Survey and Benchmark of Lightweight Block Ciphers for Wireless Sensor Networks*

Mickaël Cazorla, Kevin Marquet and Marine Minier
Universit de Lyon, INRIA
INSA-Lyon, CITI-INRIA, F-69621, Villeurbanne, France
firstname.name@insa-lyon.fr

Keywords: Lightweight Block Ciphers, Sensors, Benchmarks.

Abstract: For security applications in wireless sensor networks (WSNs), choosing best algorithms in terms of energy-efficiency and of small memory requirements is a real challenge because the sensor networks must be autonomous. In (Eisenbarth et al., 2012; Law et al., 2006), the authors have benchmarked on a dedicated platform some block-ciphers and have deduced the best candidates to use in the context of small embedded platforms. This article proposes to study on a dedicated platform of sensors most of the recent lightweight block ciphers as well as some conventional block ciphers. First, we describe the design of the chosen block ciphers with a security summary and we then present some implementation tests performed on our platform.

1 INTRODUCTION

Wireless Sensor Networks (WSNs) are composed of numerous low-cost, low-energy sensor nodes communicating at short distance through wireless links. Sensors are densely deployed to collect and transmit data of the physical world to one or few destinations called the sinks using multi-hop routing protocols. Wireless sensor networks can be really useful in many civil and military areas for collecting, processing and monitoring environmental data. A sensor node contains an integrated sensor, a microprocessor, some memories, a transmitter and an energy battery. Despite the relative simplicity of its basic components, sensor networking offers a great diversity: various hardwares (MicaZ, Telos, SkyMote, AVR or TI micro-controllers), various radio and physical layers (868MHz and 2,4GHz) using different types of modulations, various OS (TinyOS, Contiki, FreeRTOS, JITS), various constraints (real-time, energy, memory or processing), various applications (military or civil uses).

In such a context, a specific care must be invested in the design of the applications, communication protocols, operating systems and of course security protocols that will be used. Lots of protocols have been proposed to enforce the security offered by sensor networks. Despite the increasing request in this new area of research, few articles present results of real software implementations or benchmarks concerning the security primitives which can be used in sensor networks. In (Eisenbarth et al., 2012; Law et al., 2006), the authors present such results. In (Law et al., 2006), the authors present benchmark results on a MSP430, a TI 16 bits microcontroller, comparing the most famous block ciphers (including AES, MISTY1, Skipjack,...) and the different possible modes of operations. In (Eisenbarth et al., 2012), the authors present benchmark results on a ATtiny device, a 8 bits microcontroller, of 12 block ciphers, 8 lightweight block ciphers and 4 conventional block ciphers. They also introduce a comparison metric that takes into account both the code size and the cycle count.

This article presents performance results for 17 block ciphers (most of the recent lightweight block ciphers and some conventional block ciphers). This performance results are obtained by testing the dedicated implementations of the algorithm on a MSP430[2], a TI 16 bits microcontroller which is the corner stone of the nodes WSN430[3] used in the deployed Senslab platform[4] (des Roziers et al., 2011).

This paper is organized as follows: Section 2

*This work was partially supported by the French National Agency of Research: ANR-11-INS-011.

[2]http://www.ti.com/product/msp430f1611

[3]http://perso.ens-lyon.fr/eric.fleury/Upload/wsn430-docbook/

[4]http://www.senslab.info/

Table 1: Studied block ciphers. *Nb* means the size of the input/output block in bits; *Nk* means the size of the key in bits. * designates the conventional block ciphers in opposite to lightweight block ciphers.

NAME (*Nb*/*Nk*)	Reference	Struct.	Nb rounds
AES-128* (128/128)	(FIPS 197, 2001)	SPN	10
CLEFIA-128* (128/128)	(Shirai et al., 2007)	Feistel	18
DESXL (64/184)	(Leander et al., 2007)	Feistel	16
HIGHT (64/128)	(Hong et al., 2006)	Feistel	32
IDEA* (64/128)	(Lai and Massey, 1990)	Lai-Massey	8.5
KATAN & KTANTAN (32, 48, 64/80)	(Cannière et al., 2009)	Stream	254
KLEIN (64/64, 80 and 96)	(Gong et al., 2011)	SPN	12, 16, 20
LBLOCK (64/80)	(Wu and Zhang, 2011)	Feistel	32
LED (64/64 and 128)	(Guo et al., 2011)	SPN	32/48
mCrypton (64/64, 96 and 128)	(Lim and Korkishko, 2005)	SPN	12
MIBS (64/64 and 80)	(Izadi et al., 2009)	Feistel	32
Noekeon* (128/128)	(Daemen et al., 2000)	SPN	16
Piccolo (64/80 and 128)	(Shibutani et al., 2011)	Feistel	25/31
PRESENT (64/80 and 128)	(Bogdanov et al., 2007)	SPN	31
TEA & XTEA (64/128)	(Wheeler and Needham, 1994)	Feistel	64
TWINE (64/ 80 and 128)	(Suzaki et al., 2012)	Feistel	36
SEA (96/96,...)	(Standaert et al., 2006)	Feistel	Var.
SKIPJACK* (64/80)	(NIST, 1998)	Feistel	32

