Computer Networks 54 (2010) 2967–2978

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

The price of security in wireless sensor networks

Jongdeog Lee[a], Krasimira Kapitanova[b,*], Sang H. Son[b]

[a] Department of Electronics Engineering and Information Science, Korea Military Academy, Seoul, Republic of Korea
[b] Department of Computer Science, University of Virginia, Charlottesville, United States

ARTICLE INFO

Article history:
Received 20 October 2009
Accepted 17 May 2010
Available online 1 June 2010
Responsible Editor R. Molva

Keywords:
Energy consumption analysis
Symmetric-key encryption algorithm
Block cipher operation modes
Message authentication code algorithm
Wireless sensor network

ABSTRACT

With the increased application of wireless sensor networks (WSNs) to military, commercial, and home environments, securing the data in the network has become a critical issue. Several security mechanisms, such as TinySec, have been introduced to address the need for security in WSNs. The cost of security, however, still mostly remains an unknown variable. To provide a better understanding of this cost we have studied three aspects of WSNs security: encryption algorithms, modes of operation for block ciphers, and message authentication algorithms. We have measured and compared their memory and energy consumption on both MicaZ and TelosB sensor motes. The results of our experiments provide insight into the suitability of different security algorithms for use in WSN environments and could be used by WSN designers to construct the security architecture of their systems in a way that both satisfies the requirements of the application and reasonably uses the constrained sensor resources.

1. Introduction

As wireless sensor networks (WSNs) grow to be more popular and widely used, security becomes a very serious concern. Users do not want to reveal their data to unauthorized people as the disclosed information could be used for malicious purposes. This concern is even more relevant to wireless environments where anyone can overhear a message sent over the radio. Therefore, even a very useful and convenient system might not be appealing to the users if it is not secure. To address this problem, researchers in WSNs have implemented several security protocols to be used in sensor networks. Examples of such protocols are TinySec [1] and TinyECC [2].

Unfortunately, security is in general considered to be expensive. Its cost is even more noticeable in WSNs due to the limited resources of the sensor nodes. Thus, in order to provide a sufficient level of security while properly utilizing the available resources, it is important to have a good understanding of both the cost and the features of the security algorithms used. For example, if a sensor device does not have enough available memory to run a specific security protocol, it might be better to use an alternative algorithm which requires less memory but might also be less secure.

There is more to security in WSNs than just encryption algorithms. It is often the case that a particular encryption algorithm is not able to provide all of the desired security properties. Modes of operation have been introduced to allow the encryption of arbitrary long messages as well as to ensure that encrypting plaintext with the same key multiple times will result in the delivery of different ciphertexts. Message authentication code (MAC) algorithms are used to guarantee the authentication and the integrity of data. We believe that all three – encryption algorithms, operation modes, and MAC algorithms – are important parts of security in WSNs, and this is why we have chosen to study them thoroughly in this paper.

The data encryption algorithms used in WSNs are generally divided into three major categories: symmetric-key algorithms, asymmetric-key algorithms, and hash algorithms. A number of papers, [3–6], have investigated

* Corresponding author. Tel.: +1 469 744 0457; fax: +1 434 982 2214.
E-mail addresses: jdlee@kma.ac.kr (J. Lee), krasi@cs.virginia.edu (K. Kapitanova), son@cs.virginia.edu (S.H. Son).

1389-1286/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2010.05.011

using asymmetric-key algorithms in WSNs. However, the results they present reveal that despite the use of energy-efficient techniques, such as elliptic curve cryptography or dedicated cryptography coprocessors, asymmetric-key algorithms consume more energy than symmetric-key algorithms. Hash functions, on the other hand, are typically used for verifying the integrity of the exchanged messages and may increase the transmission cost [7,8]. Therefore, we have focused on symmetric-key algorithms, leaving the study of the other encryption algorithms for future work.

We have analyzed four widely used symmetric-key algorithms: AES, RC5, Skipjack, and XXTEA. We have chosen those particular algorithms because RC5 and Skipjack are implemented in TinySec, while AES is the current encryption standard and one of the most broadly used radios, CC2420, provides a hardware AES encryption. XXTEA is notable for its simplicity of implementation and small memory requirement, which make it a good candidate for running on the resource-constrained sensor motes.

We have studied five of the most widely used modes of operation in order to determine which ones are suitable for use in WSN applications. Four of these modes are standard while the fifth is an authenticated encryption (AE) mode. Compared to a standard mode which provides only confidentiality, an AE also provides data integrity and authenticity.

Unlike the cost of encryption algorithms and key management schemes, that of using MAC algorithms in WSNs has not been extensively studied. However, we believe that this is an important area worth researching since there are a number of applications transmitting non-secure information for which the integrity of the data is of significant importance (e.g. explosion detection applications). Therefore, we have implemented several MAC algorithms and studied their properties.

This work's contributions are threefold. First, we have analyzed AES, RC5, Skipjack, and XXTEA in terms of their energy consumption and memory requirements for MicaZ and TelosB motes. We have studied how the different algorithm parameters (e.g. key size) influence the energy consumption. Further, based on our study of AES, we have evaluated the tradeoffs between using hardware and software implemented security algorithms. The second contribution is the analysis of a number of block cipher modes of operations and how suitable they are for use in WSN applications. Although several papers, such as [9,10], have included operation modes in their experiments, few of them have focused on the modes themselves. Finally, to the best of our knowledge, this is the first work to measure the cost of using MAC algorithms on a sensor platform and compare the resource requirements of different MAC algorithms to determine which ones are appropriate for use in WSNs.

