SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2015; **8**:3564–3579
Published online 20 May 2015 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.1281

RESEARCH ARTICLE

Survey and benchmark of lightweight block ciphers for MSP430 16-bit microcontroller

Mickaël Cazorla, Sylvain Gourgeon, Kévin Marquet and Marine Minier*

Université de Lyon, INRIA, INSA-Lyon, CITI-INRIA, F-69621, Villeurbanne, France

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 are composed of low-power entities. In some previous works, 12 block-ciphers have been benchmarked on an ATMEL AVR ATtiny45 8-bit microcontroller and the best candidates to use in the context of small embedded platforms have been deduced. This article proposes to study on the TI 16-bit microcontroller MSP430 most of the recent lightweight block cipher proposals as well as some conventional block ciphers. First, we describe the design of the chosen block ciphers with a security and an implementation summary and we then present some implementation tests performed on our dedicated platform. Copyright © 2015 John Wiley & Sons, Ltd.

KEYWORDS
lightweight block ciphers; sensors; benchmarks

***Correspondence**
Marine Minier, CITI-INRIA, Université de Lyon, INRIA, INSA-Lyon, Lyon, France.
E-mail: marine.minier@insa-lyon.fr

1. INTRODUCTION

Wireless sensor networks (WSNs) are composed of numerous low-cost, low-energy sensor nodes communicating at a 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 integrated sensors, 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 hardware (MicaZ, Telos, SkyMote, AVR, or Texas Instruments microcontrollers), various radio and physical layers (868 MHz and 2.4 GHz) using different types of modulations, various operating systems (TinyOS, Contiki, FreeRTOS, or JITS), various constraints (real time, energy, memory, or processing), and various applications (military or civil uses).

In such a context, specific care must be invested in the design of the applications, the communication protocols, the operating systems and of course the 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 [1], the authors present benchmarking results on a 16-bit microcontroller including a MSP430 processor and compare the most famous block ciphers (including AES, MISTY1, and SKIPJACK) and the different possible modes of operations. In [2], the authors present benchmark results on a ATtiny device, an 8-bit microcontroller, of 12 block ciphers: eight lightweight block ciphers and four conventional block ciphers. They also introduce a comparison metric that takes into account both the code size and the cycle count.

This paper sums up the theoretical security provided by most of the recent lightweight block ciphers as well as some conventional block ciphers and presents benchmarking results performed on Texas Instruments 16-bit microcontroller based on an MSP430 processor[†], as the corner stone of the nodes WSN430[‡] used in the deployed IoT-lab platform[§] [3]. In total, 21 block ciphers (16 are lightweight and five are conventional) were investigated: First, we implemented those ciphers.

[†] http://www.ti.com/product/msp430fg4618
[‡] http://perso.ens-lyon.fr/eric.fleury/Upload/wsn430-docbook/
[§] https://www.iot-lab.info/

Table I. Studied block ciphers.

NAME (*Nb*/*Nk*)	Reference	Structure	*Nb* rounds	Best attack *Nb* rounds	Best attack complexity
AES-128* (128/128)	[4]	SPN	10	7	2^{100}
CLEFIA-128* (128/128)	[5]	Feistel	18	13	$2^{116.16}$
DESXL (64/184)	[6]	Feistel	16	N/A	N/A
HIGHT (64/128)	[7]	Feistel	32	27	$2^{128-\epsilon}$
IDEA* (64/128)	[8]	Lai-Massey	8.5	6	$2^{128-\epsilon}$
KATAN (32, 48, 64/80)	[9]	Stream	254	120	Practical
KTANTAN (32, 48, 64/80)	[9]	Stream	254	254	$2^{74.4}$
KLEIN (64/64, 80 and 96)	[10]	SPN	12, 16, 20	12/13/14	$2^{57}/2^{76}/2^{89}$
LBlock (64/80)	[11]	Feistel	32	23	$2^{75.36}$
LED (64/64 and 128)	[12]	SPN	32/48	12/32	$2^{60}/2^{123.8}$
mCrypton (64/64, 96 and 128)	[13]	SPN	12	8	$2^{81.6}$
MIBS (64/64 and 80)	[14]	Feistel	32	19	$2^{74.23}$
Noekeon* (128/128)	[15]	SPN	16	16	Weak related keys
Piccolo (64/80 and 128)	[16]	Feistel	25/31	14/21	$2^{73}/2^{121}$
PRESENT (64/80 and 128)	[17]	SPN	31	27	Full codebook
PRINCE (64/128)	[18]	SPN	12	10	$2^{118.56}$
SEA (96/96,...)	[19]	Feistel	Var.	$nr/2$	$2^{Nk-\epsilon}$
SIMON (64,96,128/80,128,144)	[20]	Feistel	Var.	20/21	$2^{60}/2^{63}/2^{73}$
SKIPJACK* (64/80)	[21]	Feistel	32	31	$2^{80-\epsilon}$
SPECK (64,96,128/80,128,144)	[20]	Feistel	Var.		
TWINE (64/ 80 and 128)	[22]	Feistel	36	23/25	$2^{80-\epsilon}/2^{128-\epsilon}$
XTEA (64/128)	[23]	Feistel	64	25	$2^{120.4}$

SPN, substitution–permutation network.
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 opposite to lightweight block ciphers.

All the codes are open source and available online through our GitHub repository[¶] [24]. Second, we benchmarked them on our dedicated platform. Some preliminary results were presented in [25]. The results given in [25] include only 17 ciphers and were obtained using a simulator, whereas the performance evaluations given here were obtained on the real platform.

This paper is organized as follows: Section 2 presents the 21 block ciphers, evaluates their current security based on the most recent cryptanalytic results and specifies when required the implementation choices. 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 21 block ciphers: 16 are lightweight and five are conventional. Studied block ciphers are listed in Table I in alphabetical order.

The main differences between the conventional block ciphers and the lightweight block ciphers are centered on the following:

- The block size 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 holds for the different possible key sizes (80 or 128 bits for a lightweight block cipher and 128 bits or more for a conventional one except for SKIPJACK, which is a very old block cipher).
- Lightweight block ciphers are made of elementary operations (such as binary for Exclusive-or (XOR) and logical binary (AND)) leading in increasing the number of rounds.
- Lightweight block ciphers generally extremely simplify the key schedule because of memory requirements.

In this section, we give a quick 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). Table I sums up the best attacks in the single key settings against each block cipher excluding the biclique attacks, which are dedicated to improve the exhaustive key search. We also provide some words about the way we implemented them when required.

AES-128. The AES is the current block cipher standard [4] designed by J. Daemen and V. Rijmen in 1997 and chosen as a standard in 2000. It is the most used block

[¶] http://bloc.project.citi-lab.fr/library.html.

cipher. The AES is an iterated block cipher based on a substitution–permutation network (SPN) structure that ciphers blocks of size 128 bits under 128-bit, 192-bit, or 256-bit keys. We focus here on the case of AES-128 that ciphers 128-bit blocks under a key of length 128 bits. This AES version is composed of 10 rounds that repeat four elementary mappings (`SubBytes`, `ShiftRows`, `MixColumns`, and `AddRoundKey`) on blocks seen as 4×4 byte matrices.
Security. The main security result against the AES-128 is a biclique cryptanalysis due to A. Bogdanov, D. Khovratovich, and C. Rechberger [26]. It improves the exhaustive key search using particular relations linking together the keys through the key schedule and some bytes on internal states. The time complexity of this attack on the full AES-128 version is equal to $2^{126.2}$ AES-128 encryptions, whereas the memory requirements are small and the amount of data is equal to 2^{88}. The other interesting cryptanalytic result is due to P. Derbez, P.-A. Fouque, and J. Jean in [27] that provide a dedicated meet-in-the-middle attack on the seven rounds of AES-128 where data/time/memory complexities are below 2^{100}.
Implementation. Our implementation is without tables for the `ShiftRows` and `MixColumns` operations and uses a matrix of bytes.

CLEFIA-128. CLEFIA is a conventional block cipher designed by Sony and described in [5]. It has been created to achieve good performances both in hardware and in software. It ciphers blocks of length 128 bits under keys of length 128, 192, or 256 bits. This cipher is based on a generalized Feistel structure with four data lines with two 32-bit F-functions per round. The number of rounds depends on the key length and is equal to 18, 22, or 26 according to the key size. The two F-functions call two different 8-bit S-boxes followed by a diffusion matrix multiplication inspired from the AES `MixColumns` operation. We focus our study on CLEFIA-128 with a 128-bit key.
Security. In [28], the authors present an impossible differential cryptanalysis against CLEFIA-128. With their method, they could build an impossible differential attack against CLEFIA-128 up to 13 rounds with a time complexity equal to $2^{116.16}$ encryptions.
Implementation. To store the cipher and the key, we use arrays of 8-bit numbers. The rest of the implementation follows the original specification.