presents the studied block ciphers. Section 3 presents the dedicated platform and describes the methodology used to perform our benchmarks. Section 4 provides our results and our analysis concerning the benchmarking whereas Section 5 concludes this paper.

2 THE STUDIED BLOCK CIPHERS

Our benchmarks concern 17 block ciphers, 12 are lightweight and 5 are conventional block ciphers. Studied block ciphers are listed in Table 1 in alphabetic order.

The main differences between the conventional block ciphers and the lightweight block ciphers are centered on: the block size which is in general 32, 48 or 64 bits for a lightweight block cipher and equal to 64 or 128 bits for a conventional block cipher; the same remark also holds for the different possible key sizes (smaller for lightweight block ciphers); Lightweight block ciphers also rely more on elementary operations (such as binary XOR, binary AND, etc.) leading in an increase of required number of rounds; Lightweight block ciphers generally extremely simplify the key schedule due to memory requirements.

For a complete overview of each implemented block cipher from a design point of view (without describing the key schedule) and from a cryptanalytic point of view (we limit our state of art in the case of unknown key settings and of related key settings, we do not describe attacks in the known or chosen key settings), the reader could refer to (Cazorla et al., 2013).

3 METHODOLOGY

In this section, we present the platform used to perform the benchmarks and we also describe the testing framework.

3.1 The Dedicated Platform

The MSP430 is a Texas Instrument microcontroller running with an external 8MHz clock. This microcontroller is programmable via a JTAG connection. It integrates a 48 KBytes flash memory, a 10 Koctets RAM memory, 48 configurable Inputs/Outputs, 12-bit analog-to-digital conversion pins, a watchdog, 2 serial communication ports and 2 configurable timers. This microcontroller is compatible with most of real-time operating systems such as FreeRTOS.

All the codes were written in C. We used the GCC toolchain for MSP430 family to flash programs into the microcontroller. This includes the GNU C compiler (GCC), the assembler and linker (binutils), the debugger (GDB), and some other tools needed to make a complete development environment for the MSP430. These tools can be used on Windows, Linux, BSD and most other flavors of Unix. We used msp430-gcc version 4.6.3.

Table 2: Software performance.

Algorithm	Block Size (bits)	Enc.+key: cycle count	Enc.+key : cycles/byte	Dec.+key: cycle Count	Dec.+key: cycles/byte
AES	128	30257	1891	38508	2406
CLEFIA128	128	98145	6134	101855	6365
CLEFIA192	128	150314	9394	123333	7708
CLEFIA256	128	155658	9728	145291	9080
DESXL	64	26055	3256	66913	8364
DIRnoekeon	128	26291	1643	27129	1695
HIGHT	64	32372	4046	32623	4077
IDEA	64	31402	3925	163380	20422
INDnoekeon	128	52564	3285	53435	3339
KATAN32	32	744279	186069	717056	179264
KATAN48	48	1127271	187878	1053680	175613
KATAN64	64	1518391	189798	1397924	174740
KLEIN64	64	29514	3689	100600	12575
KLEIN80	64	40278	5034	135369	16921
KLEIN96	64	51502	6437	170789	21348
KTANTAN32	32	10233211	2558302	10193489	2548372
KTANTAN48	48	10614933	1769155	10525067	1754177
KTANTAN64	64	11004783	1375597	10864265	1358033
LBlock	64	42954	5369	22005	2750
LED128	64	1341488	167686	1345152	168144
LED128_tcalc	64	268721	33590	274953	34369
LED128_tdur	64	171056	21382	173832	21729
LED64	64	894680	111835	897352	112169
LED64_tcalc	64	212409	26551	217401	27175
LED64_tdur	64	114872	14359	116280	14535
MCRYPTON64	64	107803	13475	219870	27483
MCRYPTON96	64	108499	13562	220320	27540
MCRYPTON128	64	108415	13551	220568	27571
MIBS64	64	49056	6132	52890	6611
MIBS80	64	58688	7336	39842	4980
PRESENT_SIZE	64	491602	61450	489813	61226
PRESENT_SPEED	64	364587	45573	368731	46091
Piccolo128	64	36497	4562	39600	4950
Piccolo80	64	32106	4013	34630	4328
SEA	96	119455	9954	120158	10013
SKIPJACK	64	84923	10615	123368	15421
TEA	64	8785	1098	9129	1141
TWINE	128	82003	5125	60932	3808
XTEA	64	9287	1160	9631	1203