The rest of the paper is organized as follows: We discuss the related work in Section 2. The experimental setup we have used to measure the energy consumption of the different security algorithms is presented in Section 3. Section 4 describes in more detail the block cipher encryption algorithms we have studied as well as the measurements we have performed. Operation modes are presented in Section 5, followed by MAC algorithms in Section 6. Section 7 discusses our experimental results and Section 8 concludes the paper.

2. Related work

Several papers have studied the cost of using security algorithms in a WSN environment. Ganesan et al. [11] measure the energy consumption with respect to different encryption and hash algorithms so that designers can easily predict the performance of their system. They study three different encryption algorithms (RC4, IDEA, and RC5) and two message digest algorithms (MD5 and SHA1) and measure the computational overhead they cause on 6 different platforms (Amega 103, Atmega 128, M16C/10, SA-1110, PXA250, and UltraSparc2). However, they do not study MSP430, which is the MCU used in TelosB.

Guimaraes et al. [12] evaluate the energy cost of security algorithms (TEA, Skipjack, and RC5) on Mica2 using TinySec. Their results show the impact of security on the MCU and memory usage for a single mote. However, they use Mica2 motes with a CC1000 radio. In addition, according to their results, Skipjack consumes more energy than RC5, which is not what we have found.

Law et al. [9] analyze the influence of block ciphers on WSNs and compare the MCU cycles and memory used by encryption algorithms. They have studied multiple algorithms (Skipjack, RC5, RC6, MISTY1, Rijndael, Twofish, KASUMI, and Camellia). They have also measured the memory usage of the standard modes of operation (CBC, CFB, OFB, and CTR) but have not evaluated their energy consumption. In addition, they do not specify the platform they use in their experiments. Compared to their work, we provide a more detailed evaluation of both security algorithms and modes of operation.

Chang et al. [8,13] use a setup similar to ours to measure the energy consumption introduced by hash functions and symmetric-key algorithms on Mica2 motes with a CC1000 radio and Ember sensors with an EM2420 radio. They use a PicoScope 3206 oscilloscope to sample the voltage drop across two registers and, based on the measured data, speculate about the energy consumption. They study a different subset of symmetric-key algorithms. In addition, since they run their experiments on Mica2 motes, the results they provide cannot be used as a reference for MicaZ or TelosB motes.

Jinwala et al. [10] have implemented AES and XXTEA for Mica2. They have included these algorithms into TinySec and compared their performance to the default TinySec cipher, Skipjack. The experiments were performed using the Avrora simulator, which provides the CPU cycles, throughput, and energy consumption. The authors have concluded that XXTEA with an OCB mode of operation is the optimal security combination for WSNs. Compared to their work, we study a broader set of both block ciphers and modes of operation. In addition, instead of using a simulator, our experiments are performed on actual sensor platforms.

As we mentioned earlier, none of the previous work has studied the impact of the different algorithm parameters (e.g. key size) on the energy consumption of AES and RC5. In addition, although hardware implemented

AES-128 is expected to be extremely secure, its energy consumption has not been measured. Further, no previous work has evaluated the energy and memory requirements of MAC algorithms.

Most of the previous papers have used simulation for their evaluations. We, however, believe that directly measuring the current through the motes and the latency caused by the security algorithms provides more realistic data. In addition, since different papers use different sensor platforms and usually study different encryption algorithms, it is very hard to compare the results they provide. We have tried to address this issue by experimenting with a large number of encryption algorithms on two of the currently most widely used sensor platforms.

One of the first requirements for providing a security mechanism is establishing the cryptographic keys to be used by the encryption algorithms. Due to the limited resources and the need for scalability in WSNs, the key establishment protocols used in other fields are not suitable for WSN environments. To address this problem, many studies have been done to develop and evaluate specialized key establishment protocols [14–17]. Although the energy cost incurred by key establishment is an interesting problem, it is not the focus of this paper.

3. Experimental setup

The circuit we built for our experiments is shown in Fig. 1. We used a Tektronix MSO 4034 oscilloscope to measure the energy consumption in MicaZ and TelosB sensor motes. The MicaZ motes have 4 KB of RAM, 128 KB of ROM, and use a CC2420 radio. The TelosB motes have 10 KB of RAM, 48 KB ROM, and also use a CC2420 radio. The oscilloscope allowed us to measure the voltage drop in the circuit and determine the time it took for the different algorithms to execute.

To measure the energy consumption of the algorithms we connect one of the mote's pins to the oscilloscope and use it to signal when an algorithm phase begins and finishes execution. MSP430GeneralIOC.nc and HAProfiling.h interfaces are used to control the pins for MSP430 and Atmega128, respectively. The oscilloscope registers the changes of the pin's level and displays them on the screen. We believe that this method provides accurate measurements since the controlling of the pin does not significantly affect the MCU's latency or energy consumption. Further, this setup provides additional accuracy by allowing us to specifically measure the time it takes for the MCU to encrypt and decrypt the data. This is important because we are interested in the energy consumption caused by software encryption and the MCU is the only hardware involved in that process.

We calculated the power consumed by the sensor nodes using the equation $P = I \times V_b$, where I is the current in the circuit, and V_b is the voltage of the batteries. According to the MicaZ datasheet, the processor runs at 3 V so V_b = 3 V. Since $I = V/R$, to find the current we need the voltage and the resistance. With a 10.1 Ω resistor and V = 86 mV, which we determine by using the oscilloscope, the current is I = 8.52 mA. In that case the power is P = 25.54 mW. This is the power consumed by MicaZ's MCU when it is operating. By multiplying this value by the execution time, we determine the total energy consumption of each security algorithm.