DESL and DESXL. DESL and DESXL are two lightweight variants of the data encryption standard (DES) proposed by G. Leander, C. Paar, A. Poschmann, and K. Schramm in [6]. The main idea is to simplify the DES round function using a single S-box instead of eight and to discard the initial and final permutations of the DES to limit the size of the hardware implementation. So, DESL iterates 16 rounds of a classical Feistel network and takes a block of size 64 bits as the DES under a key of size 56 bits, whereas DESXL uses, as DESX, a whitening method to reinforce the security with a key of length 184 \bits with 64-bit blocks. We only consider DESXL in our implementations for clear security reasons.
Security. Up to our knowledge, no attack has been exhibited against DESL and DESXL. It seems logical as the authors of [6] repaired all the known weaknesses of the DES, especially by choosing a new well-suited S-box.
Implementation. To store the block to cipher and the key, we use arrays of 8-bit numbers. We used particular tables to simplify the key schedule.

HIGHT. HIGHT is a dedicated lightweight block cipher proposed at Cryptographic Hardware and Embedded Systems (CHES) 2006 [7]. It takes blocks of size 64 bits under keys of length 128 bits. It iterates 32 rounds of a modified eight-branch Feistel network where the XOR operation is sometimes replaced by a modular addition. The two internal functions of the Feistel network consist in XOR operations combined with left or right rotations.
Security. One of the best known attacks in the unknown key settings against HIGHT is an impossible differential attack proposed in [29] against the 27 rounds of HIGHT with a complexity slightly lower than the exhaustive search. In [30], the authors propose a biclique attack (which is a dedicated cryptanalysis) against the full rounds of HIGHT with a computational complexity of $2^{126.4}$ encryptions, faster than exhaustive search based on eight-round bicliques. In [31], the authors propose a related key attack against the full rounds of HIGHT faster than a exhaustive key search using four related keys.
Implementation. The subkey generation uses constants produced by a linear feedback shift register. We use a 128-byte table to store those constants. The key and the block to cipher are stored in tables of 8-bit numbers.

IDEA. IDEA [8] is a conventional block cipher that uses 64-bit blocks with a 128-bit key. It is composed of 8.5 identical rounds. It is based on the Lai–Massey scheme and interleaves operations on 16-bit words from different groups (modular additions, modular multiplications, and XORs). It is one of the most widely used block ciphers, due to its inclusion in several cryptographic packages, such as Pretty Good Privacy (PGP).
Security. Since its publication, IDEA resisted intensive cryptanalytic efforts, and no attack on the full IDEA version exists. The most significant attack is a six-round attack faster than the exhaustive key search that exploits the weak key-schedule algorithm of IDEA and combines square-like techniques with linear cryptanalysis [32]. In [33], the authors apply and extend the biclique framework to IDEA and for the first time describe an approach to noticeably speed up the key recovery for the full 8.5-round IDEA. In addition to these attacks, three relatively large and easily detectable classes of weak keys were found [34,35].

Implementation. Because of the complexity of the extended Euclidean algorithm, the subkeys are generated and put in a 52-entry table. To store the block to cipher and the key, we use arrays of 8-bit numbers.

KATAN and KTANTAN. KATAN and KTANTAN are two block ciphers based on a stream cipher design proposed at CHES 2009 [9]. They both take as input blocks of sizes 32, 48, or 64 bits under an 80-bit key and iterate during 254 rounds, a kind of stream ciphers composed of two linear feedback shift registers and nonlinear operations. The KATAN and KTANTAN differ from their key schedules: in KATAN, the 80-bit key is loaded into a register, which is repeatedly clocked, whereas in KTANTAN, the key is burnt (i.e., fixed) and the only possible 'flexibility' is the choice of subkey bits.
Security. Against KTANTAN, the authors of [36] propose a meet-in-the-middle attack that recovers the 80-bit secret key of the full rounds KTANTAN-{32, 48, 64} at the time complexity of $2^{72.9}$, $2^{73.8}$, and $2^{74.4}$, respectively, each requiring four chosen plaintexts. The best attack against KATAN is a conditional differential cryptanalysis described in [37] and [38]. In [37], the authors propose a conditional differential cryptanalysis with a practical complexity in the single key settings against KATAN-{32, 48, 64} respectively on 78, 70, and 68 rounds. In [38], the same authors extend their previous results in the related key settings against KATAN-{32, 48, 64} respectively on 120, 103, and 90 rounds always with a practical complexity in all cases.
Implementation. These variants make use of two round functions. At each round, the choice of using one function or the other is made using a precomputed bit *IR*. The difference between those functions is just an additional XOR in the case where this bit is '1'. Therefore, we use a constant table in which each *i*-th cell contains a value that must be used in the *i*-th round. This value is a bit field full of '1' when the XOR must be applied and '0' when not (e.g., in the case of a 32-bit security key, a 64-bit int is set to 0 x FFFFFFFFFFFFFFFF). We store the block to cipher and the key in tables of 64-bit numbers.

KLEIN. In [10], the authors propose a new lightweight block cipher called KLEIN. It ciphers blocks of size 64 bits under a key of length 64, 80, and 96 bits with a variable number of rounds equal to 12, 16, or 20. It is based on an SPN structure that mixes together elementary operations coming from the AES and from the PRESENT.
Security. In [39], the authors exploit the existence of differentials of unexpectedly high probability coming from the combination of the 4-bit S-box and the byte-oriented `MixColumns` operation to construct practical and experimentally verified chosen-plaintext key-recovery attacks on up to eight rounds of KLEIN-64. More recently, in [40], V. Lallemand and M. Naya-Plasencia managed to completely cryptanalyze KLEIN with a 64-bit key using a differential attack and to endanger 13 rounds of KLEIN-80 and 14 rounds of KLEIN-96.
Implementation. To store the block to cipher and the key, we use arrays of 8-bit numbers.

LBlock. LBlock has been proposed in [11]. It ciphers blocks of size 64 bits under keys of size 80 bits using 32 rounds of a modified Feistel network. The round function is composed of a subkey addition, eight S-boxes applied in parallel followed by a 4-bit permutation.
Security. Even if LBlock is a very recent algorithm, seven papers present cryptanalytic results against it. The best attack [28] is an impossible differential attack on 23 rounds with a complexity equal to $2^{75.36}$. The study in [41] proposes a biclique attack against a full round version of LBlock with a complexity slightly lower than the exhaustive key search and proposes a modified key schedule algorithm to prevent this attack from happening. In [42], the authors propose to apply a new cryptanalytic method called zero-correlation linear attack against LBlock, and they managed to mount a 22-round attack with a complexity slightly lower than the exhaustive key search that works for both key schedule functions of LBlock. In an improved version of this attack, [43] proposed an attack using some relations between keys (i.e. in the related key setting) on the 24 rounds of LBlock.
Implementation. To store the block to cipher and the key, we use arrays of 8-bit numbers.

LED. LED is a lightweight block cipher [12] that ciphers 64-bit blocks under keys of length 64 or 128 bits (and could be adapted for a 80-bit key). The number of rounds is 32 for a 64-bit key and 48 for a 128-bit key. Each block to cipher is represented by a 4×4 nibble matrix. LED is an SPN block cipher that uses the same inner transformations than the AES but on nibbles and optimized for hardware applications. One of the main originality of LED is the absence of a key schedule; instead, the key is XORed every four rounds. This absence is compensated by an increased number of rounds when compared with the AES.
Security. The security of LED is studied in two papers. In [44], the authors investigate the security of LED against the meet-in-the-middle attacks. They are able to mount meet-in-the-middle attacks against eight rounds of LED-64 and 16 rounds of LED-128 with complexities slightly lower than the exhaustive key search. In [45], the authors present results concerning differential cryptanalysis of LED. They first show the attacks for LED-64 reduced to 12 and 16 rounds and finally present an observation on full LED in the related key settings. In [46], the authors manage to attack 32 out of the 48 rounds of LED-128.
Implementation. We made three implementations of this block cipher. The first one (LEDxx) is a standard one and does not make use of pf tables. The others use eight lookup tables in the `SubCells`, `ShiftRows`, and `MixColumnsSerial`. In one of our implementations (LEDxx_tdur), these tables are precomputed, and we do not evaluate the cost of building them. In the others, these tables are computed before encryption and decryption. To store the block to cipher and the key, we use arrays of 8-bit numbers.

mCrypton. mCrypton [13] is the lightweight version of the block cipher Crypton [47]. mCrypton is a 64-bit block cipher with three possible key lengths 64, 96, or 128 bits. It

uses an SPN structure repeated during 12 rounds that acts on a 4×4 nibble matrix. The round function uses four elementary transformations: an S-box layer, a bit permutation, a column-to-row transposition, and a subkey addition.
Security. In [48] and [49], the security of mCrypton is scrutinized in the related key settings. In [48], the author shows that 8-round mCrypton with 128-bit key is vulnerable to a related-key rectangle attack. In [49], the authors construct 9-round related-key impossible differential attacks against mCrypton-96 and mCrypton-128. In [50], using internal collisions attack, the authors managed to attack eight out of the 12 rounds of mCrypton. Some other results [51,52] concern a biclique cryptanalysis.
Implementation. We implement the solution proposed in the reference paper, using a 8-bit array for the block to cipher and a 16-bit table for the key.