3.2 Methodology

We measured the performance of the algorithms as well as the memory consumption. To obtain the performance, we used simulator coming with mspdebug. This simulator is able to give the number of clock cycles spent at any point of the program execution. Although it is only a simulator, it is cycle-accurate and the experiments we made on real hardware confirmed the results obtained.

Concerning the memory consumption, we distinguish between the need of read-only memory (ROM) and writable memory (RAM). The ROM is used to store the code as well as tables that do not need to be modified – for instance, the F-table of skipjack. We obtain the size of the ROM needed simply by declaring as *static const* the concerned variables and getting the size of the text section in the *elf* file. In order to get the size of RAM needed, mspdebug tells us until which address the execution stack was modified.

4 RESULTS

4.1 CPU Cycles and Energy Consumption

Table 2 gives the performance of the algorithms.

4.2 Memory Requirements

Table 2 reports the memory consumption of the algorithms. It shows the requirements of read-only memory (code + read-only tables) as well as the amount of RAM needed to store the stack and modifiable data.

Table 3: Memory usage. The column RAM means RAM requirement in bytes whereas the column ROM means Size of read-only data in bytes.

Function	RAM	ROM	Function	RAM	ROM	Function	RAM	ROM
AES	19	4460	CLEFIA128	180	4780	CLEFIA192	268	5010
CLEFIA256	268	4924	DESXL	112	16816	DIRnoekeon	34	2710
HIGHT	18	3130	IDEA	82	3140	INDnoekeon	34	2784
KATAN32	1881	5816	KATAN48	1969	7076	KATAN64	1953	8348
KLEIN64	36	5486	KLEIN80	38	5676	KLEIN96	39	5862
KTANTAN32	614	10516	KTANTAN48	702	11764	KTANTAN64	790	16252
LBlock	13	3568	LED128	41	2648	LED128_tcalc	41	2948
LED128_tdur	41	2264	LED64	41	2670	LED64_tcalc	41	2498
LED64_tdur	41	2264	MCRYPTON64	18	2726	MCRYPTON96	20	2834
MCRYPTON128	24	3108	MIBS64	29	3184	MIBS80	16	3138
PRESENT_SIZE	142	4964	PRESENT_SPEED	142	4814	Piccolo128	91	2510
Piccolo80	79	2434	SEA	24	2804	SKIPJACK	19	6628
TEA	13	1354	TWINE	23	2216	XTEA	11	1394