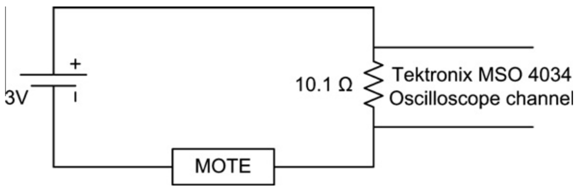


Fig. 1. Experimental setup.

Since TelosB is more power-efficient than MicaZ, its energy consumption is lower. When the MCU is active the voltage drop across the resistor in TelosB is 25.1 mV. For the current and power values we get I = 2.49 mA and P = 7.47 mW, which is less than a third of the power consumed by a MicaZ mote.

To verify that the numbers we have calculated are realistic we validate our results against the MicaZ and TelosB datasheets. According to the MicaZ datasheet the current draw when the MCU is active should be 8 mA and our measurements show 8.52 mA. Similarly for TelosB, the datasheet specifies the current in active mode to be 1.8 mA and the value we have measured is 2.49 mA. We do not consider a difference of about 0.50–0.70 mA significant since there might be slight current fluctuations in the circuit we have built, some differences due to the resistors, and, in addition, our measurement tools are not perfect.

4. Block ciphers

In this section we give a more detailed overview of the symmetric-key algorithms we have studied. All of the algorithms are blocks ciphers that have been used outside the WSNs community for many years. Block ciphers are known to be the most efficient in terms of energy consumption and latency when compared to other security algorithms. This is why they are generally believed to be the most suitable for use in WSN environments.

4.1. Studied block ciphers

4.1.1. AES

The Advanced Encryption Standard (AES) algorithm, also known as Rijndael, is a block cipher adopted as an encryption standard by the US government. AES is a substitution–permutation network and is one of the most popular symmetric-key cryptography algorithms. AES is fast in both software and hardware and is relatively easy to implement. It operates on a 4×4 array of bytes and has a fixed block size of 128 bits and a key size of 128, 192, or 256 bits with 10, 12, and 14 number of rounds, respectively.

4.1.2. RC5

An advantage of RC5 is its flexibility. Unlike other encryption algorithms, RC5 has a variable block size (32, 64, or 128 bits), number of rounds (0…255), and key size

(0...255 bits). The values of those parameters determine the level of security of the algorithm.

4.1.3. Skipjack

Skipjack is a block cipher developed by the US National Security Agency (NSA). The algorithm is an unbalanced Feistel network with 32 rounds. Skipjack uses an 80-bit key for encryption and decryption and the size of the data blocks is 64 bits. Since all of its parameters are constants, Skipjack is not as flexible as AES or RC5.

4.1.4. XXTEA

XXTEA, also called Corrected Block TEA, is the successor of XTEA and was designed to correct flaws in the original Block TEA (Tiny Encryption Algorithm). XXTEA has a small memory requirement which makes it suitable for use in resource constraint environments such as embedded systems. XXTEA has a key size of 128 bits. The length of the input plaintext is equal to the length of the output ciphertext which means that XXTEA does not require using operation modes.

4.2. Security strength

The best public cryptanalysis for AES is a related-key attack for a 256-bit key while using nine rounds. A chosen-plaintext attack can break up to eight rounds of 192-bit and 256-bit AES and seven rounds of 128-bit AES [18]. Currently AES-128, AES-192, and AES-256 (the number in the name of the algorithm corresponds to the size of the key in bits) are running on 10, 12, and 14 rounds, respectively. Thus, none of them are breakable yet.

The best known cryptanalysis for Skipjack is an attack against 31 of its 32 rounds using differential cryptanalysis [19]. Therefore, Skipjack is currently considered to be unbreakable. According to RSA Labs, however, this will soon change since 80-bit keys would be breakable by 2010 [20].

The best reported attack on XXTEA has used six rounds [21]. Therefore, XXTEA with more than six rounds can be considered secure.

Currently, brute force is the only cryptanalysis that can be applied against AES, Skipjack, and XXTEA when they are used with their correct number of rounds and key length. Because of that, we can directly compare the security strength of AES, Skipjack, and XXTEA based on the length of the key they use. The following hierarchy can be built: AES-256 > AES-192 > AES-128 = XXTEA (128-bit) > Skipjack (80-bit).

In RC5, in contrast to the above algorithms, the number of rounds does not depend on the key size. Therefore, RC5's security strength is determined by both the key size and the number of rounds. This makes RC5 hard to compare to the other block ciphers. There are two types of attacks against RC5 – differential and linear cryptanalysis. RC5 has appeared to be extremely resistant to linear attacks [22]. A differential attack described by Biryukov and Kushilevitz in 1998 remains the best published result. This attack's data requirements for RC5 with a 64-bit block size and a varying number of rounds were estimated in [22]. According to [7], using more than 16 rounds is considered to be sufficient protection against security attacks. Based on this, we have used RC5-32/18/16 to compare the security cost of RC5 and the other block ciphers, where 32 is the block size in bits, 18 is the number of rounds, and 16 is the key size in bytes.

4.3. Performance

We have performed experiments on both MicaZ and TelosB motes using the setup described above. For the complete results from our experiments refer to [23]. In addition to measuring the energy consumption of AES, RC5, Skipjack, and XXTEA we have also measured their memory usage. The results presented in Table 1 show that AES requires a significant amount of both RAM and ROM. On a MicaZ mote AES takes up about half of the available RAM. In contrast to AES, XXTEA is the lightest among all algorithms.

