Survey and Benchmark of Block Ciphers for Wireless Sensor Networks

YEE WEI LAW, JEROEN DOUMEN, and PIETER HARTEL
University of Twente

Cryptographic algorithms play an important role in the security architecture of wireless sensor networks (WSNs). Choosing the most storage- and energy-efficient block cipher is essential, due to the facts that these networks are meant to operate without human intervention for a long period of time with little energy supply, and that available storage is scarce on these sensor nodes. However, to our knowledge, no systematic work has been done in this area so far. We construct an evaluation framework in which we first identify the candidates of block ciphers suitable for WSNs, based on existing literature and authoritative recommendations. For evaluating and assessing these candidates, we not only consider the security properties but also the storage- and energy-efficiency of the candidates. Finally, based on the evaluation results, we select the most suitable ciphers for WSNs, namely Skipjack, MISTY1, and Rijndael, depending on the combination of available memory and required security (energy efficiency being implicit). In terms of operation mode, we recommend Output Feedback Mode for pairwise links but Cipher Block Chaining for group communications.

Categories and Subject Descriptors: C.3 [**Special-Purpose and Application-Based Systems**]—*Real-time and embedded systems*; C.4 [**Performance of Systems**]—*Measurement techniques*; E.3 [**Data Encryption**]—*Standards*

General Terms: Measurement, Performance, Security

Additional Key Words and Phrases: Sensor networks, cryptography, block ciphers, energy efficiency

1. INTRODUCTION

A wireless sensor network (WSN) is a network composed of a large number of sensors that (1) are physically small, (2) communicate wirelessly among each other, and (3) are deployed without prior knowledge of the network topology. Due to the limitation of their physical size, the sensors tend to have storage space, energy supply, and communication bandwidth so limited that every possible means of reducing the usage of these resources is aggressively sought. For example, a sensor typically has 8∼120 KB of code memory and 512∼4096 bytes of data memory. The energy supply of a sensor is such that it will be depleted in

This work is partially supported by the EU under the IST-2001-34734 EYES project.
Authors' address: Department of Computer Science, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands; email: {yee.wei.law, jeroen.doumen, pieter.hartel}@utwente.nl.

Table I. Comparison of the EYES Node with Smart Dust and the Intel Research Mote

	Smart Dust	EYES Node	Intel Mote
CPU	8-bit, 4 MHz	16-bit, 8 MHz	16-bit, 12 MHz
Flash memory	8 KB	60 KB	512 KB
RAM	512 B	2 KB	64 KB
Frequency	916 MHz	868.35 MHz	900 MHz
Bandwidth	10 kbps	115.2 kbps	100 kbps

less than 3 days if operated constantly in active mode [Zhang et al. 2004]. The transmission bandwidth ranges from 10 kbps to 115 kbps. Table I compares the sensor node used in the EYES project [van Hoesel et al. 2003] with Smart Dust [Hill et al. 2000] and the Intel Research mote [Kling 2003].

Karlof and Wagner [2003] made an interesting observation that WSNs will more likely ride Moore's Law *downward*, that is, instead of relying on the computing power to double every 18 months, we are bound to seek ever cheaper solutions. However looking at the current development of WSNs (Table I), computing power is indeed increasing, though not necessarily at the rate predicted by Moore's Law. Either way, we are conservative and assume that the hardware constraints of WSNs will remain constant for some time to come.

Cryptographic algorithms are an essential part of the security architecture of WSNs. Using the most efficient and sufficiently secure algorithm is thus an effective means of conserving resources. By 'efficient' in this article we mean requiring little storage and consuming little energy. Although transmission consumes more energy than computation, our focus in this article is on computation and we can only take transmission energy into account when considering the security scheme as a whole. The essential cryptographic primitives for WSNs are block ciphers, message authentication codes (MACs) and hash functions. Among these primitives, we are only concerned with block ciphers, because MACs can be constructed from block ciphers [Preneel 1998], and hash functions are relatively cheap. Meanwhile, public-key algorithms are well-known to be prohibitively expensive [Carman et al. 2000].

Our selection of block ciphers is Skipjack [NIST 1998], RC5 [Rivest 1995], RC6 [Rivest et al. 1998], Rijndael [Daemen and Rijmen 1999], Twofish [Schneier et al. 1998], MISTY1 [Matsui 1997], KASUMI [3GPP 1999] and Camellia [Aoki et al. 2001b]. Although Rijndael has been selected by the American National Institute of Standards and Technology (NIST) as the Advanced Encryption Standard (AES) after a five-year long standardization process that included extensive benchmarking on a variety of platforms ranging from smart cards [Hachez et al. 1999] to high end parallel machines [Worley et al. 2001], the selection of Rijndael for our platform is *not* obvious. This is because the fact that Rijndael is *on average* the best performer on a range of standard platforms, does not mean that it also performs best on our platform, which is Texas Instruments' 16-bit RISC-based MSP430F149 [Texas Instruments, Inc. 2001], chosen for its ultra-low power consumption. This microcontroller has 60 KB of Flash memory, 2 KB of RAM, 16 registers, an instruction set of 51 instructions and 5 power-saving modes, and is typically used in sensor systems. The fact that it is not commonly used in smart cards or any mainstream 32/64-bit computing

platform, which are the platforms the AES committee mainly focused on, further suggests the necessity of our study. Although with the advent of IEEE 802.15.4 or ZigBee [IEEE 2003], hardware implementations of AES are expected to appear, there would still be customized sensor nodes that are not equipped with such hardware for cost and other reasons (e.g., our nodes), and software implementations offer more flexibility (e.g., in software updates), so our benchmark results are therefore still applicable.

The contribution of this article is three-fold: (1) to identify the candidates of block ciphers suitable for WSNs based on existing literature; (2) to provide a systematic methodology for assessing the suitability of the ciphers, in terms of both security and efficiency; and (3) to select the suitable ciphers for WSNs based on the evaluation results.

Our evaluation framework consists of (1) literature survey, and (2) benchmarking. The rest of this article is organized as follows. Section 2 contains the literature survey of the block ciphers, explaining why they are selected and why others are not, as well as shedding light on their security properties. Section 3 details aspects of our benchmarking. Section 4 provides our observation and evaluation results. Section 5 concludes.

2. LITERATURE SURVEY

A typical cipher consists of three components: (1) the encryption algorithm, (2) the decryption algorithm and (3) the key expansion algorithm (also known as key scheduling or key setup). The key expansion algorithm expands the *user key* or *cipher key* to a larger intermediate key, to allow (ideally) all bits of the cipher key to influence every round of the encryption algorithm. For most ciphers, key expansion only needs to be done once to cater for both encryption and decryption; for other ciphers however, key expansion has to be done separately for encryption and decryption (e.g. for Rijndael). The most important parameters of a block cipher are (1) the *key length*(s) it supports, (2) the *block size* and (3) the *nominal number of rounds*. In the ensuing discussion, for each cipher we give the reasons why we have chosen to evaluate the cipher, as well as providing status quo information on its security strength. Table II explains some of the terms we use in the following discussion. Note that in Table II we adopt Lenstra and Verheul's [2001] definition of *security margin*. The definition can be understood as follows. Suppose (1) c_{DES} is the number of computations required to break DES, (2) c_X is the number of computations required to break cipher X, and (3) an attacker that can afford c_{DES} computations starting from 1982, can afford c_X computations starting from year y, then the security of cipher X in year y is computationally equivalent to the security of DES in 1982, or in other words, the security margin of cipher X is y. The year 1982 is chosen as the baseline because DES was standardized in 1977 and set for review in 1982. If the best known attack against a cipher with key length k is exhaustive key search, y can be calculated according to Equation 1 [Lenstra and Verheul 2001]:

$$y = 1982 + \frac{30}{23}(k - 56) \tag{1}$$

Table II. Terminology in Cryptanalysis

Term	Definition
Encryption/decryption oracle	An abstraction that is independent of the adversary, but encrypts/decrypts plaintexts/ciphertexts for the adversary on request.
Known-plaintext attack	An attack whereby the adversary, who has no access to the encryption or decryption oracle, uses a number of known plaintext-ciphertext pairs in his analysis to recover the key.
Chosen-plaintext attack	An attack whereby the adversary, who has no access to decryption oracle, feeds a number of plaintexts of his choice to the encryption oracle, in order to get the corresponding ciphertexts that would help in his analysis to recover the key.
Chosen-ciphertext attack	Similar to chosen-plaintext attack, except that the adversary (1) has access to the decryption oracle instead of the encryption oracle, (2) the adversary feeds the oracle with ciphertexts of his choice, and (3) the adversary uses a potentially different kind of analysis.
Linear cryptanalysis	A known-plaintext attack whereby the adversary studies the linear approximation to: Plaintext$[i_1, i_2, \ldots, i_a] \oplus$ Ciphertext$[j_1, j_2, \ldots, j_b] =$ Key$[k_1, k_2, \ldots, k_c]$, where $i_1, i_2, \ldots, i_a, j_1, j_2, \ldots, j_b, k_1, k_2, \ldots, k_c$ denote fixed bit locations [Matsui 1993].
Differential cryptanalysis	A chosen-plaintext attack whereby the adversary encrypts a pool of plaintext pairs with chosen differences and filters those pairs that have the expected ciphertext differences—these pairs reveal internal behaviour of the cipher that helps the adversary determine bits of the secret key [Biryukov 1999].
Security margin	The year until which breaking the cipher requires more effort than breaking DES in 1982 [Lenstra and Verheul 2001].

Hence in the discussion below, when a cipher with 80-bit keys has a security margin of 2013, or when a cipher with 128-bit keys has a security margin of 2076, we consider the cipher to be "secure."

2.1 Skipjack

We have chosen to evaluate Skipjack because it is used in TinySec [Karlof et al. 2004], SenSec [Li et al. 2005], and TinyKeyMan [Liu et al. 2005]. As TinySec is an optional part of TinyOS, the de facto operating system for WSNs, to the user community of TinyOS, justifying the use of Skipjack seems like a favor waiting to be fulfilled.

2.1.1 *Security.* Skipjack is a 64-bit block cipher with an 80-bit key. The fact that it is declassified by the NSA raises natural suspicions over its security. However since its declassification in 1998, Skipjack has resisted years of cryptanalysis. The best attack on Skipjack with full 32 rounds is still

exhaustive key search. With reduced rounds, the best known attack is Biham et al.'s [1999] cryptanalysis with impossible differentials that breaks 31-round Skipjack slightly faster than exhaustive search using 2^{34} chosen plaintexts, 2^{64} memory, and 2^{78} encryptions. Reichardt and Wagner [2002] observe that as the nonlinear permutation of Skipjack affects only a quarter of the bits at each round, it is relatively easy to follow differentials across multiple rounds. However, they also show that there are no meaningful truncated differentials for Skipjack with full 32 rounds, providing heuristic evidence that Skipjack *may be* secure against truncated differential distinguishing attacks.

