Triggered Architecture (TTA) to develop an area-efficient ASIP design for accelerating the execution of RC4 and AES in [24]. In addition to the standard functional units, the processor includes four 256 $\times$ 8-bit RAM-based LUT units, a 32-bit unit for converting between byte and word

**Table 2.** Specialized processor architectures for AES execution

Design	Technology	Area	Clock [MHz]	Throughput [Mbit/s]	Power [mW/MHz]
Ref. [22]	.18 $\mu\text{m}$	n/a	188	17	n/a
Ref. [23]	.18 $\mu\text{m}$	6.25 mm <sup>2</sup>	100	64	0.90
Ref. [24]	.13 $\mu\text{m}$	70 kgates	100	68	n/a
Ref. [25]	.18 $\mu\text{m}$	2.25 mm <sup>2</sup>	14	1.8	1.2

representations of the AES State, and a unit for performing a combined 32-bit Mix-Columns and AddRoundKey operation in a single clock cycle. The LUT units eliminate main memory accesses in the same way as the custom instruction of [26]. The size of our TTA processor that supports AES and RC4 is 69.4 kgates in a 0.13  $\mu\text{m}$  ASIC technology. The throughput is 68.5 Mbit/s for AES using precomputed roundkeys at 100 MHz. Power consumption was not evaluated in this study.

A microcoded cryptoprocessor designed for executing DES, AES, and Elliptic Curve Cryptography (ECC) has been published in [25]. The data path contains an expansion/permutation unit, a shifter, four memory-based LUTs, two logic units, and a register file consisting of sixteen 256-bit registers. The processor can be reconfigured by modifying the microcoded program and the contents of the LUTs. The encryption throughput for AES is 1.83 Mbit/s at 13.56 MHz with on-the-fly key expansion. The hardware area in a 0.18  $\mu\text{m}$  technology is 2.25  $\text{mm}^2$  and the power consumption for AES is 16.3 mW.

#### 4.1 WSN Suitability of Specialized Processor Architectures

Compared to the most compact AES hardware implementations of Section 3, the reviewed specialized processor architectures result in significantly larger areas, lower performances, and higher power consumptions. Their benefits are in programmability and/or reconfigurability compared to dedicated hardware and in performance when compared to general-purpose processors of the same application domain.

The cost of the processor presented in [23] is high and thus it is poorly suited for WSN nodes. On the contrary, the special operation units presented in [24,26] can be used for increasing the performances of the main processors in WSN nodes. Whereas [26] dedicates its unit for AES only, the LUT unit of [24] is suited for other tasks as well. The performance results in these two papers are considerably better than in [22]. If a 32-bit general-purpose processor is considered to be used in a WSN node, the complete processor design of [24] with its special support for AES is a feasible solution. Even though the AES performance of [25] is lower than e.g. in [24], the processor can be suitable for WSN nodes which frequently need to perform also ECC computations.

## 5 Conclusions

A large part of WSN applications require cryptographic protection. Due to the constraints of WSN nodes, their cryptographic implementations should be low-cost and energy-efficient. In this paper, we reviewed hardware architectures proposed for AES implementations in such environments. The survey considered both dedicated hardware and specialized processor designs. According to our survey, currently 8-bit dedicated hardware designs seem to be the most feasible solutions for WSN nodes. Alternatively, compact special functional units can be used for extending the instruction sets of WSN node processors for efficient AES execution. The reviewed designs often offer significantly higher throughput at their maximum clock speed than what is actually required for WSN communications. Hence, considerable power savings can be achieved by decreasing the clock speed from its maximum without affecting the wireless data rates

of nodes. We believe that the review presented in this paper is valuable for designers evaluating and developing AES implementations for environments in which low cost and low power consumption are key requirements, beyond WSNs as well.