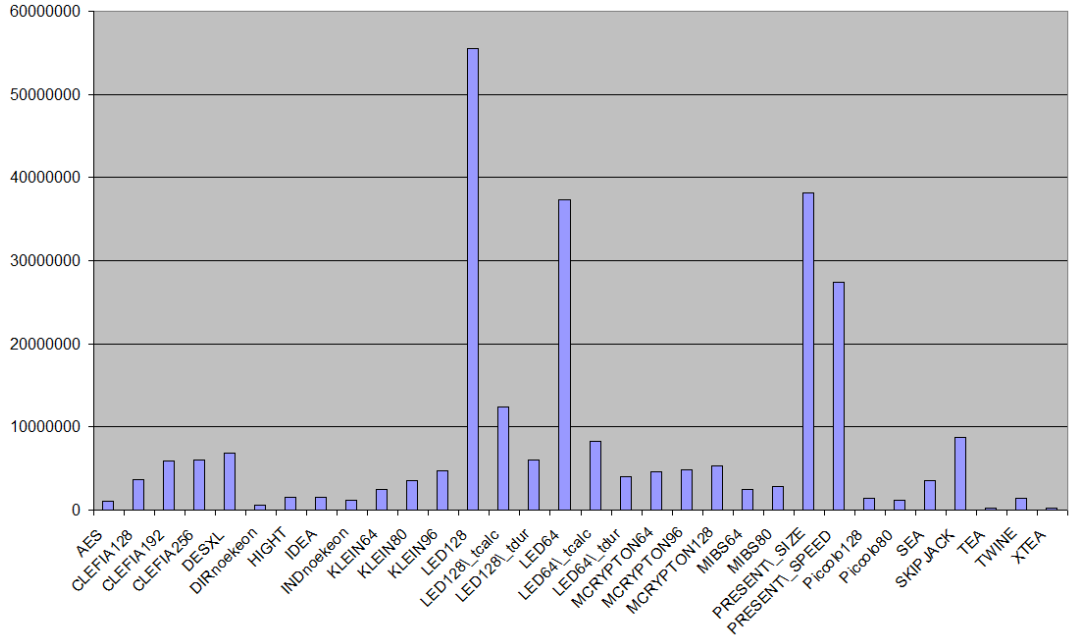

Figure 1: Metric introduced in (Eisenbarth et al., 2012): code size $\times$ cycle count product/block size.

We can see that the requirement of RAM is very similar and very small, except for the CLEFIA and the KATAN families. The memory requirements of these functions is due to the use of large tables in the key scheduling phase.

On the contrary, the need of read-only memory is very different from one algorithm to an other. Whereas TEA and XTEA requires only 1354 and 1394 bytes of ROM to execute, KTANTAN requires 16252 bytes in its 64-bits version. The ROM consumption of the KATAN family is due to the tables used to store the bitfields (see Section 2).

4.3 Analysis

We consider 6 different metrics here: cycle count for enc.+key and for dec.+key, cycles/bytes for enc.+key and for dec.+key, code size (in bytes), RAM use (in bytes) and the metric introduced in (Eisenbarth et al., 2012) that is code size $\times$ cycle count product, normalized by the block size (see Fig. 1). We detail in this Section some particular observations.

First, due to sensor memory requirement, we consider compact implementations. As shown in Table 3, TEA ans XTEA have memory size less than 1500 bytes whereas NOEKEON, LED mCrypton, Piccolo, SEA and TWINE have memory footprint between 2000 and 3000 bytes which is really reasonable. At the contrary, all the KATAN and KTANTAN version have huge memory footprints due to their particular design which has the same cost when enciphering/deciphering 32, 48 or 64 blocks in parallel. In

terms of RAM occupancy, HIGHT, LBlock, mCrypton, MIBS, Skipjack, TEA and XTEA require less than 20 bytes of RAM which is really performing.

Concerning performance, TEA, XTEA and the AES are the only ones that require less than 2000 cycles/byte. Some lightweight designs have poor performance: KATAN, KTANTAN, LED, mCrypton and PRESENT whereas the others (DESXL, NOEKEON, HIGHT, KLEIN, LBlock, Piccolo, TWINE) use less than 5500 cycles/bytes. IDEA is efficient in encryption but as expected and due to the key schedule inefficient in decryption.

Lastly, the combined metric in Figure 1 first shows the excellent size vs. performance trade-off offered by the AES. Among the low-cost ciphers, NOEKEON, TEA and XTEA have also an excellent behavior. In the same way, HIGHT, Piccolo and TWINE provide good trade-offs whereas KATAN and KTANTAN are not present in the Figure due to their too bad behaviors.

5 CONCLUSIONS

We have presented here some benchmarks performed on lightweight block ciphers, the traditional ones and the new ones on a dedicated platform which is a sensor. In total, 17 ciphers have been implemented and analyzed keeping in mind that the compactness is an important issue in the sensor world. They show that some well-suited block ciphers such as Piccolo, TWINE, XTEA or the AES have good performance considering the trade-off between code size and cycle count. We also see that most of the ciphers specially dedicated to hardware (such as LED, PRESENT or KATAN and KTANTAN) have poor results.

REFERENCES

Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J. B., Seurin, Y., and Vikkelsoe, C. (2007). PRESENT: An Ultra-Lightweight Block Cipher. In *Cryptographic Hardware and Embedded Systems - CHES 2007*, LNCS 4727, pages 450–466. Springer.

Cannière, C. D., Dunkelman, O., and Knezevic, M. (2009). Katan and ktantan - a family of small and efficient hardware-oriented block ciphers. In *Cryptographic Hardware and Embedded Systems - CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 272–288. Springer.

Cazorla, M., Marquet, K., and Minier, M. (2013). Survey and benchmark of lightweight block ciphers for wireless sensor networks. Cryptology ePrint Archive, Report 2013/??? http://eprint.iacr.org/.