4.3.1. Software AES

We have measured the execution times of key setup, encryption, and decryption and calculated the corresponding energy consumption of AES-128, AES-192, and AES-256. The results for a MicaZ mote are shown in Table 2. As we can see, all three phases – key setup, encryption, and decryption – are influenced by the key size. The longer the key, the longer it takes for the phases to execute. Note that for all three algorithms – AES-128, AES-192, and AES-256, the encryption time is about 43% of the decryption time. This is a reasonable ratio since the decryption phase is more complicated and requires more computations.

The results we measured for TelosB (Table 3) were quite different – the decryption phase took more than 10 times longer than the encryption phase. We can see that running AES on TelosB takes longer than running it on MicaZ. However, since the energy consumption of TelosB is lower, both key setup and encryption require less energy to execute. This is not the case with the decryption phase, though. Despite the lower energy consumption of TelosB, decryption

Table 1
Encryption algorithm memory usage on MicaZ and TelosB sensor motes.

Encryption algorithm	MicaZ		TelosB	
	RAM (KB)	ROM (KB)	RAM (KB)	ROM (KB)
RC5	0.2	2.5	0.2	6
AES	2	10	1.8	9
Skipjack	0.6	10	0.04	7.5
XXTEA	0.049	3.1	0.04	3.8

Table 2
AES: Influence of the key size on the energy consumption of a MicaZ sensor mote.

Key size (bits)	Key setup		Encryption		Decryption	
	(ms)	(μJ)	(ms)	(μJ)	(ms)	(μJ)
128	2.44	62.32	1.53	39.08	3.52	89.90
192	2.68	68.45	1.82	46.48	4.25	108.55
256	3.01	76.88	2.11	53.89	4.98	127.19

Table 3
AES: Influence of the key size on the energy consumption on a TelosB sensor mote.

Key size (bits)	Key setup		Encryption		Decryption	
	(ms)	(μJ)	(ms)	(μJ)	(ms)	(μJ)
128	3.58	26.74	3.77	28.16	43.20	322.70
192	3.97	29.66	4.60	34.36	51.00	380.97
256	6.66	49.75	5.58	41.68	66.00	493.02

on TelosB consumes more energy than on MicaZ since it takes significantly more time to execute. As a result, the total energy cost of AES is higher for TelosB motes. This is an interesting observation since, as we shall see in the following subsections, TelosB is usually more power-efficient than MicaZ.

4.3.2. Hardware AES

Both the CC2420 and the CC2520 radios provide a hardware implementation of AES-128. When no operation modes are used the software implementation of AES can only encrypt messages that are one-block-long. Therefore, in order to provide a fair comparison between the software and hardware implementation of AES, we have used a CC2420 radio in its stand-alone mode which also encrypts only one-block-long messages. The stand-alone mode does not have distinct key setup and decryption phases. Since the key setup phase is included in the encryption phase, we have measured the time necessary to execute the combined *setup* + *encryption* phase on a CC2420 radio. The current is higher compared to that when only the MCU is active, because now we are also using the radio. It is $I = 26.14$ mA for MicaZ and $I = 21.19$ mA for TelosB. Table 4 shows the results for the hardware AES encryption for both MicaZ and TelosB. This is by far the cheapest encryption when either time or energy consumption is considered. However, a constraint is that this type of encryption is available only for CC2420 and CC2520 radios and not every platform comes equipped with them.

Table 4
Hardware AES encryption for MicaZ and TelosB sensor motes.

Architecture	Encryption (ms)	Encryption energy (μJ)
MicaZ	0.023	1.83
TelosB	0.225	14.30

Again we see that running AES on MicaZ is significantly more energy-efficient than running it on TelosB. A reasonable explanation of this phenomenon would be the different MCUs on MicaZ and TelosB. It is possible that MicaZ's Atmega128 MCU has a better hardware support for the operations that AES performs than the MSP430 MCU of TelosB.

4.3.3. RC5

Since we can vary all three parameters of RC5 – word size, number of rounds, and key size – we have used different combinations of values for these parameters to fully understand their influence on the energy consumption caused by the encryption algorithm:

- The usual word size for encryption is 32 bits (4 bytes). To study the impact of the word size on the time it takes to perform key setup, encryption, and decryption, we have also run RC5 using 16-bit words.
- The number of rounds has a proportional effect on the security of RC5 [22]. To see how the number of rounds influences the energy consumption we have executed RC5 with 4, 8, 12, 16, and 18 rounds.
- We have also used different values for the key size. We have used the same key sizes as in AES – 128, 192, and 256 bits.

With two options for the word size, five for the number of rounds, and three for the key size, we have thirty different experimental combinations of RC5 parameters. For each combination we have measured the time necessary to perform the key setup, encryption, and decryption phases and calculated the energy consumed during each of those phases. Again we have run the experiments on both MicaZ and TelosB nodes.

As expected, a longer word size leads to longer execution times for both the key setup and the encryption/decryption phases. The results for the energy consumption on MicaZ and TelosB motes for RC5 when the key size and the number of rounds were kept constant (rounds = 4, key size = 128 bits) are shown in Fig. 2.