MIBS. MIBS [14] uses a Feistel structure with data block length of 64 bits and key lengths of 64 or 80 bits and is composed of 32 rounds. The internal F-function, inspired from the Camellia block cipher [53], acts on nibbles and is composed of a subkey addition, an S-box layer, a linear mixing layer, and a nibble-wise permutation.
Security. In [54], the authors present linear attacks on up to 18-round MIBS, the first ciphertext only attacks on 13-round MIBS, a differential analysis on 14-round MIBS, and an impossible differential attack on 12-round MIBS. In [55], the same authors manage to improve their attacks on the 19 rounds of MIBS-80 with a time complexity of $2^{74.23}$ encryptions. These attacks do not threaten the full 32-round MIBS but significantly reduce its security margin.
Implementation. We use a table for the S-box function and another for its inverse. We store the block to cipher and the key on tables of 8-bit numbers.

Noekeon. Noekeon [15] is a conventional block cipher with a block length and a key length of 128 bits submitted to the Nessie project. It is a substitution–linear transformation network in bit-slice mode that allows very fast and compact implementations. It is similar to Serpent and uses cyclic shifts and bit-wise Boolean operations followed by an S-box layer that acts on nibbles. It is composed of 16 rounds followed by a simple output transformation. Noekeon has two key schedules, one for applications where related-key attacks are not considered dangerous and one for applications where related-key attacks can be mounted.
Security. Noekeon has not been chosen by the Nessie project because of the analysis carried out by Knudsen and Raddum in [56]. They show that there exist many related keys for which plaintexts of certain differences result in ciphertexts of certain differences with high probabilities independent of the key schedule used. It is also shown that for six out of the seven S-boxes that satisfy the design criteria of the Noekeon designers, the resulting block ciphers are vulnerable to either a differential attack, a linear attack, or both. It is concluded that Noekeon is not designed according to an optimal diffusion strategy.
Implementation. We implement two variants of the algorithm. In the first one (*INDNoekeon* in the following), a key scheduling phase is applied. In the second one (*DIRNoekeon*), this phase is skipped, and we use the cipher itself in replacement. The block to cipher and the key are stored in tables of 32-bit entries.

Piccolo. Piccolo [16] is a 64-bit block cipher supporting 80-bit and 128-bit keys. It mixes together a four-branch Feistel structure followed by a byte permutation *RP*. The two F-functions that are called in the Feistel layer act on 16-bit words and are composed of an S-box layer applied at nibble level and of a `MixColumns`-like operation that acts at nibble level followed by a subkey addition. Piccolo is one of the first block cipher that has a hardware implementation requiring less than 1000 gates.
Security. All the results concerning the security of Piccolo focus on a biclique cryptanalysis. The best results in this direction are presented in [57] and [58] where bicliques on full round versions of Piccolo-80 and Piccolo-128 are slightly lower than the exhaustive key search and are described. In [44], the authors presented a meet-in-the-middle attack without the whitening keys on 14 rounds of Piccolo-80 and 21 rounds of Piccolo-128. In [59], two related-key attacks are presented on the same number of rounds with better complexities.
Implementation. We use one table storing the results of the multiplication of 0..15 by 2 and another for the multiplication by 3. We store the block to cipher and the key in tables of 16-bit numbers.

PRESENT. PRESENT is the most famous lightweight block cipher presented at CHES 2007 [17]. It ciphers blocks of length 64 bits under keys of lengths 80 or 128 bits. The number of rounds is equal to 31. The round function is a simple SPN network composed of a subkey addition, an S-box layer calling always the same nibble S-box, and a bit permutation layer.
Security. PRESENT has attracted a lot of cryptanalytic attention because of very particular linear biases. The papers [60–63] study the linear behavior of PRESENT regarding multiple linear trails. This kind of cryptanalysis allows to mount multi-linear attacks on up to 27 rounds of PRESENT but using all the codebook. Two bicliques with complexities about the same than the ones of the exhaustive key search against the two versions of PRESENT are also proposed in [57].
Implementation. We store the block to cipher and the key in tables of 16-bit numbers. We propose two different implementations of PRESENT. The first one (PRESENT_SIZE) optimized the size of the stored data directly implementing the bit permutation layer, whereas the second one (PRESENT_SPEED) takes advantage of a table to speed up the bit permutation layer.

PRINCE. PRINCE is a recent lightweight block cipher [18] designed for low-latency purpose. It ciphers blocks of length 64 bit under a key of 128 bits. The main part of the cipher is enclosed into an Even–Mansour construction. It

uses 12 rounds of an involutive SPN structure to limit hardware footprint. It has not been designed to resist related key attacks.
Security. Because of its originality, novel design, and low number of rounds, PRINCE has attracted the attention of a large number of cryptanalysts. The best attack published in [64] manages to attack 10 rounds out of the total number of 12 using multiple differentials and particular properties of PRINCE for the whole key recovering process. The data complexity of this attack is $2^{57.94}$, and the time complexity is $2^{60.62}$, corresponding to 118.56 security bits, instead of 126 for the generic attacks.
Implementation. We store the block to cipher and the key in tables of 64-bit numbers.

SEA. SEA-*n*, *b* [19] is a lightweight block cipher that takes an *n*-bit block under a key of length also *n*. It acts on words of size *b* bits and has *nr* rounds. It is a very suitable block cipher as *n* could take the values 48, 96, 144, and so on. It is based on a modified two-branch Feistel network. The F-function of the Feistel is constructed using elementary operations and is composed of an addition with the subkey, an S-box layer that acts on a *b*-bit word and words and bit rotations. The recommended number of rounds *nr* is equal to $3n/4 + n/b + 2 * \left\lfloor \frac{b}{2} \right\rfloor$.
Security. To our knowledge, there is no security analysis published about SEA except the ones included in the original paper [19]. The best attack found by the designers allows to cryptanalyze *nr*/2 rounds, leaving a large security margin.
Implementation. To store the block to cipher and the key, we use arrays of 16-bit numbers. We have only implemented the version of SEA with a 96-bit key and a 96-bit block to cipher.

SIMON and SPECK. SIMON and SPECK are two families of lightweight block ciphers proposed by the National Security Agency in [20]. SIMON is a hardware-oriented design, whereas SPECK is more software-oriented. The key lengths and the block lengths are really variable: each supports block sizes of 32, 48, 64, 96, and 128 bits, with up to three key sizes (taken among 64, 72, 96, 128, 144, 192, and 256 bits) to go along with each block size. The number of rounds mainly depends on the two previous parameters. For example, SIMON with blocks of length 64 bits and keys of length 96 bits has 42 rounds and SPECK with blocks of length 64 bits and keys of lengths 96 bits has 26 rounds. Both are Feistel networks based on really simple operations (three left circular shifts, one AND, and two XORs for SIMON and two left circular shifts, a modular addition, and one XOR for SPECK).
Security. A differential analysis has been published at Fast Software Encryption 2014 [65]. This paper presents differential attacks of round-reduced versions of SIMON with up to 18/32, 19/36, 25/44, 35/54, and 46/72 rounds for the 32-bit, 48-bit, 64-bit, 96-bit, and 128-bit versions, respectively. In [66], SIMON32/64, 48/72, and 48/96 are studied. The authors mount integral or zero-correlation attacks against 20 or 21 rounds of each instance with respective complexities equal to 2^{63}, 2^{60}, and 2^{73} encryptions. SPECK has clearly been less studied.
Implementation. For each possible version of SIMON and SPECK, we fit as much as possible the storing possibilities according to the block size and the key size. We present not all the possible versions of those ciphers but only the ones that seem the most adequate.

SKIPJACK. SKIPJACK [21] is a block cipher developed by the National Security Agency and declassified in 1998. SKIPJACK uses an 80-bit key and 64-bit data blocks. It is an unbalanced Feistel network with 32 rounds. It has two types of rounds, called Rule A and Rule B. Each round is described as a linear feedback shift register with an additional nonlinear keyed *G* permutation. Rule B is basically the inverse of Rule A with minor positioning differences. SKIPJACK applies eight rounds of Rule A, followed by eight rounds of Rule B, followed by another eight rounds of Rule A, followed by another eight rounds of Rule B. *G* is a four-round Feistel permutation composed of an 8-bit S-box and an 8-bit subkey addition.
Security. SKIPJACK has been subject to intensive analysis as summed up in [67]. The currently most successful attack against the cipher is the impossible differential attack, which breaks 31 out of the 32 rounds, marginally faster than the exhaustive search [68].
Implementation. We store the block to cipher and the key in tables of 8-bit numbers.