Differential cryptanalysis aside, it is easy to see exhaustive key search as a promising attack against Skipjack due to its relatively short key length of 80 bits, but there is a computationally cheap way to increase the effective key length of any block cipher. This mechanism, originally conceived for DES by Rivest in 1984, is known as the DESX contruction [Kilian and Rogaway 1996]. In fact this is the mechanism Li et al. [2005] use to extend Skipjack to what they call Skipjack-X in their SenSec framework. The mechanism works as follows. Denote P as the plaintext, K as an 80-bit key, K_1 and K_2 as two 64-bit keys, then Skipjack-$\mathrm{X}_{K,K1,K2}(\mathrm{P}) = K_2 \oplus$ Skipjack$_K(K_1 \oplus P)$. The new key, of Skipjack-X, is therefore $80 + 64 + 64 = 208$ bits long. Applying Kilian and Rogaway's [1996] analysis, we can show that an adversary's advantage at distinguishing between a Skipjack-X and a random permutation is bounded by $t/2^{(143-\log_2 m)}$, where t is the total number of trial encryptions/decryptions the adversary can perform, and m is the number of plaintext-ciphertext pairs the adversary can obtain. This means the effective length of Skipjack-X is $143 - \log_2 m$. However, if we apply Biryukov and Wagner's [2000] *sliding with a twist* cryptanalysis on Skipjack-X, we will get an attack that requires $2^{(64+1)/2} = 2^{32.5}$ known plaintexts and $2^{80-1+32.5} = 2^{111.5}$ offline trial Skipjack encryptions. This means the 208-bit keys that are used with Skipjack-X are only about as strong as 111-bit keys—such a strategy does not seem to constitute a sound investment of memory. Therefore we do not think of Skipjack-X as a worthy replacement for Skipjack in WSNs.

To conclude, we note that while mainstream cryptography is moving away from keys smaller than 128 bits, Skipjack with full 32 rounds is secure as of today, with a security margin of 2013.

2.2 RC5

We have chosen to evaluate RC5 [Rivest 1995] because RC5 is a well-known algorithm that has been around since 1995 without crippling weaknesses. Although `distributed.net` has managed to crack a 64-bit RC5 key in RSA Laboratories' Secret-Key Challenge after 1757 days of computing involving 58,747,597,657 distributed work units, the standard key length of RC5 is 128 bits and RC5 has managed to withstand years of cryptanalysis.

In terms of design, RC5 is an innovative cipher that uses *data-dependent* rotations and is fully parameterised in word size, number of rounds, and key length. Such flexibility is rare among mainstream ciphers. RC5 is also designed to be suitable for hardware as well as software implementations. Last, Perrig

et al. [2001], Xue and Ganz [2003] and Liu et al. [2005] choose to use RC5 for WSNs. We would like to find out if their choice is justified.

2.2.1 *Security.* RC5 is conventionally represented as RC5-w/r/b, where w, the word size, is the number of bits in a word of the target computing platform, r is the number of rounds, and b, the key length, is the number of bytes of a key. The attacks found up to year 1998 have been summarized in Kaliski and Yin's [1998] technical report. The best attack cited is by Biryukov and Kushilevitz [1998], which breaks RC5-32/12/16 with just 2^{44} chosen plaintexts (compared with 2^{43} known plaintexts for DES), and RC5-32/16/16 with 2^{61} chosen plaintexts. Biryukov and Kushilevitz therefore recommend increasing the number of rounds to at least 18.

After Biryukov and Kushilevitz [1998], Borst et al. [1999] manage to reduce the storage requirements drastically for attacking RC5 (with 64-bit block size) and RC6 (with 128-bit block size) using the linear hull effect [Nyberg 1995]. This is known as the first *experimentally executed* known plaintext attacks on reduced versions of RC5 (with up to 5 rounds).

Shimoyama et al. [2000] introduce another route of attack based on statistical correlations derived from χ^2 tests. They have managed to derive the last round key of up to 17 rounds by using a chosen plaintext attack. While Shimoyama et al. use a chosen plaintext attack, Miyaji et al. [2002] use a known plaintext attack, which breaks RC5-32/10/16 with $2^{63.67}$ plaintexts at a probability of 90%.

The attacks described so far are theoretical. Ironically, intended as a security feature, data-dependent rotation invites implementation-based attacks. One such attack (on RC5-32/12/16) has been proposed by Handschuh and Heys [1998] which only needs about 2^{20} encryption timings and a time complexity between 2^{28} and 2^{40}. Kelsey et al. [1998] describe another implementation-based attack by observing the processor flags.

To conclude, in view of Biryukov and Kushilevitz's [1998] recommendation and Shimoyama et al.'s [2000] discovery, RC5-32/18/16 (18 rounds) should be secure.

2.3 RC6

We have chosen to evaluate RC6 [Rivest et al. 1998] because like RC5, RC6 is parameterized and has a small code size. RC6 is one of the five finalists that competed in the AES challenge and according to some AES evaluation reports [Nechvatal et al. 2000; Worley et al. 2001], it has reasonable performance. Last, RC6 has been chosen as the algorithm of choice by Slijepcevic et al. [2002] for WSNs. We are interested in seeing if their choice is justified.

2.3.1 *Security.* The technical report of the Cryptography Research and Evaluation Committee (CRYPTREC), Information-Technology Promotion Agency (IPA) of Japan [CRYPTREC 2001] has a summary of the attacks known up to year 2001. The first notable attack is by Knudsen and Meier [2000]. Their attack is based on statistical correlations derived from χ^2 tests. By observing the nonuniformness of the five least significant bits from each of the two Feistel halves in RC6, they require approximately $2^{13.8+16.2r}$ plaintexts to distinguish

$3 + 2r$-round RC6 from a pseudorandom permutation. For RC6 with 17 rounds, they estimate that at most, about 2^{118} plaintexts are required.

At the same time, Gilbert et al. [2000] present a theoretical attack of 14-round RC6 based on an iterative statistical weakness found in RC6's round function. The attack requires 2^{118} known plaintexts, a memory size of 2^{112} and 2^{122} operations.

Takenaka et al. [2002, 2003] propose the Transition Matrix Computation technique for evaluating security against χ^2 attacks, by which they are able to deduce the "weakest key" of RC6 against χ^2 attacks.

Shimoyama et al. [2002] show that the 64-bit target extended key of 18-round RC6 with a weak key (which exists once in every 2^{90} keys), can be recovered with a probability of 95% by *multiple linear cryptanalysis* with $2^{127.423}$ known plaintexts, a memory of size 2^{64}, and $2^{193.423}$ computations of the round-function.

To conclude, RC6-32/20/16 (20 rounds) is secure.

2.4 Rijndael

We have chosen to evaluate Rijndael [Daemen and Rijmen 1999] because Rijndael is the Advanced Encryption Standard, mandated by the NIST of the United States, chosen after extensive scrutiny and performance evaluation (`http://csrc.nist.gov/encryption/aes`). It is also one of the ciphers recommended by the New European Schemes for Signature, Integrity and Encryption (NESSIE) Consortium (`www.cryptonessie.org`), and Japan's CRYPTREC [2003]. Rijndael, as the AES, is well studied and there are efficient implementations on a wide range of platforms (`www.rijndael.com`). We would however like to obtain first-hand experience of evaluating Rijndael on our particular platform, which has never been studied before during the evaluation process of the AES.

2.4.1 *Security.* Like most modern block ciphers, Rijndael is designed with resistance against differential and linear cryptanalysis in mind, using the latest results in cryptographic research [Daemen and Rijmen 1999]. For example, Cheon et al.'s [2002] impossible differential cryptanalysis requires $2^{91.5}$ chosen ciphertexts and 2^{122} encryptions to attack 6-round Rijndael-128 (128-bit keys). Gilbert et al. [2000] describe (1) an attack on 7-round Rijndael-196 and Rijndael-256 with 2^{32} chosen plaintexts and a complexity of about 2^{140}, and (2) an attack on 7-round Rijndael-128 with 2^{32} chosen plaintexts and a computational complexity slightly less than exhaustive search. Ferguson et al. [2001a] report a related-key attack of 9-round Rijndael-256 with time complexity 2^{224}, which is of course far from practical. No attack is known for Rijndael with more than 9 rounds.

On the other hand, although Rijndael has been chosen as the AES, the security of Rijndael has gone through twists and turns of controversy. The algebraic nature of Rijndael [Ferguson et al. 2001b], has interestingly opened up possible avenues of other nontraditional attacks as summarized in the Crypto-Gram Newsletter of September 2002 [Schneier 2002a]. It started with Courtois and Pieprzyk [2002a, 2002b] presenting evidence that the security of Rijndael might

not grow exponentially as intended with the number of rounds. Their technique is based on expressing the S-boxes of Rijndael in an overdefined system of multivariate quadratic equations that can be solved by an algorithm called XSL. XSL is in turn based on XL [Courtois et al. 2002]. The security of Rijndael therefore lies with the computational complexity of XL, which to date remains an open problem [Courtois and Patarin 2003]. In spite of Moh's [2002] dispute, whether the technique would *not* work, remains to be proven [Schneier 2002b]. In the meantime, Murphy and Robshaw [2002b] derive an alternative representation of Rijndael that is easier to cryptanalyse, by embedding Rijndael in a cipher called BES which uses only simple algebraic operations in $GF(2^8)$. They show that Rijndael encryption can then be described by an extremely sparse overdetermined multivariate quadratic system over $GF(2^8)$, whose solution would recover the key. In another paper, Murphy and Robshaw [2002a] argue that while XSL does not have estimates accurate enough to substantiate claims of the existence of a key recovery attack, XSL does help solve their $GF(2^8)$ system of equations more efficiently than Courtois et al.'s $GF(2)$ system of equations. Combining Coppersmith's [2002] correction of Courtois et al.'s estimates, Murphy et al. further deduce that the security of Rijndael-128 would be reduced from the theoretical complexity of exhaustive key search, 2^{128} to 2^{100}, *if* XSL is a valid technique.

On the other front, Fuller and Millan [2002] unravel serious linear redundancy in the only nonlinear component—the S-box, of Rijndael: the 8×8 S-box behaves actually like a 8×1 S-box. They, by studying the invariance properties of the local connection structure of affine equivalence classes, discover that the outputs of the S-box are all equivalent under affine transformation. The essence of their discovery can be summarized in the following simple mathematical expression: Let $b_i(x)$ and $b_j(x)$ be two distinct outputs of the S-box, then there exists a nonsingular 8×8 matrix D_{ij} and a binary constant c_{ij} such that $b_j(x) = b_i(x)D_{ij} \oplus c_{ij}$.

Independently of the above development, Filiol shocked the scientific community in January 2003 by announcing a break of Rijndael with his plaintext-dependent repetition codes cryptanalysis technique [Filiol 2003]. By detecting bias in the Boolean functions of Rijndael, Filio claimed that he was able to obtain 2 bits of a Rijndael key with only 2^{31} ciphertexts and a computational complexity of mere $\mathcal{O}(2^{31})$. Fortunately, several independent cryptographers were quick to dismiss the claim [Courtois et al. 2003].

To conclude, the research on Rijndael has entered an interesting era. More and more previously unknown properties are now being discovered and analyzed [Barkan and Biham 2002; Tri Van Le 2003; Youssef and Tavares 2002]. Despite the above debate, we are adopting the recommendation of NESSIE [Preneel et al. 2003] and CRYPTREC [2003] that Rijndael is secure.

2.5 Twofish

We have chosen to evaluate Twofish [Schneier et al. 1998] because it allows a wide range of tradeoffs between size and speed. It is also designed to be efficient on a wide range of platforms. Since Twofish is more efficient than RC6 on some

embedded processor platforms, for example, on 8-bit Z80 [Sano et al. 2001] and on 8-bit 6805 [Keating 1999], and since we are investigating RC6, Twofish is also included in our investigation.