These results are not surprising since a longer block naturally needs more time for encryption and decryption. The

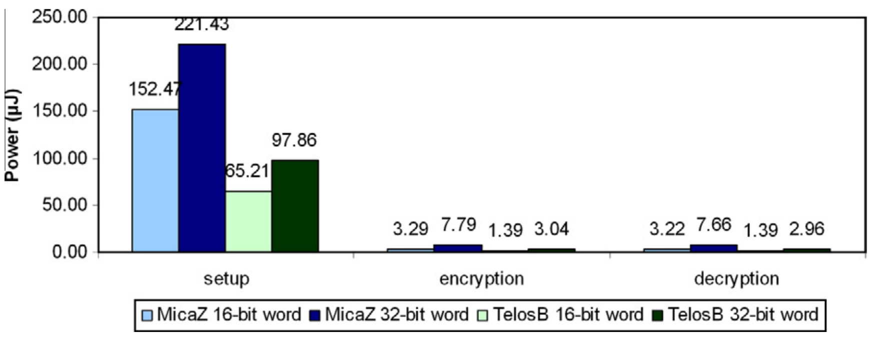


Fig. 2. RC5: Influence of the word size on the energy consumption on MicaZ and TelosB sensor motes.

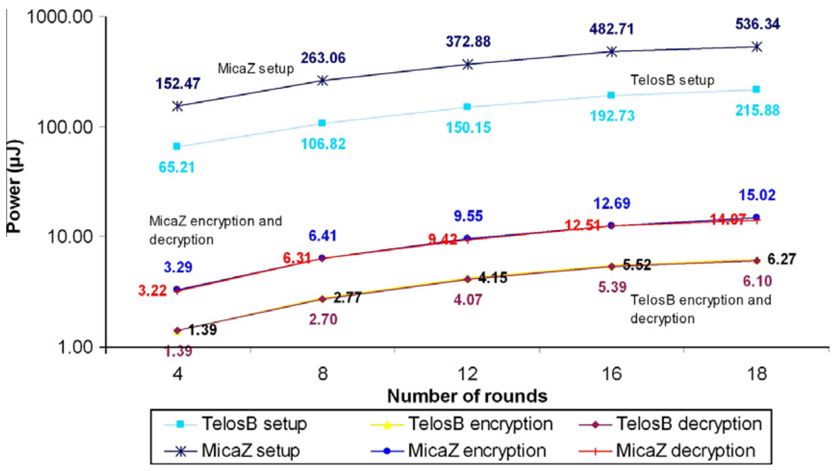


Fig. 3. RC5: Influence of the number of rounds on the energy consumption on MicaZ and TelosB sensor motes.

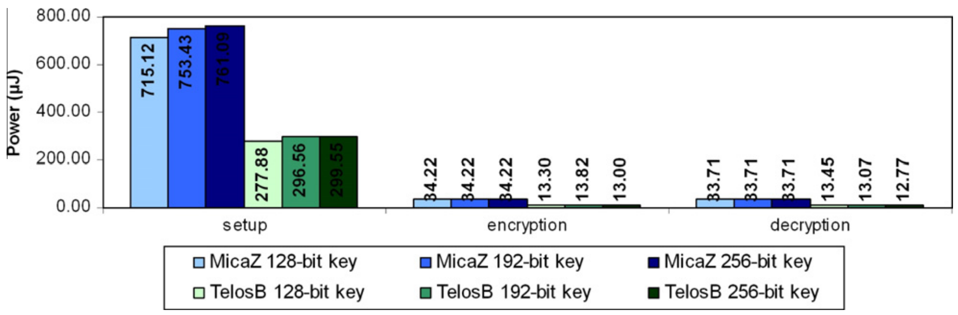


Fig. 4. RC5: Influence of the key size on the energy consumption on MicaZ and TelosB sensor motes.

Table 5
Energy consumption of Skipjack on MicaZ and TelosB sensor motes.

Platform	Encryption		Decryption	
	(ms)	(µJ)	Energy	(µJ)
MicaZ	0.22	5.52	0.22	5.52
TelosB	0.35	2.63	0.35	2.63

Table 6
Energy consumption of XXTEA on MicaZ and TelosB sensor motes.

Platform	Encryption		Decryption	
	(ms)	(µJ)	(ms)	(µJ)
MicaZ	1.94	49.55	1.86	47.50
TelosB	2.34	17.48	2.39	17.85

increased key setup time, on the other hand, can be justified by the way RC5 works. The task of the key setup phase is to expand the user's key to fill an expanded key-table S. Since the elements of that table are words, the longer the words are, the bigger the array will be and the longer its setup will take.

Changing the number of rounds also influences the execution time of both the key setup and the encryption/decryption phases. An interesting observation is that increasing the number of rounds by a constant number results in increasing the execution time of all three phases by a constant value. The values we measured when keeping the word size and the key size constant (word size = 16 bits, key size = 128 bits) on MicaZ and TelosB motes are presented in Fig. 3. Note that we have used a logarithmic scale to better visualize the changes in the algorithm's energy consumption.

Increasing the number of rounds has a steady price. For our current RC5 parameter configuration (word size = 16 bits and key size = 128 bits), increasing the number of rounds by one increases the length of the key setup phase by 1.08 ms and the encryption and decryption phases by 0.03 ms on average. This means that the energy required for one round is 27.46 mJ for key setup and 0.79 mJ for encryption and decryption. These results reveal that the energy consumption increases linearly with the number of rounds. This linear dependence is again due to RC5's nature. The key setup phase performs the key expansion, which is not only related to the word size but also to the number of rounds. The expanded key-table S consists of $t = 2 \times r + 1$ words, where r is the number of rounds. Therefore, the higher the number of rounds, the bigger the expanded key-table, which causes the key expansion to take more time.