TEA and XTEA. Tiny Encryption Algorithm (TEA) is an old block cipher notable for its simplicity. It was designed in 1994 by D. Wheeler and R. Needham [23]. Because of the many weaknesses found against TEA (see, for example, [69], for more details), TEA has been replaced by XTEA in [70]. XTEA is a 64-bit block cipher with a 128-bit key. It is based on a Feistel network, and the recommended number of rounds is 64. The internal F-function is really simple and is composed of left and right shifts, XORs, and additions.
Security. Many papers have analyzed the security of XTEA. We will focus here on the most recent publications. In [29], the authors present an impossible differential attacks on 23-round XTEA. In [71], a three-subset meet-in-the-middle attack is applied against the 25 rounds of XTEA with nine known plaintexts and $2^{120.4}$ XTEA computations.
Implementation. We store the block to cipher and the key in tables of 32-bit numbers.

TWINE. TWINE is a lightweight 64-bit block cipher [22] having a 80-bit or 128-bit key. It employs a generalized Feistel structure with 16 branches. It has 36 rounds whatever the key length. The internal F-function, repeated eight times per round, is just composed of a subkey addition and of a single S-box that acts on nibbles.
Security. In [72], the authors present two biclique attacks on TWINE-80 and TWINE-128 with time complexities equal to $2^{79.10}$ and $2^{126.82}$, respectively, with a data

requirement for the two attacks equal to 2^{60}. In [73], Zero-correlation attacks against 23 and 25 rounds respectively of TWINE-80 and TWINE-128 with a complexity near the exhaustive search are presented.
Implementation. We store the subkeys and the block to cipher in tables of 8-bit numbers.

Conclusion In conclusion, we could notice that all the studied block ciphers have a sufficient security margin to be employed in real-life applications. The most risky ones seem to be KLEIN, Noekeon, and SKIPJACK due to attacks that cover all the rounds or almost all rounds. We sum up the best attacks against each block cipher in Table I.

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 8-MHz clock. This microcontroller is programmable via a Joint Test Action Group connection. It integrates a 48-KB flash memory, a 10-KB RAM memory, 48 configurable inputs/outputs, 12-bit analog-to-digital conversion pins, a watchdog, two serial communication ports, and two configurable timers. This microcontroller is compatible with most real-time operating systems such as FreeRTOS.

All the codes were written in C. We used the GNU's Not UNIX (GNU) C compiler (GCC) toolchain for MSP430 family to flash programs into the microcontroller. This includes the GCC, the assembler and linker (binutils), the GNU debugger (GDB), and some other tools needed to make a complete development environment for the MSP430. These tools can be used on Windows, Linux, Berkeley Software Distribution (BSD), and most other flavors of Unix. We used msp430-gcc version 4.6.3.

3.2. Methodology

We measure the performances of the algorithms as well as the memory consumption. We got these performances on two platforms. First, we experimented on an `msp430fg4618` microcontroller, which includes 8-KB RAM and a 256-KB flash memory and an MSP430 processor. Second, we also used the simulator coming with MSPDebug to get results on another platform[‖] used in the WSN430[**] sensors deployed in the IoT-lab platform[††] [3]. Although it is only a simulator, this simulator is cycle-accurate and able to give the number of clock cycles spent at any point of the program execution. In this paper, we only present the results obtained on the real platform because they are really similar (which was expected because both platforms have similar characteristics). The results obtained using the simulator are the ones given in [25].

Concerning the memory consumption, we distinguish between the need of read-only memory (ROM) and working memory. 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. The working memory is used to store the execution stack. It is important to distinguish between ROM and RAM uses because the physical characteristics of the underlying memory are very different. Hence, ROM can be implemented using very cheap, dense, and low-consumption memory type such as NOR flash. Dynamic RAM is most often used as the working memory, but it is much more expensive, is less dense, and especially consumes much more energy power than flash NOR.

In order to obtain the size of the ROM, we simply declare as *static const* the concerned variables and getting the size of the `.text` section in the *elf* file. To get the size of the RAM needed, we rely on mspdebug, which indicates until which address the execution stack was modified.

The entire environment used to obtain experimental results is open-source [24] and available from http://zenodo.org/record/12807#.VIbEemey7YM.

4. RESULTS

4.1. CPU cycles and energy consumption

4.1.1. Results.

Tables II and III give the performances of the algorithms for the different compilation options: –O3 and –Os. The –O3 option is the most optimizing possible option under GCC that minimizes execution time by all possible methods such as loop unrolling, function inlining, and register renaming. The –Os option considers all size optimizations to reduce code size.

4.1.2. Analysis.

As shown in Tables II and III, four different metrics are considered here: cycle count for encryption + key schedule and for decryption + key schedule, cycles/byte for encryption + key schedule and for decryption + key schedule. Of course, we are mainly interested in the cycles/byte for both encryption and decryption.

Concerning performances with the –O3 option (Table II), the best encryption/decryption processes in cycles/byte are for the AES, Noekeon, SPECK64, and SPECK128 and require less than 1000 cycles/byte, which is really competitive. Those ciphers are also competitive when the –Os option is considered, less than 2000 cycles/byte (Table III). Note also that many ciphers such as CLEFIA128, HIGHT, KLEIN, LBlock, Piccolo, and SEA use less than 2000 cycles/byte when used with the –O3 option, which is also competitive.

Surprisingly, SPECK96 has lower performances than SPECK128 even if it has less rounds. This is due to the fact

[‖] http://www.ti.com/product/msp430fg4618
[**] http://perso.ens-lyon.fr/eric.fleury/Upload/wsn430-docbook/
[††] https://www.iot-lab.info/

Table II. Software performances with the –O3 option.

Function	Block size (bits)	Encryption: cycle count	Encryption: cycles/byte	Decryption: cycle count	Decryption: cycles/byte
AES	128	10357	647	12365	772
CLEFIA128	128	23804	1487	21109	1319
CLEFIA192	128	34467	2154	25382	1586
CLEFIA256	128	36802	2300	30094	1880
DESXL	64	27334	3416	56189	7023
DIRnoekeon	128	9510	594	9717	607
HIGHT	64	8607	1075	8384	1048
IDEA	64	19489	2436	112290	14036
INDnoekeon	128	19 017	1188	19224	1201
KATAN32	32	286724	71681	285798	71449
KATAN48	48	429380	71563	434461	72410
KATAN64	64	610083	76260	598743	74842
KLEIN64	64	6524	815	10504	1313
KLEIN80	64	9188	1148	14514	1814
KLEIN96	64	11223	1402	18274	2284
KTANTAN32	32	6199955	1549988	6131917	1532979
KTANTAN48	48	6284448	1047408	6282196	1047032
KTANTAN64	64	6446104	805763	6446382	805797
LBlock	64	14097	1762	6910	863
LED128	64	165389	20673	169145	21143
LED128_tcalc	64	38462	4807	43857	5482
LED128_tdur	64	31196	3899	37584	4698
LED64	64	110334	13791	112566	14070
LED64_tcalc	64	27652	3456	31348	3918
LED64_tdur	64	21203	2650	25075	3134
MCRYPTON64	64	12455	1556	17430	2178
MCRYPTON96	64	12578	1572	17816	2227
MCRYPTON128	64	12591	1573	17912	2239
MIBS64	64	16437	2054	21001	2625
MIBS80	64	20280	2535	20490	2561
PRESENT_SIZE	64	132831	16603	128639	16079
PRESENT_SPEED	64	103950	12993	103171	12896
Piccolo128	64	14356	1794	15332	1916
Piccolo80	64	13503	1687	14299	1787
PRINCE	64	308691	38586	308761	38595
SEA	96	16853	1404	16908	1409
SKIPJACK	64	37077	4634	37056	4632
TWINE80	64	14356	1794	9131	1141
TWINE128	64	17993	2249	9131	1141
SIMON64_96	64	7937	992	7856	982
SIMON64_128	64	9627	1203	9542	1192
SIMON96_96	96	41913	3492	41916	3493
SIMON96_144	96	43611	3634	43614	3634
SIMON128_128	128	34193	2137	34196	2137
SPECK64_96	64	4229	528	3214	401
SPECK64_128	64	4391	548	3338	417
SPECK96_96	96	15 374	1281	8340	695
SPECK96_144	96	16194	1349	11113	926
SPECK128_128	128	11206	700	7500	468
XTEA	64	215339	26917	211477	26434

Table III. Software performances with the –Os option.