2.5.1 *Security.* One of the earliest cryptanalytic attempts [Mirza and Murphy 1999] from outside the Twofish design team found flaws in the original security assessment of the key schedule, but this resulted in no practical implication. One of the unprecedented features of Twofish is its key-dependent S-boxes (conventional S-boxes are fixed), but Murphy [2000] and Murphy and Robshaw [2002c], using primarily (1) the fact that the key-dependent S-boxes are determined only by half of the key, and (2) differential cryptanalysis, show that key-dependent S-boxes, as used in Twofish, provide no additional security over well-designed fixed S-boxes, and may in fact improve the range of options available to attackers. Nevertheless, it is still an open question whether any practical attack will result from Murphy et al.'s analysis [Schneier et al. 1999a; Kelsey 2000]. Biham and Furman's [2000] impossible differential cryptanalysis needs 1.82×2^{128} one-round computations and 2^{64} chosen plaintexts to break 6-round Twofish with 128-bit keys. The best known attack is Lucks' [2002] saturation attack, which requires 2^{127} chosen plaintexts and 2^{126} encryptions (two times faster than exhaustive search) to break 7-round Twofish with 128-bit keys.

To conclude, 16-round Twofish is secure.

2.6 MISTY1

We have chosen to evaluate MISTY1 [Matsui 1997] because MISTY1 is one of the CRYPTREC-recommended [2003] 64-bit ciphers and is the predecessor of KASUMI, the 3GPP-endorsed encryption algorithm [3GPP 1999]. MISTY1 is specifically designed to resist differential and linear cryptanalysis. MISTY1 is designed for high-speed implementations on hardware as well as software platforms by using only logical operations and table lookups. We find MISTY1 to be particularly suitable for 16-bit platforms. MISTY1 is a royalty-free open standard documented in RFC2994 [Ohta and Matsui 2000].

2.6.1 *Security.* Babbage and Frisch [2001] demonstrate the possibility of a 7th order differential cryptanalytic attack on 5-round MISTY1. According to them, none of the S-boxes with optimal linear and differential properties has an optimal behaviour with respect to higher order differential cryptanalysis, however as improvement, the number of rounds of the *FI* function could be increased. As we will see later, KASUMI, derived from MISTY1, incorporated such improvement, in that it has four instead of three S-boxes in its *FI* function.

Kühn [2001] found an impossible differential attack on 4-round MISTY1 using 2^{38} chosen plaintexts and 2^{62} encryptions. In the same paper, a collision-search attack has also been described—the attack requires 2^{28} chosen plaintexts and 2^{76} encryptions. In a later paper, Kühn [2002] shows that the *FL* function introduces a subtle weakness in 4-round MISTY1. This weakness allows him to launch a slicing attack with as few as $2^{22.25}$ chosen plaintexts, with

a memory requirement of $2^{34.2}$ bytes and time complexity of 2^{45}, on 4-round MISTY1.

The best known attack on 5-round MISTY1 so far is Knudsen and Wagner's [2002] integral cryptanalysis, at a cost of 2^{34} chosen plaintexts and 2^{48} time complexity.

To conclude, MISTY1 with full 8 rounds is secure.

2.7 KASUMI

We have chosen to evaluate KASUMI [3GPP 1999] because KASUMI, as the 3GPP-endorsed encryption algorithm [3GPP 1999], is presumably well-suited for embedded applications, and has gone through considerable expert scrutiny.

2.7.1 *Security*. KASUMI more or less inherits the security and performance benefits of MISTY1.

At the same time, reporting attacks on MISTY1 and MISTY2, Kühn [2001] is also able to extend the attacks to KASUMI. His impossible differential attack on 6-round KASUMI requires 2^{55} chosen plaintexts and 2^{100} encryptions.

Kang et al. [2001a, 2001b] prove that 3-round KASUMI is not a pseudorandom permutation ensemble but 4-round KASUMI is a pseudorandom permutation ensemble. However Tanaka et al. [2001] show that 4-round KASUMI without the *FL* functions can be attacked using effective 2nd order differentials. The attack can be used to find the 6 sub-keys at the 4th round, at a cost of 1416 chosen plaintexts and $2^{22.11}$ times *FO* function operations.

To conclude, KASUMI with full 8 rounds is secure.

2.8 Camellia

We have chosen to evaluate Camellia [Aoki et al. 2001a] because Camellia is one of the NESSIE- and CRYPTREC-recommended [2003] 128-bit ciphers. Camellia is designed for high-speed implementations on hardware as well as software platforms by using only logical operations and table lookups. Aoki et al. [2001b] claim their hardware implementation of Camellia occupies only 7.875K gates using a 0.11 μm CMOS ASIC library and is in the smallest class among all existing 128-bit block ciphers. Camellia is designed not only to be resistant to differential cryptanalysis, linear cryptanalysis, higher order differential attacks, interpolation attacks, related-key attacks, truncated differential attacks, boomerang attacks, and slide attacks, but also with a large safety margin in view of anticipated progress in cryptanalysis techniques [Aoki et al. 2001a; Matsui and Tokita 2000]. Futhermore, Camellia is royalty-free.

2.8.1 *Security*. He and Qing [2001] discover that the Square attack [Daemen et al. 1997] is not only applicable to the Square block cipher but also to ciphers with a Feistel structure. The complexity of their attack on 6-round Camellia is 3328 chosen plaintexts and 2^{112} encryptions. Sugita et al. [2001] found a nontrivial 7-round impossible differential. Lee et al.'s [2002] truncated differential cryptanalysis of 7-round Camellia without the FL/FL^{-1} functions, requires only 192 encryptions, but $2^{82.6}$ chosen plaintexts, to recover an 8-bit key. Yeom et al. [2002] propose a Square attack on 9-round Camellia with

256-bit keys, at a cost of $2^{60.5}$ chosen plaintexts and $2^{202.2}$ encryptions. Hatano et al.'s [2002] attack of 11-round Camellia with 256-bit keys, requires 2^{93} chosen plaintexts and $2^{255.6}$ encryptions, which is just a little less than brute force search.

To conclude, 18-round Camellia is secure.

2.9 Other Ciphers That Are Not Considered

-DES [Schneier 1996] is not considered because it is inefficient, involving 3 encryptions. IDEA and SAFER$_{++}$ (of the SAFER family of ciphers) are not considered, because there are concerns in the NESSIE consortium about IDEA's key schedule, and certain structural properties of SAFER$_{++}$ [NESSIE Consortium 2003]. Among the AES finalists, MARS [Burwick et al. 1999] and Serpent [Anderson et al. 1998] are not considered. For MARS, the reason is its high algorithmic complexity, which results in the highest RAM and ROM usage among the AES finalists [Nechvatal et al. 2000; Sano et al. 2001; Schneier and Whiting 2001], and it actually received the least number of votes during the last round of voting for the AES. While Serpent was the 1st runner-up during the last round of voting, it consistently performs poorly in software encryption and decryption [Nechvatal et al. 2000; Worley et al. 2001] due to its large security margins—researchers who voted for Serpent prioritized security over performance. MARS and Serpent also have not been submitted to, nor considered by NESSIE or CRYPTREC. Blowfish [Schneier 1994] is not considered because it is superseded by Twofish, which improves on Blowfish's key schedule, and en/decryption speed [Schneier et al. 1999b]. SHACAL-2 [Handschuh and Naccache 2000] is a new 256-bit hash function-based block cipher recommended by NESSIE, but has not been been studied by NIST or CRYPTREC. The specific reasons it is not chosen for our investigation are that (1) the security offered by a key length of 256 bits is out of proportion with the relative lack of physical security in sensor nodes; (2) using 256-bit keys requires twice the storage that is required by 128-bit keys, causing an unnecessary strain on the available memory that is already scarce—note that 256-bit keys are not used in TinySec [Karlof et al. 2004] either.

3. METHODOLOGY AND CONSIDERATION

For benchmarking, we consider: (1) the cipher parameters, (2) the cipher operation modes, (3) the compiler toolchain, and (4) the implementation sources.

3.1 Cipher Parameters

Table III lists the parameters we have adopted for each cipher (some of them actually have fixed, unadjustable parameters but we list them anyway for clarity's sake). The number of rounds for each cipher is nominal except for RC5, where 18 is used instead of the nominal 12, for security reasons already described in Section 2.2. Although RC5 and RC6 allow variable word size, without the backing of relevant cryptanalytic research, we are not sure how many rounds to use if we pick a nonstandard word size of 16 bits, which is the word size of

Table III. Cipher Parameters (Lengths in Bytes)

Cipher	Skipjack	RC5	RC6	Rijndael	Twofish	MISTY	KASUMI	Camellia
Block length	8	8	16	16	16	8	8	16
Key length	10	16	16	16	16	16	16	16
Rounds	32	18	20	10	16	8	8	18

Table IV. Comparison of Operation Modes

Operation Mode	On Ciphertext Error...	On Synchronization Error...
Cipher-Block Chaining (CBC)	An erroneous bit affects the entire current block and the corresponding bit of the next block.	Lost blocks need to be retransmitted to decrypt the next block.
Cipher Feedback Mode (CFB)	An erroneous bit affects the corresponding bit of the current block and the entire next block.	Lost blocks need to be retransmitted to decrypt the next block.
Output Feedback Mode (OFB)	An erroneous bit affects the corresponding bit of the current block.	Lost blocks do not need to be retransmitted.
Counter (CTR)	Similar to OFB.	Similar to OFB.

our platform. Therefore, for RC5 and RC6, we are using the standard word size of 32 bits.

3.2 Operation Modes

The naïve approach of encrypting a message longer than one block, by dividing the message into multiple blocks and encrypting the blocks separately, is called the *electronic codebook mode* (ECB) mode. ECB is insecure since an adversary can construct valid ciphertexts from the original ciphertext by arbitrarily rearranging, repeating and/or omitting blocks from the original ciphertext. The more secure operation modes in Table IV are used in practice. These operation modes do not only affect the security, but also the energy efficiency of the encryption scheme, which will be measured in the following investigation.

As stated in Table IV, different operation modes also have different fault tolerance characteristics against ciphertext errors (where ciphertext bits are changed during transmission), and synchronization errors (where whole ciphertext blocks are lost), but we will explain why these characteristics do not really matter.

In case of a ciphertext error, a node would typically either request the corresponding packet to be retransmitted, or ignore the error. If retransmission is requested, regardless of the operation mode used, the same penalty, in the form of energy required for transmitting a packet, applies. Otherwise, using different modes only means putting up with different degrees of error, for example, CBC and CFB have to put up with more errors compared with OFB and CTR according to Table IV.