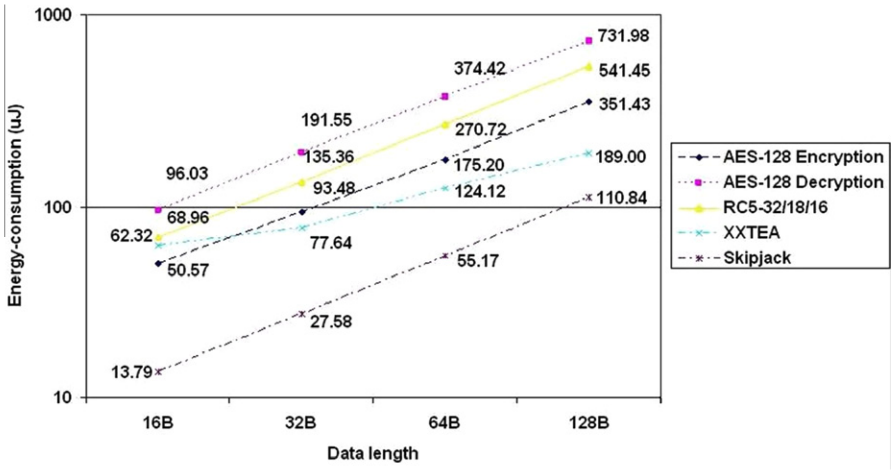


Fig. 5. Energy consumption of block ciphers on MicaZ sensor motes.

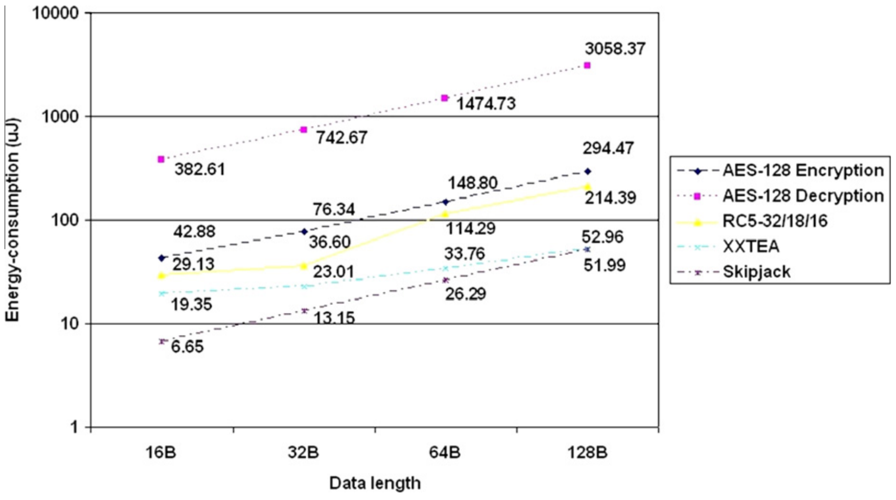


Fig. 6. Energy consumption of block ciphers on TelosB sensor motes.

Varying the key size only influences the key setup phase and not the encryption or decryption phases. The values we measured for MicaZ and TelosB when the number of rounds and the word size were kept constant (rounds = 18 and word size = 32 bits) are presented in Fig. 4. In this case the difference between the 128-bit key and the 256-bit key execution is about 2 ms. In other cases, for example when the word size is 16 bits and 12 encryption rounds are used, this difference is only 0.2 ms. These results show that even doubling the key size does not lead to considerable changes in the energy consumption.

RC5's key setup phase is influenced by the key size because the expanded key-table S is constructed using the user key. The longer the user key is, the longer it takes to copy it to the array which is subsequently used to alter the semi-random values in S.

4.3.4. Skipjack

Skipjack does not have a key setup phase because the key is determined from the very beginning. Since all the parameters of the algorithm are constants, we only had to perform measurements using a single set of parameter values per sensor platform. The result for both MicaZ and TelosB are presented in Table 5.

4.3.5. XXTEA

Just like Skipjack, XXTEA does not have a separate key setup phase. We have measured the energy consumption of XXTEA using blocks of size 2 words, which is the minimum block size allowed. Since XXTEA has a fixed key size (128 bits) and the number of rounds depends on the number of words, again only measurements with one set of values were performed per platform. The results of those measurements are provided in Table 6.

4.3.6. Comparison

Figs. 5 and 6 show the energy consumption of each of the block ciphers for 16-, 32-, 64-, and 128-byte messages. We have used a logarithmic scale in both figures to better visualize the results. The cost of key setup is not included in the results. Cipher block chaining (CBC) is used as the default mode of operation. Some of the conclusion we can draw based on our measurements are:

1. Skipjack is the most energy-efficient algorithm among the four block ciphers we have studied. However, even though XXTEA consumes slightly more energy than Skipjack, it provides stronger security.
2. RC5 consumes less energy than AES except during the encryption phase on MicaZ.
3. On a TelosB mote, the energy consumption of the decryption phase of AES is more than 10 times that of RC5.

These results, however, could change depending on the mode of operation that is used. Some modes allow decryption to be performed without using a decryption block, which can significantly speed up the execution.

Since we have not tested the in-line encryption of the CC2420 radio and the stand-alone mode does not operate on data blocks longer than 16 bytes, we refer to the CC2420 datasheet for comparison information [24]. It shows that encrypting a 98-byte message and producing a 12-byte MAC takes up only 99 μs. This means that the consumed energy is about 7.4 μJ, which is much less than that of any of the software encryptions.

5. Modes of operations

As mentioned earlier, the modes of operation are used together with block ciphers to support messages longer than the block length. The National Institute of Standards and Technology (NIST) recommends five standard modes: Electronic codebook (ECB), CBC, Cipher feedback (CFB), Output feedback (OFB), and Counter (CTR). We have investigated all of these modes except ECB since it is rarely used due to its non-secure property of always producing the same ciphertext for a given plaintext under a given key.