Function	Block size (bits)	Encryption: cycle count	Encryption: cycles/byte	Decryption: cycle count	Decryption: cycles/byte
AES	128	20 411	1275	23 907	1494
CLEFIA128	128	24 268	1516	22 710	1419
CLEFIA192	128	36 722	2295	27 524	1720
CLEFIA256	128	38 273	2392	32 392	2024
DESXL	64	25 504	3188	62 864	7858
DIRnoekeon	128	10 429	651	10 640	665
HIGHT	64	10 497	1312	10 117	1264
IDEA	64	20 672	2584	113 439	14 179
INDnoekeon	128	20 852	1303	21 063	1316
KATAN32	32	297 449	74 362	308 108	77 027
KATAN48	48	470 415	78 402	471 156	78 526
KATAN64	64	637 783	79 722	625 319	78 164
KLEIN64	64	9985	1248	14 719	1839
KLEIN80	64	13 313	1664	19 579	2447
KLEIN96	64	17 101	2137	24 943	3117
KTANTAN32	32	6 387 314	1 596 828	6 393 725	1 598 431
KTANTAN48	48	6 533 858	1 088 976	6 558 041	1 093 006
KTANTAN64	64	6 698 940	837 367	6 712 206	839 025
LBlock	64	14 097	1762	6979	872
LED128	64	596 284	74 535	597 701	74 712
LED128_tcalc	64	100 730	12 591	110 093	13 761
LED128_tdur	64	54 540	6817	63 809	7976
LED64	64	397 591	49 698	398 536	49 817
LED64_tcalc	64	82 722	10 340	88 860	11 107
LED64_tdur	64	36 532	4566	42 576	5322
MCRYPTON64	64	33 701	4212	107 671	13 458
MCRYPTON96	64	33 910	4238	107 920	13 490
MCRYPTON128	64	34 031	4253	108 053	13 506
MIBS64	64	21 642	2705	18 506	2313
MIBS80	64	20 970	2621	21 514	2689
PRESENT_SIZE	64	162 064	20 258	168 687	21 085
PRESENT_SPEED	64	126 190	15 773	125 042	15 630
Piccolo128	64	15 102	1887	16 051	2006
Piccolo80	64	14 625	1828	15 400	1925
PRINCE	64	510 473	63 809	510 599	63 824
SEA	96	25 395	2116	25 383	2115
SKIPJACK	64	37 084	4635	37 066	4633
TWINE80	64	29 719	3714	23 430	2928
TWINE128	64	33 363	4170	23 430	2928
SIMON64_96	64	8728	1091	8773	1096
SIMON64_128	64	9217	1152	9264	1158
SIMON96_96	96	42 286	3523	42 341	3528
SIMON96_144	96	43 588	3632	43 645	3637
SIMON128_128	128	34 719	2169	34 790	2174
SPECK64_96	64	4724	590	3372	421
SPECK64_128	64	4905	613	3502	437
SPECK96_96	96	18 404	1533	10 594	882
SPECK96_144	96	18 976	1581	12 101	1008
SPECK128_128	128	15 620	976	10 453	653
XTEA	64	353 694	44 211	345 314	43 164

Table IV. Memory usage with the -O3 option.

Function	Stack size (bytes)	Data Size (bytes)
AES	88	6458
CLEFIA128	390	6960
CLEFIA192	590	7642
CLEFIA256	588	7730
DESXL	162	15254
DIRnoekeon	42	1954
HIGHT	24	2216
IDEA	174	7206
INDnoekeon	42	2026
KATAN32	3822	7100
KATAN48	3898	5600
KATAN64	616	5628
KLEIN64	66	3350
KLEIN80	64	3434
KLEIN96	68	3646
KTANTAN32	1868	18104
KTANTAN48	1984	16740
KTANTAN64	2124	16688
LBlock	28	1802
LED128	144	6622
LED128_tcalc	100	7018
LED128_tdur	100	5586
LED64	144	6298
LED64_tcalc	100	6708
LED64_tdur	98	5234
MCRYPTON64	36	9218
MCRYPTON96	42	9570
MCRYPTON128	36	9642
MIBS64	52	2176
MIBS80	46	2610
PRESENT_SIZE	278	2518
PRESENT_SPEED	274	2512
Piccolo128	180	1276
Piccolo80	156	1258
PRINCE	400	20358
SEA	32	1866
SKIPJACK	42	3568
TWINE80	52	2150
TWINE128	78	2346
SIMON64_96	32	6410
SIMON64_128	72	1862
SIMON96_96	130	3452
SIMON96_144	170	4006
SIMON128_128	108	3388
SPECK64_96	20	928
SPECK64_128	20	940
SPECK96_96	62	1514
SPECK96_144	58	1664
SPECK128_128	54	1296
XTEA	30	2086

Table V. Memory usage with the -Os option.

Function	Stack size (bytes)	Data Size (bytes)
AES	36	2340
CLEFIA128	350	2756
CLEFIA192	528	2912
CLEFIA256	528	2862
DESXL	164	11594
DIRnoekeon	42	1394
HIGHT	28	1506
IDEA	156	1694
INDnoekeon	42	1450
KATAN32	3734	4098
KATAN48	3894	4914
KATAN64	612	5638
KLEIN64	70	2466
KLEIN80	68	2486
KLEIN96	74	2510
KTANTAN32	1164	7522
KTANTAN48	1286	8250
KTANTAN64	1430	8978
LBlock	28	2010
LED128	78	1486
LED128_tcalc	78	1540
LED128_tdur	78	1442
LED64	78	1502
LED64_tcalc	78	1534
LED64_tdur	78	1436
MCRYPTON64	40	1930
MCRYPTON96	48	2052
MCRYPTON128	40	2290
MIBS64	40	1948
MIBS80	40	2392
PRESENT_SIZE	274	2682
PRESENT_SPEED	274	2572
Piccolo128	170	1484
Piccolo80	146	1470
PRINCE	88	3678
SEA	28	1098
SKIPJACK	42	3560
TWINE80	52	1090
TWINE128	78	1278
SIMON64_96	26	1066
SIMON64_128	26	1090
SIMON96_96	80	2288
SIMON96_144	80	2310
SIMON128_128	74	2112
SPECK64_96	22	952
SPECK64_128	22	964
SPECK96_96	68	1466
SPECK96_144	62	1658
SPECK128_128	58	1372
XTEA	34	1700

that the operations performed by SPECK96 act on the 48-bit words stored on 64-bit words as for SPECK128. More precisely, to keep the modular addition and the subtraction bijective, we add, for SPECK96, a left shift on 16 bits to store the block to cipher on the most significant bits of the word. So, this fact adds an elementary operation performed each round in the case of SPECK96. This explains the supplementary cost paid by SPECK96. The same kind of reasoning also holds for SIMON96.

Some lightweight designs have poor performances with the –O3 option: IDEA (in the decryption direction), KATAN, KTANTAN, LED, PRESENT, and PRINCE. The bad behavior of IDEA in the decryption process comes from the subkey derivation that requires only in the deciphering direction some greatest common divisor computations that are really slow. The bad behavior of KATAN, KTANTAN, and PRESENT come from their bit-oriented nature that does not fit well with the register size of the microcontroller. LED is penalized by its huge number of rounds (32 or 48). Most of the other lightweight block ciphers behave relatively well considering this metric. The results obtained using the –Os option in terms of cycles/byte are about the same.

4.2. Memory requirements

4.2.1. Results.

Table IV and V report the memory consumption of the algorithms according to the different compilation options. It shows the requirements of ROM (code + read-only tables) as well as the amount of RAM needed to store the stack and the modifiable data. We can see that the requirement of RAM is very similar and very small, except for the CLEFIA and the KATAN and KTANTAN 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 ROM is very different from one algorithm to another. Whereas all the versions of SPECK64 require less than 1000 B of ROM to execute, KTANTAN requires more than 16 000 B for all version with the –O3 option. The ROM consumption of the KATAN and KTANTAN families is due to the tables used to store the bit fields (Section 2).

4.2.2. Analysis.

Concerning memory usage, we consider the following metrics: ROM use (in bytes) and RAM use (in bytes).

First, because of sensor memory requirement, we consider compact implementations. As shown in Table V with the –Os option, many lightweight block ciphers have memory sizes less than 1500 B: these are Noekeon, SEA, TWINE for all versions, SIMON64, and SPECK

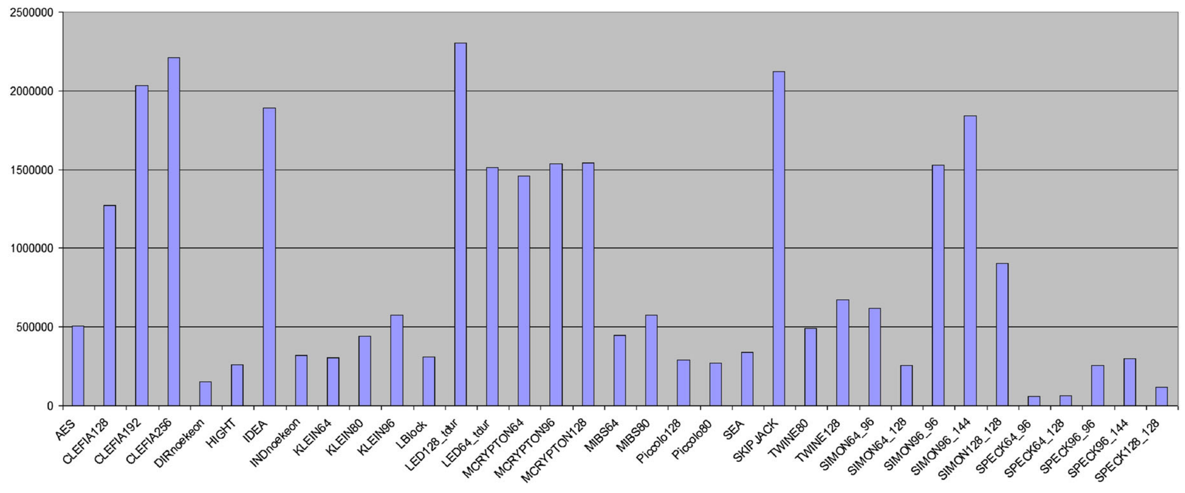

Figure 1. Metric introduced in [2] for the best algorithms and the –O3 compilation option: code size × cycle count product/block size.