To put synchronization error into context, we first need to know how ciphertext blocks are transmitted. Usually, a network packet corresponds to one or more ciphertext blocks, and every packet contains an *initialization vector* (IV)

in the header, allowing it to be decrypted independently from other packets, as is the case with TinySec. So a lost packet does not constitute a synchronization error, unless the packet is fragmented, *and* one or more of the fragments are lost. When a node discovers it has lost a packet fragment, there are typically 3 options: (1) request the fragment to be retransmitted, (2) abort receiving all ensuing fragments, or (3) wait for the ensuing fragments to arrive naturally until a time-out. If option 1 is taken, then the same retransmission penalty applies to all operation modes. If option 2 is taken, all ensuing fragments are lost and so the same penalty in the form of information loss applies to all operation modes. If however option 3 is taken, only OFB and CTR can decrypt all received fragments (all fragments that arrived naturally) according to Table IV. That is to say, in theory, using option 3 means putting up with different degrees of information loss according to the operation mode used. For example, if 3 fragments are transmitted, each of the fragments contains 4 ciphertext blocks, and if only the 1st and 3rd fragment are successfully received, OFB and CTR would be able to decrypt all ciphertext blocks in the 1st and 3rd fragment correctly, whereas CBC and CFB would only be able to decrypt the ciphertext blocks in the 1st fragment, and the last 3 blocks in the 3rd fragment. In practice, when one fragment is lost, ensuing fragments would likely be lost too [Szewczyk et al. 2004], in which case the same information loss penalty applies to all operation modes.

Summarizing from the above analysis, in the event of a ciphertext error or synchronization error, using different modes either leads to the same (whether energy or information loss) penalty, or different information loss penalties. In other words, the different fault tolerance characteristics of operation modes do not result in different penalties in energy. We can thus safely consider the energy efficiency of an operation mode solely from an algorithmic point of view.

Some notes about our implementation: our CBC supports ciphertext stealing [Schneier 1996], so padding is not required. For CFB and OFB, we are using a feedback size that is equal to the block size.

3.3 Compilers

For compilation, we are currently only using IAR Systems' MSP430 C Compiler V2.20A/W32 (`www.iar.com`) with a patched linker, IAR Universal Linker of version 4.56F. For debugging and profiling, we use IAR Systems' Embedded Workbench 3.2 with the integrated C-Spy Debugger and profiler plug-in. Another viable compiler is the GNU C compiler in the MSPGCC toolchain (`mspgcc.sf.net`), however we are not using it due to the lack of profiling support by the toolchain. That said, we do not rule out the possibility of performing our benchmarks using the toolchain as it continues to mature in the future. Note that the chip supplier itself, Texas Instruments, offers only the Kickstart version of the toolchain we are using.

In our benchmarks, we compare maximum size optimization with maximum speed optimization. The IAR compiler supports 3 levels of optimization in terms of size or speed: High, Medium and Low, as well as 4 kinds of transformations: common subexpression elimination (`ELIM`), loop unrolling (`UNROLL`), function

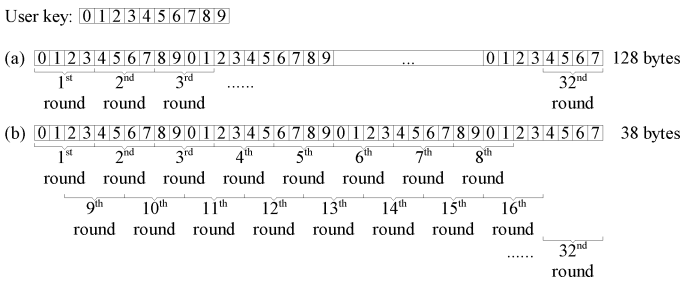

Fig. 1. Key expansion strategies for Skipjack.

inlining (`INLINE`), and code motion (`MOTION`). How these optimizations and transformations are used is described in the next section.

3.4 Implementation Sources

To avoid reinventing the wheel, we try to use and improve as much existing source code as possible. The following describes how and from where the implementation of each cipher is adapted.

The implementation of Skipjack is adapted from TinySec. Skipjack consists of executing two shift register algorithms called Rule A and Rule B: 8 rounds of Rule A, then 8 rounds of Rule B, then 8 rounds of Rule A and 8 rounds of Rule B again, resulting in a total of 32 rounds. At each round, 4 bytes of it are used, and when the key is exhausted, it is used from the beginning again (wrapped around). This offers a lot of freedom in the implementation. For example, to minimize storage, we can use exactly as many bytes for the expanded key as there are in the user key—our size-optimized implementation adopts this strategy. For another example, to minimize the number of instructions, we can expand the user key to 4×32 bytes so that the key pointer never has to wrap around (Figure 1(a))—TinySec's implementation adopts this strategy. As a compromize, we can also expand the key such that the key pointer only wraps around after 8 rounds of Rule A or Rule B (Figure 1(b))—our speed-optimized implementation adopts this strategy. Measurements show that compared to TinySec's strategy, our strategy not only saves 90 bytes of memory per expanded key but is also slightly more efficient *when speed-optimized*.

The implementation of RC5 is adapted from that of OpenSSL (`www.openssl.org`). The speed-optimized version is basically the same as the size-optimized version, with some loops unrolled.

RC6 is implemented from scratch according to the specification. The speed-optimized version does not have its loops unrolled because the resultant code size exceeds 10 kB, with marginal increase in energy efficiency.

The implementation of Rijndael is adapted from that of OpenSSL. In OpenSSL, Rijndael is fully accelerated using 10 tables, occupying 10 kB of memory (ROM in our case). Following Hachez et al.'s [1999] suggestion, we use only 4 tables—2 tables for S-boxes, 2 tables for combined S-box lookups, matrix multiplications and rotations—reducing the table space from 10 KB to 2600 bytes, in both our size- and speed-optimized implementations. In principle,

Table V. Optimizations and Transformations

Cipher	Size Optimization	Speed Optimization
Skipjack	High	High, ELIM, MOTION
RC5, RC6, Rijndael	High, ELIM, MOTION	High, ELIM
Twofish	High	High
MISTY1, KASUMI	Low, ELIM*	Low
Camellia	High, ELIM, MOTION	High, ELIM, MOTION

*Applied only to the key setup routine.

we can reduce the code size of the size-optimized version further by doing away with the last 2 tables, but the resultant code size generated by the IAR compiler far exceeds the table space saved. So the number of tables used in our case can be considered minimum.

The implementation of Twofish is adapted from Whiting's [1998] optimized implementation. Whiting's implementation by design offers 4 layers of performance tradeoffs, in decreasing en/decryption speed: (1) full keying, (2) partial keying, (3) minimal keying, and (4) zero keying. All options, except zero keying, are however impractical for MSP430F149, as the amount of table space for the key schedule is at least 1 KB, which is half the size of the RAM. Therefore, our size-optimized implementation of Twofish is based on Whiting's implementation with zero keying. Our speed-optimized implementation is also based on Whiting's implementation with zero keying, but with Worley et al.'s [2001] optimization incorporated.

The implementation of MISTY1 is adapted from Matsui and Tokita [2000]. In our size-optimized version, only the minimum number of expanded keys are stored in the RAM, while in our speed-optimized version, more expanded keys are stored to speed up en/decryption. Attemps to accelerate MISTY1 further using Botan's (`botan.randombit.net`) table-lookup technique fail, because the IAR compiler fails to terminate on medium- and high-level speed optimization.

The implementation of KASUMI is adapted from the reference implementation in the specification [3GPP 1999]. The speed-optimized version is basically the same as the size-optimized version, with some loops unrolled.

The implementation of Camellia is adapted from the reference implementation [Mitsubishi Electric Corp. 2001]. The size-optimized and speed-optimized versions are the same code, but compiled with different compiler options.

The invocation interface of each cipher follows that of OpenSSL. Table V lists the levels of optimization and kinds of transformations we use for each cipher. Furthermore, all code is compiled to use the hardware multiplier. For MISTY1 and KASUMI, the IAR compiler has trouble applying Medium/High size/speed optimization, hence only Low optimization is used. Our source code and benchmark results can be found at `http://wwwes.cs.utwente.nl/eyes/crypto_test.zip`.

4. RESULTS

We have performed our measurements in standalone mode—without interaction with an OS. We have taken care in making the interface of the cipher

implementations as uniform as possible, so that no difference in performance is a result of the difference in the interfaces. Our benchmark parameters are memory and CPU cycles. The results in terms of these two parameters are given in Section 4.1 and Section 4.2, followed immediately by our observation and analysis in Section 4.3.

4.1 Memory

We refer to two types of memory: (1) code memory, in the form of flash memory and (2) data memory, in the form of RAM. The memory organization of MSP430F149 is such that data memory ranges from address 0200h to 09FFh, whereas code memory ranges from 01100h to 0FFFFh. The IAR compiler generates 3 types of segments for MSP430: CODE, CONST and DATA. A typical policy is to put CODE and CONST segments in the code memory, and DATA segments in the data memory. Each segment type is further divided into the following subtypes:

(1) CODE: CODE (program code), CSTART and INTVEC;
(2) CONST: DATA16_AC, DATA16_C (constants including string literals), DATA16_ID (initial values of static and global variables) and DIFUNCT;
(3) DATA: CSTACK (the C runtime stack), DATA16_AN, DATA16_I (initialised variables), DATA16_N, DATA16_Z and HEAP.

Of all the above segment subtypes, we use only the underlined ones. The memory usage of these segments can be read off the list files generated by the compiler, and the results are shown in Tables VI and VII.

In Table VI, some entries have two figures separated by a comma. The first figure applies to encryption, and the second, decryption, for example, size-optimized Rijndael needs 14 bytes of RAM for setting up an encryption key, but 30 bytes for setting a decryption key. When an entry has only one figure, the figure for encryption equals the figure for decryption.

In Table VII, the code memory for each CBC module takes into account (1) the code memory for key setup, (2) the code memory for barebone encryption and decryption, (3) the code memory for lookup tables, as well as (4) the code memory for CBC-specific parts. CFB, OFB and CTR do not use the decryption function [Schneier 1996], hence the code memory for each CFB/OFB/CTR module is similarly calculated except that the code memory for decryption, decryption key setup and lookup tables for decryption are not included. Note that the storage for plaintext, ciphertext, and cipher key is not included in Table VI nor Table VII.

4.2 CPU Cycles

The computational complexity of an algorithm translates directly to its energy consumption. Assuming the energy per CPU cycle is fixed (which is justified in the Appendix), then by measuring the number of CPU cycles executed per byte of plaintext processed, we get the amount of energy consumed per byte. For example MSP430F149 draws a nominal current of 420 μA at 3 V and at a clock frequency of 1 MHz in active mode [Texas Instruments, Inc. 2001]—this means

Table VI. Data Memory Requirements

Component	Skipjack	RC5	RC6	Rijndael	Twofish	MISTY1	KASUMI	Camellia
Size-optimized:								
Expanded key	12	152	176	176	168	32	128	208
Key setup	4	64	60	14, 30	58, 50	4	58	170
CBC	54	58	94	92, 96	88, 92	56	56	142
CFB	38	42	62	60	56	40	40	110
OFB	38	42	62	60	56	40	40	110
CTR	40	44	64	62	58	42	42	112
Speed-optimized:								
Expanded key	38	152	176	176	168	64	128	208
Key setup	6	64	58	10, 26	56, 44	4	38	184
CBC	56	62, 58	98	96, 100	90, 94	60	60	146
CFB	36	42	62	60	54	40	40	110
OFB	36	42	62	60	54	40	40	110
CTR	38	44	64	62	56	42	42	112

Table VII. Code Memory Requirements

Mode	Skipjack	RC5	RC6	Rijndael	Twofish	MISTY1	KASUMI	Camellia
Size-optimized:								
CBC	1812	2076	2732	8684	9612	7972	9842	19864
CFB	850	1104	1328	4276	7786	4538	5446	12382
OFB	778	1032	1256	4204	7714	4466	5374	12310
CTR	856	1110	1406	4354	7864	4544	5452	12460
Speed-optimized:								
CBC	2610	7502	3174	9984	12538	8596	11078	29516
CFB	1068	4290	1340	4590	9302	4906	5994	17148
OFB	988	4210	1260	4510	9222	4826	5914	17068
CTR	1066	4288	1412	4662	9374	4904	5992	17220

that the energy per instruction cycle (*for the processor alone*) is theoretically 1.26 nJ.