As opposed to the standard modes which only provide confidentiality, AE such as OCB also provides data integrity and authenticity. To achieve the same level of security as AE, standard modes are combined with message authentication code (MAC) algorithms.

5.1. Stream cipher

OFB, CFB, CTR, and OCB transform a block cipher into a *stream cipher*. This means that with these modes the length of the ciphertext is the same as the length of the plaintext. In contrast, the length of CBC's output is longer than that of the input. Suppose, for example, that the currently used cipher is AES (16-byte block) and that the message we need to encrypt is 17 bytes. The length of the output from OFB, CFB, CTR, and OCB will remain 17 bytes, but when CBC is used it will be 32 bytes. Because of those additional bytes that have to be sent over the radio, using CBC in a WSN environment is not an energy-efficient choice.

5.2. Initialization vector

The *initialization vector* (IV) is a block of bits that is fed into the first encryption (or decryption) block in order to produce different ciphertexts if some plaintext is encrypted multiple times with the same key. CBC and CFB allow the use of repeatable IV but OFB, CTR, and OCB do not. It is not trivial to guarantee the uniqueness of the IV across all messages in the network. If we are using a 4-byte counter for the IV (which is the actual size of the IV header used in TinySec), the maximum number of packets under one key is 2^{32}. If the total number of motes in the network is n, the average number of packets that each mote can send will be $2^{32} \div n$. Because an application using OFB, CTR, or OCB needs to change its encryption key more frequently than one using CBC or CFB, allowing repeatable IV is more efficient.

5.3. Performance

Figs. 7 and 8 show the energy consumption of each operation mode for messages of length 16, 32, 64, and

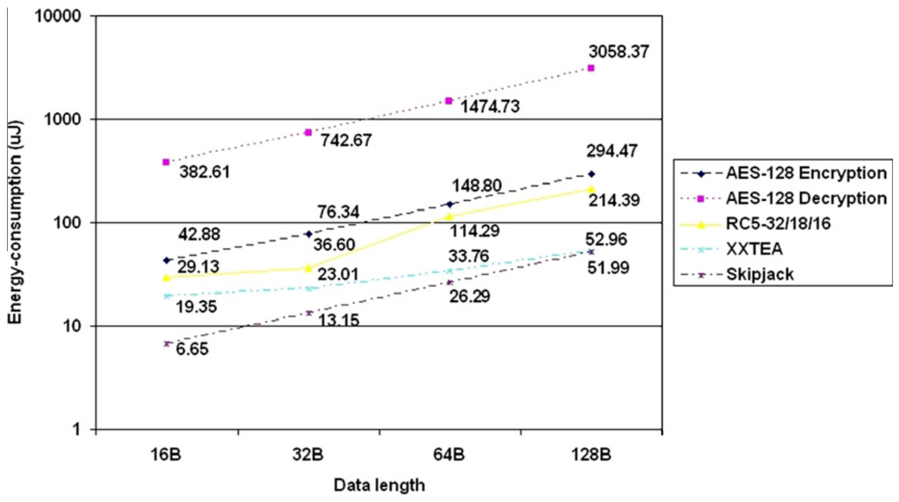


Fig. 7. Energy consumption of modes of operation based on AES-128 on a MicaZ sensor mote.

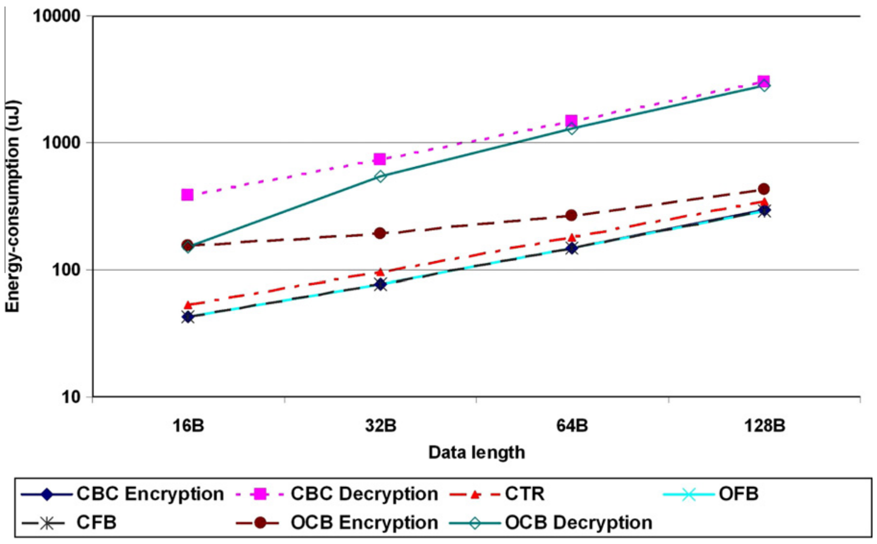


Fig. 8. Energy consumption of modes of operation based on AES-128 on a TelosB sensor mote.

128 bytes. We have used a logarithmic scale again. The results show that OFB and CFB are the most energy-efficient modes. Although OCB consumes the most energy, we can still say that it has excellent performance since, as an AE mode, it also includes the computation of a MAC. As we will see in Section 6, the cost of MAC algorithms themselves is not negligible.

Since the decryption phases for CTR, OFB, and CFB are almost identical to the encryption phases performed in reverse, the energy consumption during the two phases are similar. CBC and OCB, however, have considerable differences in the energy consumption during encryption and decryption. This difference becomes even more noticeable on a TelosB mote, where decryption takes about 10 times longer than encryption. Therefore, if an application is using TelosB motes, CTR, OFB, and CFB are more suitable since if CBC or OCB is used the decryption cost is significant.