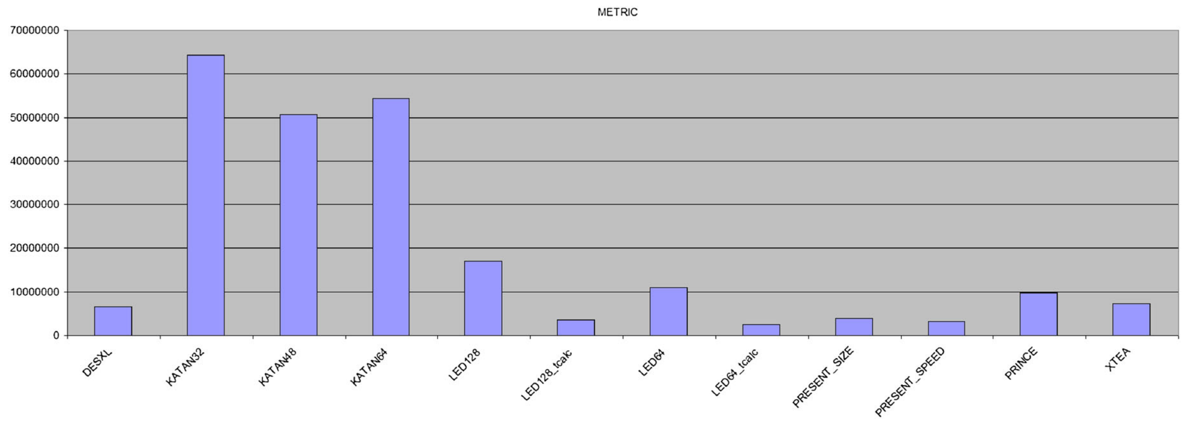

Figure 2. Metric introduced in [2] for the worth algorithms and the –O3 compilation option: code size × cycle count product/ block size.

for all versions. Most of the others have memory footprint between 2000 and 3000 B, which is really reasonable. On the contrary, all the KATAN and KTANTAN versions have huge memory footprints (more than 4000 B) due to their particular design, which has the same cost when enciphering/deciphering 32, 48, or 64 blocks in parallel.

If we want better performances using the –O3 option, then Table IV shows that Piccolo, SPECK64, and SPECK128 have memory sizes less than 1500 B. On the opposite, KTANTAN and PRINCE have really heavy memory footprints. The reasons for which PRINCE stays in this category are rather unclear.

In terms of RAM occupancy, all the block ciphers except CLEFIA, KATAN, and KTANTAN require between 30 and 300 B, which is really reasonable.

4.3. The combined metric

4.3.1. Results.

Figures 1, 2, 3, and 4 present the results obtained using the metric introduced in [2] for the encryption process (we do not provide the equivalent result for the decryption process because there are about the same few exceptions such as IDEA). This metric is equal to code size × cycle count product, normalized by the block size.

4.3.2. Analysis.

Considering the combined metric introduced in [2], we have divided, for more readability, the block ciphers into two groups: the ones with a good metric (less than 2 500 000) and the ones with a bad metric (more than 2 500 000). The results of KTANTAN are so bad that we could not include it in the figures. So, considering Figures 1 and 3, the best algorithms are Noekeon, HIGHT, KLEIN64, LBlock, Piccolo, SEA, and SPECK for all versions that behave better than the AES standard for both the –O3 and –Os options. They all have a better trade-off between memory and performances than the AES. Then, whereas HIGHT, LBlock, KLEIN64, and Piccolo cipher blocks of size 64 bits, Noekeon ciphers blocks of length 128 bits, whereas SEA and SPECK include many variants concerning block length and also key length.

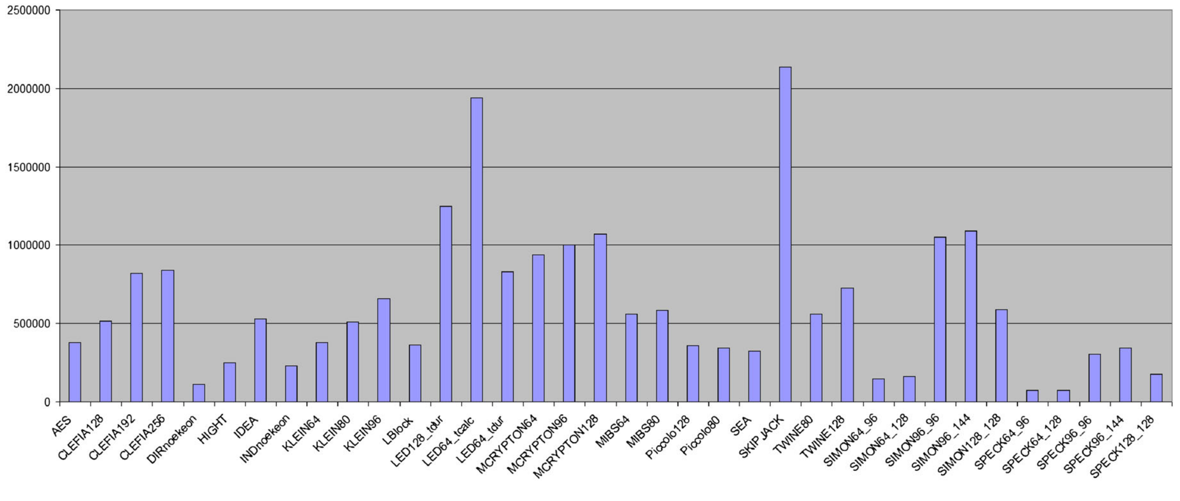

Figure 3. Metric introduced in [2] for the best algorithms and the –Os compilation option: code size × cycle count product/block size.

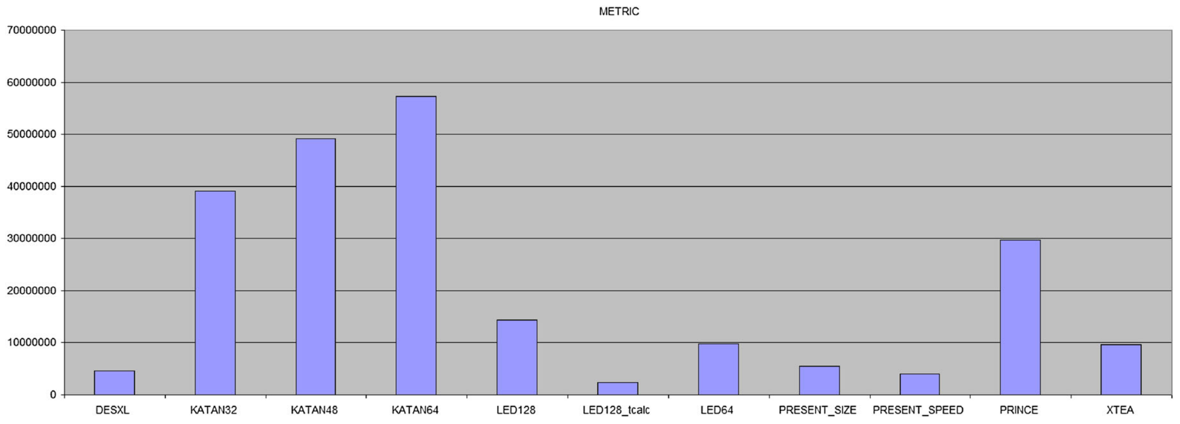

Figure 4. Metric introduced in [2] for the worth algorithms and the –Os compilation option: code size × cycle count product/block size.

So, we think that these two last ciphers are really well suited for wireless sensor networks that could require different message sizes to cipher under variable possible security bounds.

5. CONCLUSION

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 16-bit microcontroller. In total, 21 ciphers have been implemented and analyzed, keeping in mind that the compactness is an important issue in the embedded world. They show that some well-suited block ciphers such as SEA or SPECK have good performances 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 performances due to their bit-oriented nature. Finally, block ciphers with a 4-bit-oriented or 8-bit-oriented design seem to be the best compromise in terms of correct hardware implementation and good software performances.

In summary, dedicated ciphers such as SEA or SPECK with variable message lengths and variable key lengths designed using only elementary operations (such as AND and XOR) seem to be the best solutions for wireless sensor networks used. We also encourage all the authors of the studied block ciphers to send us their optimized code dedicated to the MSP430. With the consent of the authors, we will publish the codes in the dedicated GitHub repository.

In future works, we plan to estimate the complete cost of security (encryption + authentication) in wireless sensor networks. To do so, we plan to use the IoT-lab platform of sensors, and we want to estimate the latency induced by sending/receiving encrypted and authenticated messages. We also plan to implement on our sensors the finalist of the CAESAR[‡‡] competition dedicated to authenticated encryption.