Daemen, J., Peeters, M., Assche, G. V., and Rijmen, V. (2000). Nessie proposal: Noekeon. Submitted as an NESSIE Candidate Algorithm. http://gro.noekeon.org/.

des Roziers, C. B., Chelius, G., Ducrocq, T., Fleury, E., Fraboulet, A., Gallais, A., Mitton, N., Noël, T., and Vandaele, J. (2011). Using senslab as a first class scientific tool for large scale wireless sensor network experiments. In *NETWORKING 2011*, volume 6640 of *Lecture Notes in Computer Science*, pages 147–159. Springer.

Eisenbarth, T., Gong, Z., Güneysu, T., Heyse, S., Indesteege, S., Kerckhof, S., Koeune, F., Nad, T., Plos, T., Regazzoni, F., Standaert, F.-X., and van Oldeneel tot Oldenzeel, L. (2012). Compact implementation and performance evaluation of block ciphers in attiny devices. In *Progress in Cryptology - AFRICACRYPT 2012*, volume 7374 of *Lecture Notes in Computer Science*, pages 172–187. Springer.

FIPS 197 (2001). Advanced Encryption Standard. Federal Information Processing Standards Publication 197. U.S. Department of Commerce/N.I.S.T.

Gong, Z., Nikova, S., and Law, Y. W. (2011). Klein: A new family of lightweight block ciphers. In *RFID. Security and Privacy - RFIDSec 2011*, volume 7055 of *Lecture Notes in Computer Science*, pages 1–18. Springer.

Guo, J., Peyrin, T., Poschmann, A., and Robshaw, M. (2011). The led block cipher. In *Cryptographic Hardware and Embedded Systems - CHES 2011*, volume to appear of *LNCS*. Springer.

Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., and Chee, S. (2006). HIGHT: A New Block Cipher Suitable for Low-Resource Device. In *Cryptographic Hardware and Embedded Systems - CHES 2006*, LNCS 4249, pages 46–59. Springer.

Izadi, M., Sadeghiyan, B., Sadeghian, S. S., and Khanooki, H. A. (2009). MIBS: A New Lightweight Block Cipher. In *Cryptology and Network Security - CANS 2009*, LNCS 5888, pages 334–348.

Lai, X. and Massey, J. L. (1990). A proposal for a new block encryption standard. In *Advances in Cryptology - EUROCRYPT '90*, volume 473 of *Lecture Notes in Computer Science*, pages 389–404. Springer.

Law, Y. W., Doumen, J., and Hartel, P. H. (2006). Survey and benchmark of block ciphers for wireless sensor networks. *TOSN*, 2(1):65–93.

Leander, G., Paar, C., Poschmann, A., and Schramm, K. (2007). New lightweight des variants. In *Fast Software Encryption - FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 196–210. Springer.

Lim, C. H. and Korkishko, T. (2005). mCrypton - A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors. In *Workshop on Information Security Applications - WISA 2005*, LNCS 3786, pages 243–258. Springer Verlag.

NIST (1998). Skipjack and kea algorithm specification. Technical Report. http://csrc.nist.gov/groups/STM/cavp/documents/skipjack/skipjack.pdf.

Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., and Shirai, T. (2011). Piccolo: An ultra-lightweight blockcipher. In *Cryptographic Hardware*

and Embedded Systems - CHES 2011, volume 6917 of *LNCS*. Springer.

Shirai, T., Shibutani, K., Akishita, T., Moriai, S., and Iwata, T. (2007). The 128-bit blockcipher clefia (extended abstract). In *Fast Software Encryption - FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 181–195. Springer.

Standaert, F.-X., Piret, G., Gershenfeld, N., and Quisquater, J.-J. (2006). Sea: A scalable encryption algorithm for small embedded applications. In *Smart Card Research and Advanced Applications - CARDIS 2006*, volume 3928 of *Lecture Notes in Computer Science*, pages 222–236. Springer.

Suzaki, T., Minematsu, K., Morioka, S., and Kobayashi, E. (2012). Twine: A lightweight block cipher for multiple platforms. In *Selected Areas in Cryptography - SAC 2012*, volume to appear of *Lecture Notes in Computer Science*. Springer.

Wheeler, D. J. and Needham, R. M. (1994). Tea, a tiny encryption algorithm. In *Fast Software Encryption - FSE 94*, volume 1008 of *LNCS*, pages 363–366. Springer.

Wu, W. and Zhang, L. (2011). Lblock: A lightweight block cipher. In *Applied Cryptography and Network Security - ACNS 2011*, volume 6715 of *LNCS*, pages 327–344. Springer.