To evaluate the ciphers, we must determine the range of plaintext lengths that is of greatest interest to WSNs. Here is how we arrive at the choice of 8 to 96 bytes. The lower limit of 8 bytes is close to the minimum packet size (usually the size of control packets) used in mainstream link-layer protocols [Ye et al. 2002; van Dam and Langendoen 2003; Polastre et al. 2004]. The upper limit of 96 bytes is close to the maximum packet size used for simulating and benchmarking WSN protocols [Polastre et al. 2004; van Dam and Langendoen 2003].

Figure 2 shows our measurements for CBC encryption. CBC decryption consumes a slightly different number of CPU cycles, but the relative ordering between the various ciphers remains the same. Since an RC5/RC6 en/decryption executes a different number of rotations depending on the key and the plaintext/ciphertex, the figures for RC5 and RC6 in Figure 2 are obtained by averaging over 450 encryptions using a different pseudorandomly generated key and plaintext each time. Only 450 encryptions are used because (1) the keys and plaintexts are pseudorandomly generated offline and downloaded to the 60 KB ROM of the MSP430F149, and (2) they can be done in under a minute on

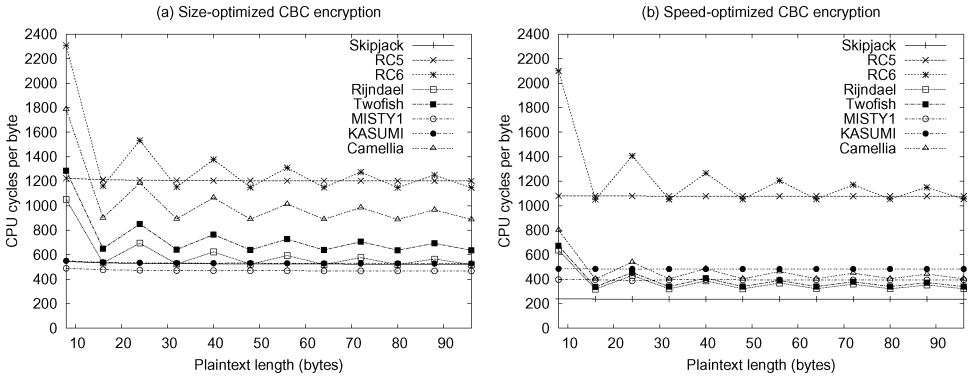

Fig. 2. CPU usage of CBC encryption when (a) size-optimized, and (b) speed-optimized.

the profiler. That the plots for the 64-bit block ciphers: Skipjack, RC5, MISTY1 and KASUMI, appear as straight lines, should be interpreted as follows: the measurements are only taken at full block lengths for these ciphers. Were the measurements taken at nonfull block lengths, the plots for these ciphers would appear jagged-like, just as the plots for other, 128-bit block ciphers do. When size-optimized, the CPU cycles per byte of Rijndael, Skipjack and KASUMI are only a few clock cycles apart, with Rijndael slightly more efficient than Skipjack, and Skipjack slightly more efficient than KASUMI at large packet sizes.

For CFB, OFB, and CTR, with the exception of RC5 and RC6, we found that the number of CPU cycles consumed per byte, y, can be approximated by

$$y \approx \frac{C_f + xC_b + \lceil \frac{x}{B} \rceil C_B}{x} \tag{2}$$

where x is the plaintext length, B is the cipher's block length (both in bytes) and C_f, C_b, and C_B are constants. In Equation 2, the first term, C_f, accounts for the function call overhead, the second term, xC_b, accounts for the overhead of organizing x bytes into B-byte blocks, and the last term accounts for the actual en/decryption process, and in CTR's case the increment of a counter. This approximation does not apply to CBC because with ciphertext stealing, CBC involves more complicated computation. This approximation does not apply to RC5 and RC6 either because an RC5/RC6 en/decryption executes a different number of rotations depending on the data and the key, resulting in a nonconstant value for C_B. Thus for RC5 and RC6, the following equation is more appropriate:

$$y \approx \frac{C_f + xC_b + \lceil \frac{x}{B} \rceil V_B}{x} \tag{3}$$

where V_B is a variable. Table VIII lists the values of C_f, C_b, C_B (for Skipjack, Rijndael, Twofish, MISTY1, KASUMI and Camellia), and the average value of V_B (for RC5 and RC6) obtained through least squares fitting. The standard error of each of the constants is less than 10^{-11}. Note that CFB, OFB and CTR consume exactly the same number of CPU cycles for both encryption and decryption.

Table VIII. Values of C_f, C_b, C_B (or V_B) for CFB, OFB and CTR

Mode	Param.	Skipjack	RC5	RC6	Rijndael	Twofish	MISTY1	KASUMI	Camellia
Size-optimized:									
CFB	C_f	94	94	94	94	94	94	94	94
	C_b	51	51	51	51	51	51	51	51
	$C_B(V_B)$	3550	9241	16924	6537	8230	3363	3849	11949
OFB	C_f	82	82	82	82	82	82	82	82
	C_b	27	27	27	27	27	27	27	27
	$C_B(V_B)$	3550	9241	16923	6537	8230	3363	3849	11949
CTR	C_f	90	90	90	90	90	90	90	90
	C_b	33	33	33	33	33	33	33	33
	$C_B(V_B)$	3593	9285	16982	6594	8287	3406	3892	12006
Speed-optimized:									
CFB	C_f	86	86	86	86	86	86	86	86
	C_b	35	35	35	35	35	35	35	35
	$C_B(V_B)$	1594	8322	16279	4573	4866	2859	3561	5885
OFB	C_f	82	82	82	82	82	82	82	82
	C_b	27	27	27	27	27	27	27	27
	$C_B(V_B)$	1594	8322	16278	4573	4866	2859	3561	5885
CTR	C_f	90	90	90	90	90	90	90	90
	C_b	33	33	33	33	33	33	33	33
	$C_B(V_B)$	1637	8366	16328	4622	4915	2902	3604	5934

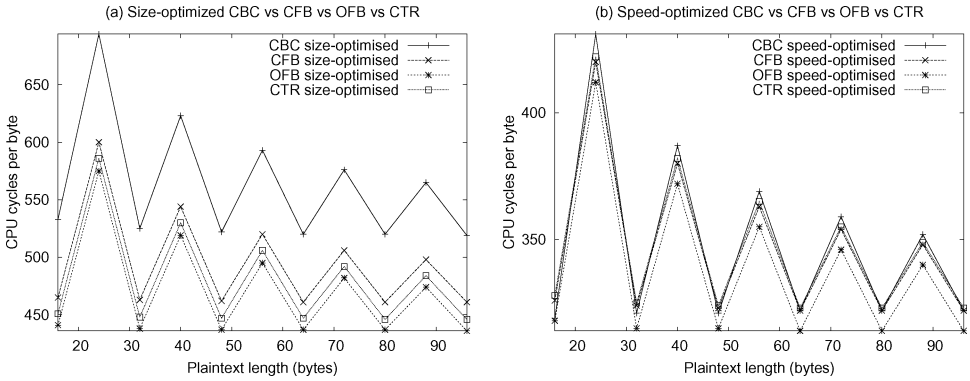

Fig. 3. CBC vs CFB vs OFB vs CTR for Rijndael encryption when (a) size-optimized, and (b) speed-optimized.

We now look at the operation modes. Although Figure 3 only compares the operation modes in the context of Rijndael encryption, the same comparison applies to all other ciphers. Figure 3 shows that in terms of efficiency, OFB > CTR > CFB > CBC when size-optimized ('>' meaning 'is more efficient than'). When speed-optimized, OFB is still the most efficient mode, but CBC > CFB > CTR at plaintext lengths = integer × block size, and CFB > CTR > CBC at other plaintext lengths—this is when ciphertext stealing is activated, thereby degrading the efficiency of CBC. Anyhow the performance difference between all modes is smaller when they are speed-optimized than when they are size-optimized.

Table IX. CPU Cycles for Key Setup (per Key)

Operation	Skipjack	RC5	RC6	Rijndael	Twofish	MISTY1	KASUMI	Camellia
Size-optimized:								
Enc	163	44927	45701	*1637	+11187	977	2564	23211
Dec	163	44927	45701	9832	10696	977	2564	23211
Enc→Dec	0	0	0	8195	10696	0	0	0
Dec→Enc	0	0	0	1637	491	0	0	0
Speed-optimized:								
Enc	187	40579	43246	1195	8049	615	1485	15335
Dec	187	40579	43246	5591	7553	615	1485	15335
Enc→Dec	0	0	0	4396	7553	0	0	0
Dec→Enc	0	0	0	1195	496	0	0	0

*Applicable only to CBC mode; 1717 cycles in CFB/OFB/CTR mode.
+Applicable only to CBC mode; 10707 cycles in CFB/OFB/CTR mode.
Note:

(1) Enc (Dec) = setting up an encryption (decryption) key from scatch.

(2) Enc→Dec (Dec→Enc) = converting an encryption (decryption) key to a decryption (encryption) key.

Apart from en/decryption, we are also interested in the efficiency of the key setup algorithms. Table IX has the results. Notice that only Rijndael and Twofish incur overhead when converting an encryption key to a decryption key, and converting a decryption key to an encryption key. For Skipjack, the speed-optimized version performs worse than the size-optimized version because the expanded key used in the speed-optimized version is 38 bytes long compared with 12 bytes in the size-optimized version (the difference in expanded key length is explained in the next section). Using a longer expanded key allows en/decryptions to execute faster.

4.3 Observation and Analysis

First about the operation mode. According to the results in the previous section, OFB is the most energy-efficient mode, and CBC is the least energy-efficient mode when there are partial (plaintext/ciphertext) blocks. OFB is the obvious choice if we only consider the case of two communicating parties. However energy-efficient communications in WSNs often require more than two parties to be involved in a secure group, for example in the form of *passive participation* [Zhu et al. 2003]. In passive participation, a node decides whether to transmit its own packets based on the packets it received from its neighbours—not reporting data that are superseded by a neighbour's, helps save energy.

Using OFB, and for that matter CFB and CTR, effectively in a group setting is problematic. For example in a group composed of nodes A, B and C, all sharing the same key K and the same IV n (or counter in case of CTR mode), when A broadcasts a packet encrypted with K and n, if B succeeds but C fails to receive the packet, C would not know it should use a different IV than n. If C decides to send a packet encrypted with K and the same IV n, all is lost because the IV in the OFB/CFB/CTR mode can never be reused. A suggestion to overcome

this is to use a longer IV and to partition the IV space by node ID. In this approach, the IV can be variable in length if the group communication protocol is to dynamically accommodate a variable number of members at the expense of protocol complexity. If the IV is fixed according to the maximum supported group size, bandwidth and energy are wasted whenever the actual group size is lower than the maximum supported group size. Either way, using OFB in the group setting is awkward and not energy-efficient. CBC in contrast is easier and safer to use for group communications [Karlof et al. 2004]. CBC is not the most efficient mode, but we may find comfort in the fact that CBC is actually quite close to other operation modes in efficiency when speed-optimized, and the difference only gets smaller with increasing plaintext length according to Equation 2.