ACKNOWLEDGEMENT

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

REFERENCES

1. Law YW, Doumen J, Hartel PH. Survey and benchmark of block ciphers for wireless sensor networks. *TOSN* 2006; **2**(1): 65–93.
2. Eisenbarth T, Gong Z, Güneysu T, Heyse S, Indesteege S, Kerckhof S, Koeune F, Nad T, Plos T, Regazzoni F, Standaert FX, van Oldeneel tot Oldenzeel Loïc. Compact implementation and performance evaluation of block ciphers in attiny devices. In *Progress in Cryptology - AFRICACRYPT 2012*, vol. 7374, LNCS. Springer: Berlin, 2012; 172–187.
3. des Roziers CB, Chelius G, Ducrocq T, Fleury E, Fraboulet A, Gallais A, Mitton N, Noël T, Vandaele J. Using senslab as a first class scientific tool for large scale wireless sensor network experiments. In *Networking 2011*, vol. 6640, LNCS. Springer: Berlin, 2011; 147–159.
4. FIPS 197. *Advanced Encryption Standard*. Federal Information Processing Standards Publication 197, 2001. U.S. Department of Commerce/N.I.S.T.
5. Shirai T, Shibutani K, Akishita T, Moriai S, Iwata T. The 128-bit blockcipher CLEFIA (extended abstract). In *Fast Software Encryption - FSE 2007*, vol. 4593, LNCS. Springer: Berlin, 2007; 181–195.
6. Leander G, Paar C, Poschmann A, Schramm K. New lightweight DES variants. In *Fast Software Encryption - FSE 2007*, vol. 4593, LNCS. Springer: Berlin, 2007; 196–210.
7. Hong D, Sung J, Hong S, Lim J, Lee S, Koo B, Lee C, Chang D, Lee J, Jeong K, Kim H, Kim J, Chee S. HIGHT: a new block cipher suitable for low-resource device. In *Cryptographic Hardware and Embedded Systems - CHES 2006*, LNCS 4249. Springer: Berlin, 2006; 46–59.
8. Lai X, Massey JL. A proposal for a new block encryption standard. In *Advances in Cryptology - EUROCRYPT '90*, vol. 473, LNCS. Springer: Berlin, 1990; 389–404.
9. Cannière CD, Dunkelman O, Knezevic M. KATAN and KTANTAN - a family of small and efficient hardware-oriented block ciphers. In *Cryptographic Hardware and Embedded Systems - CHES 2009*, vol. 5747, LNCS. Springer: Berlin, 2009; 272–288.
10. Gong Z, Nikova S, Law YW. KLEIN: A new family of lightweight block ciphers. In *RFID. Security and Privacy - RFIDSEC 2011*, vol. 7055, LNCS. Springer: Berlin, 2011; 1–18.
11. Wu W, Zhang L. LBlock: a lightweight block cipher. In *Applied Cryptography and Network Security - ACNS 2011*, vol. 6715, LNCS. Springer: Berlin, 2011; 327–344.
12. Guo J, Peyrin T, Poschmann A, Robshaw M. The LED block cipher. In *Cryptographic Hardware and Embedded Systems - CHES 2011*, vol. 6917, LNCS. Springer: Berlin, 2011; 326–341.
13. Lim CH, Korkishko T. mCrypton - a lightweight block cipher for security of low-cost RFID tags and sensors. In *Workshop on Information Security Applications - WISA 2005*, LNCS 3786. Springer Verlag: Berlin, 2005; 243–258.
14. Izadi M, Sadeghiyan B, Sadeghian SS, Khanooki HA. MIBS: a new lightweight block cipher, *Cryptology*

[‡‡] http://competitions.cr.yp.to/caesar.html.

and Network Security - CANS 2009, LNCS 5888, Kanazawa, Japan, 2009; 334–348.

15. Daemen J, Peeters M, Assche GV, Rijmen V. Nessie proposal: NOEKEON, 2000. Submitted as an NESSIE Candidate Algorithm. Available from: http://gro.noekeon.org/ [Accessed on May 2015].
16. Shibutani K, Isobe T, Hiwatari H, Mitsuda A, Akishita T, Shirai T. Piccolo: an ultra-lightweight blockcipher. In *Cryptographic Hardware and Embedded Systems - CHES 2011*, vol. 6917, LNCS. Springer: Berlin, 2011.
17. Bogdanov A, Knudsen LR, Leander G, Paar C, Poschmann A, Robshaw MJB, Seurin Y, Vikkelsoe C. PRESENT: an ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems - CHES 2007*, LNCS 4727. Springer: Berlin, 2007; 450–466.
18. Borghoff J, Canteaut A, Güneysu T, Kavun EB, Knezevic M, Knudsen LR, Leander G, Nikov V, Paar C, Rechberger C, Rombouts P, Thomsen SS, Yalçin T. Prince - a low-latency block cipher for pervasive computing applications - extended abstract. In *Advances in Cryptology - ASIACRYPT 2012*, vol. 7658, LNCS. Springer: Berlin, 2012; 208–225.
19. Standaert FX, Piret G, Gershenfeld N, Quisquater JJ. SEA: a scalable encryption algorithm for small embedded applications. In *Smart Card Research and Advanced Applications - CARDIS 2006*, vol. 3928, LNCS. Springer: Berlin, 2006; 222–236.
20. Beaulieu R, Shors D, Smith J, Treatman-Clark S, Weeks B, Wingers L. The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, *Report 2013/404*, 2013. Available from: http://eprint.iacr.org/ [Accessed on May 2015].
21. National Institute of Standards and Technology. SKIPJACK and KEA algorithm specification. *Technical Report*, 1998. Available from: http://csrc.nist.gov/groups/STM/cavp/documents/skipjack/skipjack.pdf [Accessed on May 2015].
22. Suzaki T, Minematsu K, Morioka S, Kobayashi E. TWINE: a lightweight block cipher for multiple platforms. In *Selected Areas in Cryptography - SAC 2012*, vol. 7707, LNCS. Springer: Berlin, 2012; 339–354.
23. Wheeler DJ, Needham RM. TEA, a tiny encryption algorithm. In *Fast Software Encryption - FSE 94*, vol. 1008, LNCS. Springer: Berlin, 1994; 363–366.
24. Cazorla M, Gourgeon S, Marquet K, Minier M. *GitHub repository*, 2014. DOI: 10.5281/zenodo.12807. Available from: https://github.com/kmarquet/bloc/tree/v0.2-alpha [Accessed on May 2015].
25. Cazorla M, Marquet K, Minier M. Survey and benchmark of lightweight block ciphers for wireless sensor networks. In *International Conference on Security and Cryptography - SECRYPT 2013*. SciTePress, Springer: Berlin, 2013; 543–548.
26. Bogdanov A, Khovratovich D, Rechberger C. Biclique cryptanalysis of the full AES. In *Advances in Cryptology - ASIACRYPT 2011*, vol. 7073, LNCS. Springer: Berlin, 2011; 344–371.
27. Derbez P, Fouque PA, Jean J. Improved key recovery attacks on reduced-round aes in the single-key setting. 2012. Available from: http://eprint.iacr.org/ [Accessed on May 2015].
28. Boura C, Naya-Plasencia M, Suder V. Scrutinizing and improving impossible differential attacks: Applications to CLEFIA, Camellia, LBlock and Simon. In *Advances in Cryptology - ASIACRYPT 2014, part I*, vol. 8873, LNCS. Springer: Berlin, 2014; 179–199.
29. Chen J, Wang M, Preneel B. Impossible differential cryptanalysis of the lightweight block ciphers TEA, XTEA and HIGHT. In *Progress in Cryptology - AFRICACRYPT 2012*, vol. 7374, LNCS. Springer: Berlin, 2012; 117–137.
30. Hong D, Koo B, Kwon D. Biclique attack on the full hight. In *Information Security and Cryptology - ICISC 2011*, vol. 7259, LNCS. Springer: Berlin, 2011; 365–374.
31. Koo B, Hong D, Kwon D. Related-key attack on the full hight. In *Information Security and Cryptology - ICISC 2010*, vol. 6829, LNCS. Springer: Berlin, 2010; 49–67.
32. Biham E, Dunkelman O, Keller N. A new attack on 6-round IDEA. In *Fast Software Encryption - FSE 2007*, vol. 4593, LNCS. Springer: Berlin, 2007; 211–224.
33. Khovratovich D, Leurent G, Rechberger C. Narrow-bicliques: cryptanalysis of full IDEA. In *Advances in Cryptology - EUROCRYPT 2012*, vol. 7237, LNCS. Springer: Berlin, 2012; 392–410.
34. Hawkes P. Differential-linear weak key classes of IDEA. In *Advances in Cryptology - EUROCRYPT '98*, vol. 1403, LNCS. Springer: Berlin, 1998; 112–126.
35. Biryukov A, Nakahara J, Preneel B, Vandewalle J. New weak-key classes of IDEA. In *Information and Communications Security - ICICS 2002*, vol. 2513, LNCS. Springer: Berlin, 2002; 315–326.
36. Wei L, Rechberger C, Guo J, Wu H, Wang H, Ling S. Improved meet-in-the-middle cryptanalysis of KTANTAN (poster). In *Information Security and Privacy - ACISP 2011*, vol. 6812, LNCS. Springer: Berlin, 2011; 433–438.
37. Knellwolf S, Meier W, Naya-Plasencia M. Conditional differential cryptanalysis of NLFSR-based cryptosystems. In *Advances in Cryptology - ASIACRYPT 2010*, vol. 6477, LNCS. Springer: Berlin, 2010; 130–145.