## References

1. Stankovic, J.A., Abdelzaher, T.F., Lu, C., Sha, L., Hou, J.C.: Real-time communication and coordination in embedded sensor networks. *Proceedings of the IEEE* 91(7), 1002–1022 (2003)
2. Hämäläinen, P., Kuorilehto, M., Alho, T., Hännikäinen, M., Hämäläinen, T.D.: Security in wireless sensor networks: Considerations and experiments. In: *Proc. Embedded Computer Systems: Architectures, Modelling, and Simulation (SAMOS VI) Workshop—Special Session on Wireless Sensor Networks*, Samos, Greece, pp. 167–177 (July 17–20, 2006)
3. ZigBee Alliance: ZigBee Specification Version 1.0 (December 2004)
4. Suhonen, J., Kohvakka, M., Hännikäinen, M., Hämäläinen, T.D.: Design, implementation, and experiments on outdoor deployment of wireless sensor network for environmental monitoring. In: *Proc. Embedded Computer Systems: Architectures, Modelling, and Simulation (SAMOS VI) Workshop—Special Session on Wireless Sensor Networks*, Samos, Greece, pp. 109–121 (July 17–20, 2006)
5. Avancha, S., Undercoffer, J., Joshi, A., Pinkston, J.: Security for Wireless Sensor Networks. In: *Wireless Sensor Networks*, 1st edn. pp. 253–275. Springer, Heidelberg (2004)
6. National Institute of Standards and Technology (NIST): Advanced Encryption Standard (AES), FIPS-197 (2001)
7. IEEE: IEEE Standard for Local and Metropolitan Area Networks—Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPAN), IEEE Std 802.15.4 (2003)
8. Satoh, A., Morioka, S., Takano, K., Munetoh, S.: A compact Rijndael hardware architecture with S-box optimization. In: Boyd, C. (ed.) *ASIACRYPT 2001*. LNCS, vol. 2248, pp. 239–254. Springer, Heidelberg (2001)
9. Good, T., Benaissa, M.: AES on FPGA from the fastest to the smallest. In: Rao, J.R., Sunar, B. (eds.) *CHES 2005*. LNCS, vol. 3659, pp. 427–440. Springer, Heidelberg (2005)
10. Pramstaller, N., Mangard, S., Dominikus, S., Wolkerstorfer, J.: Efficient AES implementations on ASICs and FPGAs. In: *Proc. 4th Conf. on the Advanced Encryption Standard (AES 2004)*, Bonn, Germany, May 10–12, 2005, pp. 98–112 (2005)
11. Feldhofer, M., Dominikus, S., Wolkerstorfer, J.: Strong authentication for RFID systems using the AES algorithm. In: Joye, M., Quisquater, J.-J. (eds.) *CHES 2004*. LNCS, vol. 3156, pp. 357–370. Springer, Heidelberg (2004)
12. Feldhofer, M., Wolkerstorfer, J., Rijmen, V.: AES implementation on a grain of sand. *IEE Proc. Inf. Secur.* 152(1), 13–20 (2005)
13. Chodowicz, P., Gaj, K.: Very compact FPGA implementation of the AES algorithm. In: D.Walter, C., Koç, Ç.K., Paar, C. (eds.) *CHES 2003*. LNCS, vol. 2779, pp. 319–333. Springer, Heidelberg (2003)
14. Hämäläinen, P., Hännikäinen, M., Hämäläinen, T.: Efficient hardware implementation of security processing for IEEE 802.15.4 wireless networks. In: *Proc. 48th IEEE Int. Midwest Symp. on Circuits and Systems (MWSCAS 2005)*, Cincinnati, OH, USA, August 7–10, 2005, pp. 484–487 (2005)
15. Rouvroy, G., Standaert, F.X., Quisquater, J.J., Legat, J.D.: Compact and efficient encryption/decryption module for FPGA implementation of the AES Rijndael very well suited for small embedded applications. In: *Proc. IEEE Int. Conf. on Inf. Tech.: Coding and Computing (ITCC 2004)*, Las Vegas, NV, USA, April 4–6, 2004, vol. 2, pp. 583–587 (2004)

16. Elbirt, A.J., Yip, W., Chetwynd, B., Paar, C.: An FPGA implementation and performance evaluation of the AES block cipher candidate algorithm finalists. In: Proc. 3rd AES Candidate Conf. (AES3), New York, NY, USA, April 13-14, 2000 (2000)
17. Hämäläinen, P.: Cryptographic Security Designs and Hardware Architectures for Wireless Local Area Networks. PhD thesis, Tampere Univ. of Tech. Tampere, Finland, (December 2006), Available online: [http://www.tkt.cs.tut.fi/research/daci/phd\\_hamalainenp\\_thesis.html](http://www.tkt.cs.tut.fi/research/daci/phd_hamalainenp_thesis.html)
18. Fischer, V.: Realization of the round 2 AES candidates using Altera FPGA. In: Proc. 3rd AES Candidate Conf. (AES3), New York, NY, USA, April 13-14, 2000 (2000)
19. Fischer, V., Drutarovsky, M.: Two methods of Rijndael implementation in reconfigurable hardware. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 77–92. Springer, Heidelberg (2001)
20. Hämäläinen, P., Alho, T., Hännikäinen, M., Hämäläinen, T.D.: Design and implementation of low-area and low-power AES encryption hardware core. In: Proc. 9th Euromicro Conf. Digital System Design (DSD 2006), Cavtat, Croatia (August 30-September 1, 2006), pp. 577–583 (2006)
21. Järvinen, T., Salmela, P., Hämäläinen, P., Takala, J.: Efficient byte permutation realizations for compact AES implementations. In: Proc. 13th European Signal Processing Conf. (EU-SIPCO 2005), Antalya, Turkey, September 4-8, 2005 (2005)
22. Ravi, S., Raghunathan, A., Potlapally, N., Sankaradass, M.: System design methodologies for a wireless security processing platform. In: Proc. 39th Design Automation Conf. New Orleans, LA, USA, June 10-14, 2002, pp. 777–782 (2002)
23. Lewis, M., Simmons, S.: A VLSI implementation of a cryptographic processor. In: Proc. Canadian Conf. Electrical and Computer Engineering (CCECE 2003), Montreal, Canada, May 4-7, 2003, pp. 821–826 (2003)
24. Hämäläinen, P., Heikkinen, J., Hännikäinen, M., Hämäläinen, T.D.: Design of transport triggered architecture processors for wireless encryption. In: Proc. 8th Euromicro Conf. Digital System Design (DSD 2005), Porto, Portugal, August 30-September 3, 2005, pp. 144–152 (2005)
25. Eslami, Y., Sheikholeslami, A., Gulak, P.G., Masui, S., Mukaida, K.: An area-efficient universal cryptography processor for smart cards. *IEEE Trans. VLSI Systems* 14(1), 43–56 (2006)
26. Tillich, S., Grossschädl, J., Szekely, A.: An instruction set extension for fast and memory-efficient AES implementation. In: Dittmann, J., Katzenbeisser, S., Uhl, A. (eds.) CMS 2005. LNCS, vol. 3677, pp. 11–21. Springer, Heidelberg (2005)