Next we analyze Figure 2 and Tables VI, VII, VIII, and IX cipher by cipher:

Skipjack: Skipjack produces the shortest expanded key, and requires the least code and data memory. When size-optimized, it is slightly less energy-efficient than Rijndael, but when speed-optimized, it is the most energy-efficient cipher.

RC5: RC5 requires little code memory but has poor energy efficiency because multiplication and rotation are the Archilles' heel of MSP430F149: multiplication takes 9 cycles and rotation can only be done one bit at a time. It is for the same reason that speed optimization does not improve the energy efficiency of RC5 significantly.

RC6: Like RC5, RC6 is lean in code size. The code size is in the range of 1 KB, close to the code size of Sano et al.'s [2001] implementation on the ZiLOG Z80 8-bit microprocessor. In terms of CPU cycles, our measurements are not far from Hachez et al.'s [1999] measurements on the 8-bit processor Intel 8051: when speed-optimized, key setup takes 43246 clock cycles on MSP430F149, compared to 43200 on 8051; encryption of one block takes 16265 on MSP430F149, compared to 14400 on 8051. For the same reason that applies to RC5, RC6 is poor in energy efficiency, and is barely improved by speed optimization. The observation that RC6 is a big RAM consumer and performs poorly on 8/16-bit architectures such as the MSP430F149, is confirmed by benchmarks done over a spectrum of architectures [Hachez et al. 1999; Nechvatal et al. 2000; Schneier and Whiting 2001].

Rijndael: Rijndael has moderate code size for the number of tables we use in the implementation. It is the second most energy-efficient cipher both when size-optimized and when speed-optimized.

Twofish: Twofish has larger code size than Rijndael. The tables used by Twofish cannot be economized without seriously degrading its energy efficiency. Twofish is almost as energy-efficient as Rijndael in en/decryption when speed-optimized, but is significantly worse in key agility.

MISTY1: MISTY1 has moderate code size, larger than RC5 and RC6, but smaller than Rijndael. Its RAM usage is only larger than Skipjack's. When size-optimized, MISTY1 is the most energy-efficient cipher. There is no significant speed increase in the speed-optimized version mainly because the IAR compiler has a problem applying any level of optimization above Low.

Table X. Ranking of Ciphers*

Code Memory		Data Memory		En/decryption Efficiency		Key Setup Efficiency	
CBC	OFB	CBC	OFB	CBC	OFB	Encryption	Decryption
$Skipjack_z$	$Skipjack_z$	$Skipjack_z$	$Skipjack_z$	$Skipjack_s$	$Skipjack_s$	$Skipjack_z$	$Skipjack_z$
$RC5_z$	$Skipjack_s$	$MISTY1_z$	$MISTY1_z$	$Rijndael_s$	$Rijndael_s$	$Skipjack_s$	$Skipjack_s$
$Skipjack_s$	$RC5_z$	$Skipjack_s$	$Skipjack_s$	$Twofish_s$	$Twofish_s$	$MISTY1_s$	$MISTY1_s$
$RC6_z$	$RC6_z$	$MISTY1_s$	$MISTY1_s$	$MISTY1_s$	$MISTY1_s$	$MISTY1_z$	$MISTY1_z$
$RC6_s$	$RC6_s$	$KASUMI_z$	$KASUMI_s$	$Camellia_s$	$Camellia_s$	$Rijndael_s$	$KASUMI_s$
$RC5_s$	$Rijndael_z$	$KASUMI_s$	$KASUMI_z$	$MISTY1_z$	$Rijndael_z$	$KASUMI_s$	$KASUMI_z$
$MISTY1_z$	$RC5_s$	RC5	RC5	$KASUMI_s$	$MISTY1_z$	$Rijndael_z$	$Rijndael_s$
$MISTY1_s$	$MISTY1_z$	$Twofish_z$	$Twofish_s$	$Rijndael_z$	$Skipjack_z$	$KASUMI_z$	$Twofish_s$
$Rijndael_z$	$Rijndael_s$	$Twofish_s$	$Twofish_z$	$Skipjack_z$	$KASUMI_s$	$Twofish_s$	$Rijndael_z$
$Twofish_z$	$MISTY1_s$	$RC6_z$	Rijndael	$KASUMI_z$	$KASUMI_z$	$Twofish_z$	$Twofish_z$
$KASUMI_z$	$KASUMI_z$	$Rijndael_z$	RC6	$Twofish_z$	$Twofish_z$	$Camellia_s$	$Camellia_s$
$Rijndael_s$	$KASUMI_s$	$RC6_s$	$Camellia_z$	$Camellia_z$	$Camellia_z$	$Camellia_z$	$Camellia_z$
$KASUMI_s$	$Twofish_z$	$Rijndael_s$	$Camellia_s$	$RC6_s$	$RC6_s$	$RC5_s$	$RC5_s$
$Twofish_s$	$Twofish_s$	$Camellia_z$		$RC5_s$	$RC5_s$	$RC6_s$	$RC6_s$
$Camellia_z$	$Camellia_z$	$Camellia_s$		$RC6_z$	$RC6_z$	$RC5_z$	$RC5_z$
$Camellia_s$	$Camellia_s$			$RC5_z$	$RC5_z$	$RC6_z$	$RC6_z$

*Subscript z means size-optimized, s means speed-optimized.

KASUMI: KASUMI has larger code size than MISTY1 and Rijndael. Although the key setup of KASUMI is linear [Dunkelman 2002], it takes more than 2 times as long as that of MISTY1 which is nonlinear. KASUMI is less energy-efficient than MISTY1, since it is algorithmically more complicated, and speed optimization does not provide significant improvement for the same reason given previously for MISTY1.

Camellia: Although Camellia is more energy-efficient than RC5 and RC6, it has the largest code and data memory requirements. The expanded key occupies more than 10% of the RAM, but computing the round subkeys on the fly would significantly worsen the the energy efficiency in en/decryption, which is already lacking.

To conclude our observations, we now discuss the ranking of the ciphers in Table X. Skipjack is the winner in all categories. The fact that RC5 and RC6 have small code size does not help them in their overall ranking, as they range from moderate to poor in all other categories. Rijndael and Twofish fare well if only en/decryption efficiency is taken into account. MISTY1 is excellent in key agility and data memory usage, and is moderate in code size. If the expanded key is not kept in the RAM and must be generated on the fly, for example when the amount of RAM available to security is scarce, MISTY1 has better en/decryption efficiency than Twofish. KASUMI is an average performer in all categories. Camellia is moderate when it comes to en/decryption efficiency, but does poorly in all other categories.

5. CONCLUSION

First about the operation mode. The OFB mode should be used on pairwise secure links, for example, for mutual authentication between a base station and a node. The CBC mode should be used for group communications, for example, in passive participation, as mentioned in the previous section.

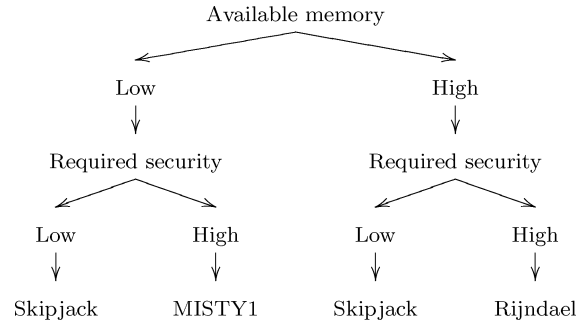

Fig. 4. Selection of an energy-efficient cipher under the constraints of available memory and required security.

On the most suitable cipher to use, we will reach our verdict by first ruling out the unlikely candidates. First we would like to emphasise that MSP430F149 is one of the most high-end in the Texas Instrument's MSP430 series. In other words, a total code memory of 59.7 KB and data memory of 2 KB is a hard limit in the MSP430 family of processors. For this reason, we first rule out Camellia, which occupies 1/5 of the total code memory even after size optimization, and even though it has decent energy efficiency for en/decryption when speed-optimized. The next to be ruled out are RC5 and RC6, which have poor energy efficiency and key agility (the ability to change keys quickly and with a minimum amount of resources). KASUMI lags behind MISTY1 in all categories, so we should consider MISTY1 instead of KASUMI. Rijndael and Twofish require about the same amount of data memory, but Rijndael has a smaller code size and better en/decryption efficiency, so Rijndael instead of Twofish should be considered. Finally the verdict is given in the form of Figure 4.

Reviewing some of the proposals in the literature so far, we conclude that there are better options to the use of RC5 and RC6 in WSNs. We note the fact that we use 18 instead of 12 rounds for RC5 based on our security analysis, may explain the difference in performance perceived by us and other researchers. One discouraging factor against the use of RC5 or RC6 is that most embedded processors do not support the variable-bit rotation instruction like ROL of the Intel architecture [Intel Corporation 1997], which RC5 and RC6 are designed to take advantage of. Another, nontechnical, discouraging factor is that they are patented.

In conclusion, we have presented a detailed benchmark for one of the most important cryptographic primitives for WSNs: block ciphers. Taking into account the security properties, storage- and energy-efficiency of a set of candidates, we have arrived at a systematically justifiable selection of block ciphers and operation modes.

ACKNOWLEDGMENTS

The authors would like to thank Adrian Perrig and the anonymous reviewers for their inspiring comments, which have vastly improved this article.

REFERENCES

3GPP. 1999. Specification of the 3GPP Confidentiality and Integrity Algorithms Document 2: KASUMI Specification. ETSI/SAGE Specification Version: 1.0.

ANDERSON, R., BIHAM, E., AND KNUDSEN, L. 1998. Serpent: A Proposal for the Advanced Encryption Standard. http://www.cl.cam.ac.uk/ftp/users/rja14/serpent.pdf.

AOKI, K., ICHIKAWA, T., KANDA, M., MATSUI, M., MORIAI, S., NAKAJIMA, J., AND TOKITA, T. 2001b. Camellia: A 128-Bit Block cipher suitable for multiple platforms. In *Proceedings of the Selected Areas in Cryptography (SAC'00)*, D. Stinson and S. Tavares, Eds. Number 2012 in LNCS. Springer-Verlag, 39–56.

AOKI, K., ICHIKAWA, T., KANDA, M., MATSUI, M., MORIAI, S., NAKAJIMA, J., AND TOKITA, T. 2001a. Specification of Camellia—A 128-Bit Block Cipher. Specification Version 2.0, Nippon Telegraph and Telephone Corporation and Mitsubishi Electric Corporation.

BABBAGE, S. AND FRISCH, L. 2001. On MISTY1 higher order differential cryptanalysis. In *3rd International Conference on Information Security and Cryptology, ICISC 2000*. LNCS, vol. 2015. Springer-Verlag, 22–36.

BARKAN, E. AND BIHAM, E. 2002. In how many ways can you write rijndael. In *Advances in Cryptology—ASIACRYPT 2002: 8th International Conference on Theory and Application of Cryptology and Information Security*, Y. Zheng, Ed. LNCS, vol. 2501. Springer-Verlag, 160–175.

BIHAM, E., BIRYUKOV, A., AND SHAMIR, A. 1999. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In *Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques*. LNCS, vol. 1592. Springer-Verlag, 12–23.