38. Knellwolf S, Meier W, Naya-Plasencia M. Conditional differential cryptanalysis of Trivium and KATAN. In *Selected Areas in Cryptography - SAC 2011*, vol. 7118, LNCS. Springer: Berlin, 2011; 200–212.
39. Aumasson JP, Naya-Plasencia M, Saarinen Markku-JO. Practical attack on 8 rounds of the lightweight block cipher KLEIN. In *Progress in Cryptology - INDOCRYPT 2011*, vol. 7107, LNCS. Springer: Berlin, 2011; 134–145.
40. Lallemand V, Naya-Plasencia M. Cryptanalysis of KLEIN (full version). 2014. Available from: http://eprint.iacr.org/ [Accessed on May 2015].
41. Wang Y, Wu W, Yu X, Zhang L. Security on LBlock against biclique cryptanalysis. In *Workshop on Information Security Applications - WISA 2012*, vol. 7690, LNCS. Springer: Berlin, 2012; 1–14.
42. Soleimany H, Nyberg K. Zero-correlation linear cryptanalysis of reduced-round LBlock. 2012. Available from: http://eprint.iacr.org/ [Accessed on May 2015].
43. Bogdanov A, Boura C, Rijmen V, Wang M, Wen L, Zhao J. Key difference invariant bias in block ciphers. In *Advances in Cryptology - ASIACRYPT 2013*, vol. 8269, LNCS. Springer: Berlin, 2013; 357–376.
44. Isobe T, Shibutani K. Security analysis of the lightweight block ciphers XTEA, LED and Piccolo. In *Information Security and Privacy - ACISP 2012*, vol. 7372, LNCS. Springer: Berlin, 2012; 71–86.
45. Mendel F, Rijmen V, Toz D, Varici K. Differential analysis of the LED block cipher. In *Advances in Cryptology - ASIACRYPT 2012*, vol. 7658, LNCS. Springer: Berlin, 2012; 190–207.
46. Dinur I, Dunkelman O, Keller N, Shamir A. Key recovery attacks on 3-round Even–Mansour, 8-step LED-128, and full AES2. In *Advances in Cryptology - ASIACRYPT 2013*, vol. 8269, LNCS. Springer: Berlin, 2013; 337–356.
47. Lim CH. A revised version of Crypton - Crypton v1.0. In *Fast Software Encryption - FSE'99*, vol. 1636, LNCS. Springer: Berlin, 1999; 31–45.
48. Park JH. Security analysis of mCrypton proper to low-cost ubiquitous computing devices and applications. *International Journal of Communication Systems* 2009; **22**(8): 959–969.
49. Mala H, Dakhilalian M, Shakiba M. Cryptanalysis of mCrypton - a lightweight block cipher for security of RFID tags and sensors. *International Journal of Communication Systems* 2012; **25**(4): 415–426.
50. Kang J, Jeong K, Sung J, Hong S, Lee K. Collision attacks on AES-192/256, Crypton-192/256, mCrypton-96/128, and Anubis. *Journal of Applied Mathematics* 2013; **2013**.
51. Song J, Lee K, Lee HwanJin. Biclique cryptanalysis on the full Crypton-256 and mCrypton-128. *Journal of Applied Mathematics* 2014; **2014**.
52. Shakiba M, Dakhilalian M, Mala H. Non-isomorphic biclique cryptanalysis and its application to full-round mCrypton. 2013. http://eprint.iacr.org/ [Accessed on May 2015].
53. Aoki K, Ichikawa T, Kanda M, Matsui M, Moriai S, Nakajima J, Tokita T. Camellia: a 128-bit block cipher suitable for multiple platforms - design and analysis. In *Selected Areas in Cryptography - SAC 2000*, vol. 2012, LNCS. Springer: Berlin, 2000; 39–56.
54. Bay A, Nakahara J, Vaudenay S. Cryptanalysis of reduced-round MIBS block cipher. In *Cryptology and Network Security - CANS 2010*, vol. 6467, LNCS. Springer: Berlin, 2010; 1–19.
55. Bay A, Huang J, Vaudenay S. Improved linear cryptanalysis of reduced-round MIBS. In *Advances in Information and Computer Security - IWSEC 2014*, vol. 8639, LNCS. Springer: Berlin, 2014; 204–220.
56. Knudsen LR, Raddum H. On Noekeon, 2001. Available from: https://www.cosic.esat.kuleuven.be/nessie/reports/phase1/uibwp3-009.pdf [Accessed on May 2015].
57. Jeong K, Kang H, Lee C, Sung J, Hong S. Biclique cryptanalysis of lightweight block ciphers present, Piccolo and LED, 2012. Available from: http://eprint.iacr.org/ [Accessed on May 2015].
58. Song J, Lee K, Lee H. Biclique cryptanalysis on lightweight block cipher: HIGHT and Piccolo. *International Journal of Computer Mathematics* 2013; **90**(12): 2564–2580.
59. Minier M. On the security of Piccolo lightweight block cipher against related-key impossible differentials. In *Progress in Cryptology - INDOCRYPT 2013*, vol. 8250, LNCS. Springer: Berlin, 2013; 308–318.
60. Nakahara J, Sepehrdad P, Zhang B, Wang M. Linear (hull) and algebraic cryptanalysis of the block cipher PRESENT. In *Cryptology and Network Security - CANS 2009*, LNCS 5888. Springer: Berlin, 2009; 58–75.
61. Collard B, Standaert FX. A statistical saturation attack against the block cipher PRESENT. In *Topics in Cryptology - CT-RSA 2009*, LNCS 5473. Springer: Berlin, 2009; 195–210.
62. Collard B, Standaert FX. Multi-trail statistical saturation attacks. In *Applied Cryptography and Network Security - ACNS 2010*, vol. 6123, LNCS. Springer: Berlin, 2010; 123–138.
63. Leander G. On linear hulls, statistical saturation attacks, PRESENT and a cryptanalysis of PUFFIN. In *Advances in Cryptology - EUROCRYPT 2011*, vol. 6632, LNCS. Springer: Berlin, 2011; 303–322.

64. Canteaut A, Fuhr T, Gilbert H, Naya-Plasencia M, Reinhard Jean-René. Multiple differential cryptanalysis of round-reduced PRINCE (full version). 2014. Available from: http://eprint.iacr.org/ [Accessed on May 2015].
65. Abed F, List E, Lucks S, Wenzel J. Differential and linear cryptanalysis of reduced-round SIMON. 2013. Available from: http://eprint.iacr.org/ [Accessed on May 2015].
66. Wang Q, Liu Z, Varici K, Sasaki Y, Rijmen V, Todo Y. Cryptanalysis of reduced-round SIMON32 and SIMON48. In *Progress in Cryptology - INDOCRYPT 2014*, vol. 8885, LNCS. Springer: Berlin, 2014; 143–160.
67. Kim J, Phan RCW. A cryptanalytic view of the NSA's Skipjack block cipher design. In *Advances in Information Security and Assurance - ISA 2009*, vol. 5576, LNCS. Springer: Berlin, 2009; 368–381.
68. Biham E, Biryukov A, Shamir A. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In *Advances in Cryptology - EUROCRYPT '99*, vol. 1592, LNCS. Springer: Berlin, 1999.
69. Kelsey J, Schneier B, Wagner D. Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA. In *Information and Communication Security - ICICS'97*, vol. 1334, LNCS. Springer: Berlin, 1997; 233–246.
70. Needham RM, Wheeler DJ. Tea extensions. *Technical report*, Computer Laboratory, University of Cambridge, Cambridge, UK, October 1997.
71. Sasaki Y, Wang L, Sakai Y, Sakiyama K, Ohta K. Three-subset meet-in-the-middle attack on reduced XTEA. In *Progress in Cryptology - AFRICACRYPT 2012*, vol. 7374, LNCS. Springer: Berlin, 2012; 138–154.
72. Çoban M, Karakoç F, Özkan B. Biclique cryptanalysis of TWINE. 2012. Available from: http://eprint.iacr.org/ [Accessed on May 2015].
73. Wang Y, Wu W. Improved multidimensional zero-correlation linear cryptanalysis and applications to LBlock and TWINE. In *Information Security and Privacy - ACISP 2014*, vol. 8544, LNCS. Springer: Berlin, 2014; 1–16.