BIHAM, E. AND FURMAN, V. 2000. Improved impossible differentials on twofish. In *Progress in Cryptology—INDOCRYPT 2000: First International Conference in Cryptology in India*. LNCS, vol. 1977. Springer-Verlag, 80–92.

BIRYUKOV, A. 1999. Methods of cryptanalysis. Ph.D. thesis, Technion.

BIRYUKOV, A. AND KUSHILEVITZ, E. 1998. Improved Cryptanalysis of RC5. In *Advances in Cryptology—EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques*. LNCS, vol. 1403. Springer-Verlag, 85–99.

BIRYUKOV, A. AND WAGNER, D. 2000. Advanced slide attacks. In *Advances in Cryptology—EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques*. LNCS, vol. 1807. Springer-Verlag, 589–606.

BORST, J., PRENEEL, B., AND VANDEWALLE, J. 1999. Linear cryptanalysis of RC5 and RC6. In *Fast Software Encryption, 6th International Workshop, FSE '99*, L. Knudsen, Ed. LNCS, vol. 1636. Springer-Verlag, 16–30.

BURWICK, C., COPPERSMITH, D., D'AVIGNON, E., GENNARO, R., HALEVI, S., JUTLA, C., JR., S. M. M., O'CONNOR, L., PEYRAVIAN, M., SAFFORD, D., AND ZUNIC, N. 1999. MARS—a candidate cipher for AES. http://researchweb.watson.ibm.com/security/mars.pdf.

CARMAN, D., KRUUS, P., AND MATT, B. 2000. Constraints and approaches for distributed sensor network security. Tech. Rep. #00-010, NAI Labs.

CHEON, J., KIM, M., KIM, K., AND J.-Y. LEE, S. W. K. 2002. Improved impossible differential cryptanalysis of rijndael and crypton. In *4th International Conference on Information Security and Cryptology, ICISC 2001*, K. Kim, Ed. LNCS, vol. 2288. Springer-Verlag, 39–49.

CHIEN, P. and Wen, V. 1998. CS199—StrongARM Energy Measurement Report. Online slides: http://www.cs.berkeley.edu/~vwen/strongarm/slides/cs199.ppt.

COPPERSMITH, D. 2002. Re: Impact of Courtois and Pieprzyk results. Forum message at http://aes.nist.gov/aes/.

COURTOIS, N., GOUBIN, L., MEIER, W., AND TACIER, J.-D. 2002. Solving underdefined systems of multivariate quadratic equations. In *PKC 2002*. LNCS, vol. 2274. Springer-Verlag, 211–227.

COURTOIS, N., JOHNSON, R., JUNOD, P., PORNIN, T., AND SCOTT, M. 2003. Did Filiol Break AES? Cryptology ePrint Archive: Report 2003/022.

COURTOIS, N. AND PATARIN, J. 2003. About the XL Algorithm over GF(2). In *Topics in Cryptology—CT-RSA 2003, The Cryptographers' Track at the RSA Conference 2003*, M. Joye, Ed. LNCS, vol. 2612. Springer-Verlag, 141–157.

COURTOIS, N. AND PIEPRZYK, J. 2002a. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. Cryptology ePrint Archive: Report 2002/044.

COURTOIS, N. AND PIEPRZYK, J. 2002b. Cryptanalysis of block ciphers with overdefined systems of equations. In *Advances in Cryptology—ASIACRYPT 2002: 8th International Conference on Theory and Application of Cryptology and Information Security*, Y. Zheng, Ed. LNCS, vol. 2501. Springer-Verlag, 267–287.

CRYPTREC. 2001. Analysis of RC6. 暗号アルゴリズム及び関連技術の評価報告 (trans.: Evaluation report of cryptographic algorithms and related technologies) no. 1086.

CRYPTREC. 2003. 電子政府推奨暗語の仕様書 (trans.: Specification of e-government-recommended ciphers). `http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/cryptrec20030425_spec01%.html`.

DAEMEN, J., KNUDSEN, L., AND RIJMEN, V. 1997. The block Cipher SQUARE. In *Fast Software Encryption, 4th International Workshop, FSE '97*, E. Biham, Ed. LNCS, vol. 1267. Springer-Verlag, 149–165.

DAEMEN, J. AND RIJMEN, V. 1999. AES Proposal: Rijndael.

DUNKELMAN, O. 2002. Comparing MISTY1 and KASUMI. NESSIE Public Report NES/DOC/TEC/WP5/029/a, Computer Science Department, Technion. Dec.

FERGUSON, N., KELSEY, J., LUCKS, S., SCHNEIER, B., STAY, M., WAGNER, D., AND WHITING, D. 2001a. Improved Cryptanalysis of Rijndael. In *Fast Software Encryption, 7th International Workshop, FSE 2000*, B. Schneier, Ed. LNCS, vol. 1978. Springer-Verlag, 213–230.

FERGUSON, N., SCHROEPPEL, R., AND WHITING, D. 2001b. A Simple Algebraic Representation of Rijndael. In *Selected Areas in Cryptography, 8th Annual International Workshop, SAC 2001*. LNCS, vol. 2259. Springer-Verlag, 103–111.

FILIOL, E. 2003. Plaintext-Dependant Repetition Codes Cryptanalysis of Block Ciphers—The AES Case. Cryptology ePrint Archive: Report 2003/003.

FULLER, J. AND MILLAN, W. 2002. On Linear Redundancy in the AES S-Box. Cryptology ePrint Archive: Report 2002/111.

GILBERT, H., HANDSCHUH, H., JOUX, A., AND VAUDENAY, S. 2000. A statistical attack on RC6. In *Fast Software Encryption, 7th International Workshop, FSE 2000*. LNCS, vol. 1978. Springer-Verlag, 64–74.

GILBERT, H. AND MINIER, M. 2000. A collision attack on 7 rounds of Rijndael. In *Proceedings of the 3rd AES Conference (AES3)*.

HACHEZ, G., KOEUNE, F., AND QUISQUATER, J.-J. 1999. cAESar results: Implementation of four AES candidates on two smart cards. In *2nd AES Candidate Conference (AES2)*.

HANDSCHUH, H. AND HEYS, H. 1998. A timing attack on RC5. In *Selected Areas in Cryptography '98, SAC'98*, S. Tavares and H. Meijer, Eds. LNCS, vol. 1556. Springer-Verlag, 306–318.

HANDSCHUH, H. AND NACCACHE, D. 2000. SHACAL. In *Proceedings of the First Open NESSIE Workshop*.

HATANO, Y., SEKINE, H., AND KANEKO, T. 2002. Higher order differential attack of *Camellia*(II). In *Selected Areas in Cryptography. 9th Annual International Workshop, SAC 2002*, K. Nyberg and H. Heys, Eds. LNCS, vol. 2595. Springer-Verlag, 129–146.

HE, Y. AND QING, S. 2001. Square Attack on Reduced Camellia Cipher. In *Information and Communications Security: Third International Conference, ICICS 2001*, S. Qing, T. Okamoto, and J. Zhou, Eds. LNCS, vol. 2229. Springer-Verlag, 238–245.

HILL, J., SZEWCZYK, R., WOO, A., HOLLAR, S., CULLER, D., AND PISTER, K. 2000. System architecture directions for networked sensors. *SIGOPS Oper. Syst. Rev. 34,* 5, 93–104.

IEEE. 2003. IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs).

Intel Corporation 1997. *Intel Architecture Software Developer's Manual Volume 2: Instruction Set Reference*. Intel Corporation.

KALISKI, B. AND YIN, Y. 1998. On the Security of the RC5 Encryption Algorithm. Tech. Rep. TR-602, RSA Laboratories. Sept.

KANG, J.-S., SHIN, S.-U., HONG, D., AND YI, O. 2001a. Provable security of KASUMI and 3GPP encryption mode $f8$. In *Advances in Cryptology—ASIACRYPT 2001: 7th International Conference*

on the Theory and Application of Cryptology and Information Security, C. Boyd, Ed. LNCS, vol. 2248. Springer-Verlag, 255–271.

KANG, J.-S., YI, O., HONG, D., AND CHO, H. 2001b. Pseudorandomness of MISTY-Type Transformations and the Block Cipher KASUMI. In *Proceedings of the 6th Australasian Conference on Information Security and Privacy, ACISP 2001*, V. Varadharajan and Y. Mu, Eds. LNCS, vol. 2119. Springer-Verlag, 60–73.

KARLOF, C., SASTRY, N., AND WAGNER, D. 2004. TinySec: A link layer security architecture for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*. ACM Press, New York, NY, USA, 162–175.

KARLOF, C. AND WAGNER, D. 2003. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's Ad Hoc Networks Journal, Special Issue on Sensor Network Applications and Protocols 1*, 2–3, 293–315.

KEATING, G. 1999. Performance Analysis of AES candidates on the 6805 CPU core. In *2nd AES Candidate Conference (AES2)*.

KELSEY, J. 2000. Key Separation in Twofish. Tech. Rep. #7, Counterpane Internet Security, Inc. Apr.

KELSEY, J., SCHNEIER, B., WAGNER, D., AND HALL, C. 1998. Side channel cryptanalysis of product ciphers. In *Computer Security (ESORICS'98)*. LNCS, vol. 1485. Springer-Verlag, 97–110.

KILIAN, J. AND ROGAWAY, P. 1996. How to protect DES against exhaustive key search. In *Advances in Cryptology—CRYPTO '96: 16th Annual International Cryptology Conference*. Number 1109 in LNCS. Springer-Verlag.

KLING, R. 2003. Intel mote: An Enhanced Sensor Network Node. In *International Workshop on Advanced Sensors, Structural Health Monitoring and Smart Structures*.

KNUDSEN, L. AND MEIER, W. 2000. Correlations in RC6 with a reduced number of rounds. In *Fast Software Encryption, 7th International Workshop, FSE 2000*. LNCS, vol. 1978. Springer-Verlag, 94–108.

KNUDSEN, L. AND WAGNER, D. 2002. Integral cryptanalysis. In *Fast Software Encryption, 9th International Workshop, FSE 2002*, J. Daemen and V. Rijmen, Eds. LNCS, vol. 2365. Springer-Verlag, 112–127.

KÜHN, U. 2001. Cryptanalysis of reduced-round MISTY. In *Advances in Cryptology—EUROCRYPT 2001*. LNCS, vol. 2045. Springer-Verlag, 325–339.

KÜHN, U. 2002. Improved Cryptanalysis of MISTY1. In *Fast Software Encryption, 9th International Workshop, FSE 2002*. LNCS, vol. 2365. Springer-Verlag, 61–75.

LEE, S., HONG, S., LEE, S., LIM, J., AND YOON, S. 2002. Truncated differential cryptanalysis of Camellia. In *4th International Conference on Information Security and Cryptology, ICISC 2001*, K. Kim, Ed. LNCS, vol. 2288. Springer-Verlag, 32–38.

LENSTRA, A. K. AND VERHEUL, E. R. 2001. Selecting cryptographic key sizes. *Journal of Cryptology 14*, 4, 255–293.

LI, T., WU, H., WANG, X., AND BAO, F. 2005. SenSec Design. Tech. Rep. TR-I2R-v1.1, InfoComm Security Department,Institute for Infocomm Research. Feb.

LIU, D., NING, P., AND LI, R. 2005. Establishing pairwise keys in distributed sensor networks. *ACM Trans. Inf. Syst. Secur. 8*, 1, 41–77.

LUCKS, S. 2002. The saturation attack—A Bait for Twofish. In *Fast Software Encryption, 8th International Workshop, FSE 2001*. LNCS, vol. 2355. Springer-Verlag, 1–15.

MATSUI, M. 1993. Linear Cryptanalysis of DES. In *Advances in Cryptology—EUROCRYPT '93: Workshop on the Theory and Application of Cryptographic Techniques*. LNCS, vol. 765. Springer-Verlag, 386–397.

MATSUI, M. 1997. New Block Encryption Algorithm MISTY. In *Fast Software Encryption, 4th International Workshop, FSE '97*, E. Biham, Ed. LNCS, vol. 1267. Springer-Verlag, 54–68.

MATSUI, M. AND TOKITA, T. 2000. MISTY, KASUMI and Camellia Cipher Algorithm. *Mitsubishi Electric ADVANCE (Cryptography Edition) 100*, 2–8.

MIRZA, F. AND MURPHY, S. 1999. An observation on the key schedule of twofish. In *Proceedings of the 2nd AES Conference (AES2)*.

MITSUBISHI ELECTRIC CORP. 2001. `http://info.isl.ntt.co.jp/crypt/camellia/dl/camellia.c`.

MIYAJI, A., NONAKA, M., AND TAKII, Y. 2002. Known plaintext correlation attack against RC5. In *Topics in Cryptology—CT-RSA 2002, The Cryptographers' Track at the RSA Conference 2002*, B. Preneel, Ed. LNCS, vol. 2271. Springer-Verlag, 131–148.

MOH, T. 2002. On the Courtois-Pieprzyk's Attack on Rijndael . Web page: `http://www.usdsi.com/aes.html`.

MURPHY, S. 2000. The key Separation of twofish. In *Proceedings of the 3rd AES Conference (AES3)*.

MURPHY, S. AND ROBSHAW, M. 2002a. Comments on the Security of the AES and the XSL Technique. `http://www.isg.rhul.ac.uk/~mrobshaw/rijndael/xslnote.pdf`.

MURPHY, S. AND ROBSHAW, M. 2002b. Essential algebraic structure within the AES. In *Advances in Cryptology—CRYPTO 2002, 22nd Annual International Cryptology Conference*, M. Yung, Ed. LNCS, vol. 2442. Springer-Verlag, 1–16.

MURPHY, S. AND ROBSHAW, M. 2002c. Key-dependent s-boxes and differential cryptanalysis. *Des. Codes Cryptography 27*, 3, 229–255.

NECHVATAL, J., BARKER, E., BASSHAM, L., BURR, W., DWORKIN, M., FOTI, J., AND ROBACK, E. 2000. Report on the Development of the Advanced Encryption Standard (AES). Tech. rep., NIST.

NESSIE CONSORTIUM 2003. *Portfolio of recommended cryptographic primitives*. NESSIE Consortium.

NIST 1998. *Skipjack and KEA Algorithm Specifications Version 2.0*. NIST.

NYBERG, K. 1995. Linear approximations of block ciphers. In *Advances in Cryptology—EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques*. LNCS, vol. 950. Springer-Verlag, 439–444.

OHTA, H. AND MATSUI, M. 2000. A Description of the MISTY1 Encryption Algorithm. RFC 2994, Network Working Group, IETF. Nov.

PERRIG, A., SZEWCZYK, R., WEN, V., CULLER, D., AND TYGAR, J. 2001. SPINS: Security protocols for sensor networks. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*. ACM Press, 189–199.

POLASTRE, J., HILL, J., AND CULLER, D. 2004. Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM Press, 95–107.

PRENEEL, B. 1998. Cryptographic primitives for information authentication—state of the art. In *State of the Art in Appplied Cryptography*, B. Preneel and V. Rijmen, Eds. LNCS, vol. 1528. Springer-Verlag, 50–105.

PRENEEL, B., BIRYUKOV, A., OSWALD, E., ROMPAY, B. V., GRANBOULAN, L., DOTTAX, E., MURPHY, S., DENT, A., WHITE, J., DICHTL, M., PYKA, S., SCHAFHEUTLE, M., SERF, P., BIHAM, E., BARKAN, E., DUNKELMAN, O., QUISQUATER, J.-J., CIET, M., SICA, F., KNUDSEN, L., PARKER, M., AND RADDUM, H. 2003. NESSIE Security Report. Deliverable D20, NESSIE Consortium. Feb.

REICHARDT, B. AND WAGNER, D. 2002. Markov truncated differential cryptanalysis of skipjack. In *Selected Areas in Cryptography: 9th Annual International Workshop (SAC 2002)*. LNCS, vol. 2595. Springer-Verlag, 110–128.

RIVEST, R. 1995. The RC5 Encryption Algorithm. In *Proceedings of the 1994 Leuven Workshop on Fast Software Encryption*. Springer-Verlag, 86–96.

RIVEST, R., ROBSHAW, M., SIDNEY, R., AND YIN, Y. 1998. The RC6^{TM} Block Cipher. Specification version 1.1.

SANO, F., KOIKE, M., KAWAMURA, S., AND SHIBA, M. 2001. Performance evaluation of aes finalists on the high-end smart card. In *Proceedings of the 3rd AES Conference (AES3)*.

SCHNEIER, B. 1994. Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish). In *Fast Software Encryption, Cambridge Security Workshop Proceedings*. LNCS. Springer-Verlag, 191–204.

SCHNEIER, B. 1996. *Applied Cryptography: Protocols, Algorithms and Source Code in C*, 2nd ed. John Wiley & Sons, Inc.

SCHNEIER, B. 2002a. AES News. Crypto-gram newsletter, Counterpane Internet Security, Inc. Sept.

SCHNEIER, B. 2002b. More on AES Cryptanalysis. Crypto-gram newsletter, Counterpane Internet Security, Inc. Oct.

SCHNEIER, B., KELSEY, J., WHITING, D., WAGNER, D., HALL, C., AND FERGUSON, N. 1998. Twofish: A 128-Bit Block Cipher. http://www.schneier.com/paper-twofish-paper.pdf.

SCHNEIER, B., KELSEY, J., WHITING, D., WAGNER, D., HALL, C., AND FERGUSON, N. 1999a. On the twofish key schedule. In *Selected Areas in Cryptography '98, SAC'98*, S. Tavares and H. Meijer, Eds. LNCS, vol. 1556. Springer-Verlag, 27–42.

SCHNEIER, B., KELSEY, J., WHITING, D., WAGNER, D., HALL, C., AND FERGUSON, N. 1999b. *The Twofish Encryption Algorithm: A 128-Bit Block Cipher*. Wiley.

SCHNEIER, B. AND WHITING, D. 2001. A performance comparison of the five AES finalists. In *Proceedings of the 3rd AES Conference* (AES3).

SHIMOYAMA, T., TAKENAKA, M., AND KOSHIBA, T. 2002. Multiple linear cryptanalysis of a reduced round RC6. In *Fast Software Encryption, 9th International Workshop, FSE 2002*, J. Daemen and V. Rijmen, Eds. Vol. 2365. Springer-Verlag, 76–88.

SHIMOYAMA, T., TAKEUCHI, K., AND HAYAKAWA, J. 2000. Correlation Attack to the Block Cipher RC5 and the Simplified Variants of RC6. In *Proceedings of the 3rd AES Conference (AES3)*.

SLIJEPCEVIC, S., TSIATSIS, V., ZIMBECK, S., SRIVASTAVA, M., AND POTKONJAK, M. 2002. On communication security in wireless ad-hoc sensor networks. In *11th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*. 139–144.

SUGITA, M., KOBARA, K., AND IMAI, H. 2001. Security of reduced version of the block cipher camellia against truncated and impossible differential cryptanalysis. In *Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security*, C. Boyd, Ed. LNCS, vol. 2248. Springer-Verlag, 193–207.

SZEWCZYK, R., POLASTRE, J., MAINWARING, A., AND CULLER, D. 2004. Lessons from a sensor network expedition. In *Proceedings of the 1st European Workshop Wireless Sensor Networks (EWSN 04)*. LNCS, vol. 2920. Springer-Verlag, 307–322.

TAKENAKA, M., SHIMOYAMA, T., AND KOSHIBA, T. 2002. Theoretical Analysis of "Correlations in RC6". Cryptology ePrint Archive: Report 2002/176.

TAKENAKA, M., SHIMOYAMA, T., AND KOSHIBA, T. 2003. Theoretical analysis of χ^2 attack on RC6. In *Proceedings of the 8th Australasian Conference on Information Security and Privacy (ACISP2003)*. LNCS, vol. 2727. Springer-Verlag, 142–153.

TANAKA, H., ISHII, C., AND KANEKO, T. 2001. On the strength of KASUMI without FL functions against higher order differential attack. In *3rd International Conference on Information Security and Cryptology, ICISC 2000*. LNCS, vol. 2015. Springer-Verlag, 14–21.

TEXAS INSTRUMENTS, INC. 2001. MSP430x13x, MSP430x14x Mixed Signal Microcontroller. Datasheet.

TRI VAN LE. 2003. Novel Cyclic and Algebraic Properties of AES. Cryptology ePrint Archive: Report 2003/108.

VAN DAM, T. AND LANGENDOEN, K. 2003. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the First International Conference on Embedded Networked Sensor Systems*. ACM Press, 171–180.

VAN HOESEL, L., DULMAN, S., HAVINGA, P., AND KIP, H. 2003. Design of a low-power testbed for wireless sensor networks and verification. Tech. Rep. TR-CTIT-03-45, Centre for Telematics and Information Technology, University of Twente, The Netherlands. Sept.

WHITING, D. 1998. http://www.schneier.com/code/twofish-optimized-c.zip.

WORLEY, J., WORLEY, B., CHRISTIAN, T., AND WORLEY, C. 2001. AES Finalists on PA-RISC and IA-64: Implementations & performance. In *Proceedings of the 3rd AES Conference (AES3)*.

XUE, Q. AND GANZ, A. 2003. Runtime security composition for sensor networks (SecureSense). In *IEEE Vehicular Technology Conference (VTC Fall 2003)*.

YE, W., HEIDEMANN, J., AND ESTRIN, D. 2002. An energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the IEEE Infocom*. USC/Information Sciences Institute, IEEE, New York, NY, USA, 1567–1576.

YEOM, Y., PARK, S., AND KIM, I. 2002. On the security of CAMELLIA against the square attack. In *Fast Software Encryption, 9th International Workshop, FSE 2002*, J. Daemen and V. Rijmen, Eds. LNCS, vol. 2365. Springer-Verlag, 128–142.

YOUSSEF, A. AND TAVARES, S. 2002. On Some Algebraic Structures in the AES Round Function. Cryptology ePrint Archive: Report 2002/144.

ZHANG, P., SADLER, C. M., LYON, S. A., AND MARTONOSI, M. 2004. Hardware design experiences in ZebraNet. In *2nd International Conference on Embedded Networked Sensor Systems*. ACM Press, 227–238.

ZHU, S., SETIA, S., AND JAJODIA, S. 2003. LEAP: Efficient security mechanisms for large-scale distributed sensor networks. In *10th ACM Conference on Computer and Communications Security (CCS '03)*. ACM Press, 62–72.

Received May 2005; revised October 2005; accepted December 2005