5.4. Analysis

TinySec [1], one of the well-known link layer security architectures, uses CBC as its default mode of operation. CBC was chosen because it allows the use of a repeatable IV. However, CBC's output is longer than the input, which is not a desirable property in WSN environments. Since CFB allows for repeating the IV and also has a stream cipher property, we believe that it could be a successful replacement of CBC. In addition, CFB shows the best performance among all operation modes. Admittedly, OCB includes a MAC, which makes it an efficient operation mode, but it requires the use of unique IV.

6. Message authentication code

Using encryption without an authentication mechanism has been proven to be insecure [1]. Without authentication, the receiver cannot know who the source of the message is or if the data has been manipulated. Attackers can take advantage of this weakness and flip bits in the message, which results in predictable changes to the plaintext. These changes allow the attackers to infer information about the original message. Authentication code is used to address this vulnerability.

6.1. Studied MACs

Cipher block chaining MAC (CBC-MAC) is the customary way to construct a MAC from a block cipher. CBC-MAC carries the encryption as CBC, and takes the last result as a MAC. An important property is that CBC and CBC-MAC must use a different key since otherwise even if changes are made to the message the MAC tag could still be valid. In addition, CBC-MAC is not secure for arbitrary-length messages. Bellare et al. [25] have suggested three approaches to fix these problems. We have chosen to apply the most recommended one – encrypting the message length and XORing this with the first block of the plaintext.

Another variation of CBC-MAC is XMAC. It operates with one key and supports variable-length messages. In order to fix the above mentioned problems of CBC-MAC, Black et al. [26] derive three different keys from one key and also use a padding technique.

CMAC, recommended by NIST, also resolves the two disadvantages of CBC-MAC. In contrast to XMAC, only one key is used for the computation. Itawa et al. [27] have optimized the key length without any security loss. Notice that the key length of CBC-MAC, XMAC, and CMAC is $2k$, $k + 2n$, and k, respectively, where k is the key length of the underlying encryption algorithm and n is the block size of the cipher.

HMAC, a keyed-Hash MAC, is calculated not by using a block cipher but by using an iterative cryptographic hash function such as MD5 or SHA-1. The hash function divides the message into fixed-size blocks (e.g. 128 bits for MD5) and iterates over them with a compression function. We have used HMAC-MD5 for our experiments. Although col-

Table 7
Memory usage of MAC algorithms on MicaZ and TelosB sensor motes.

MAC algorithm	MicaZ		TelosB	
	RAM	ROM (KB)	RAM (KB)	ROM (KB)
CBC-MAC	1	6.6	1	5.9
XMAC	1	8.1	1	6.1
CMAC	1	5.8	1	5.1
HMAC	0.1	32	0.1	11

lisions in MD5 and SHA-1 have been reported, in general HMAC is not affected by them [28].

6.2. Security strength

The security strength of a MAC algorithm is related to that of the underlying encryption algorithms. No practical attacks against the combination of a secure encryption algorithm and any of the MAC algorithms above have been reported. Under this condition, the security strength of a MAC depends on its length. Conventional security protocols use 8- or 16-byte MACs. TinySec [1], however, is using a 4-byte MAC. The reasons for this choice are: (1) WSNs cannot afford sending additional 8 or 16 bytes for a MAC. Since the default packet length of TinyOS is 36 bytes, a 8- or 16-byte MAC is too long. (2) On a Mica2 platform with a CC1000 radio, it takes 20 months for an attacker to forge a 4-byte MAC [1]. Admittedly, this time is decreased to 3 months for MicaZ or TelosB because they are equipped with a CC2420 radio that has a bandwidth six times wider than that of CC1000. However, if an application changes its security key every 3 months, the probability that the attackers can take advantage of a forged MAC is extremely low. Based on this argument, we have measured the energy and memory requirements of selected 4-byte long MAC algorithms.

6.3. Performance

Table 7, Figs. 9 and 10 show the results of our experiments. We have used AES-128 to compute the block-cipher-based MACs. As we can see from the measurements, the block-cipher-based MACs have low memory usage but high energy consumption. Since AES itself takes up much memory, block-cipher-based MACs do not increase the memory usage significantly. It is the opposite for the hash-function-based MAC. Although HMAC takes up much ROM space because of the MD5 implementation, it demonstrates low energy usage, especially on the TelosB platform. As we can see from Figs. 9 and 10, compared to the block-cipher-based MACs, HMAC's energy consumption is less dependent on the message length.

7. Discussion

In this section we summarize the results from our experiments. First, we have measured the memory and energy requirements of block ciphers. Our results show that Skipjack and XXTEA have good performance in terms of both memory usage and energy consumption. On the other hand, although RC5 and AES have higher resource requirements, they can provide a higher level of security than Skipjack and XXTEA. The energy consumption of AES and RC5 depends heavily on the key size and the number of rounds, respectively. Also AES performs differently during encryption and decryption. The energy consumption of its decryption phase is more than double compared to that of its encryption phase on a MicaZ platform, and more than 10 times higher on a TelosB platform. Although RC5's encryption phase consumes more energy than that of AES on MicaZ, overall RC5 needs less memory and energy.

We have also studied modes of operation for block cipher algorithms and taken into account two factors: output length and IV. Our results show that CFB is the most efficient among all standard modes as it does not send extra bytes and permits the use of repeatable IV. Moreover, using CFB reduces the ROM requirements since no decryption block is necessary. Jinwala et al. [10] have shown that OCB is the most efficient mode of operation if the cost of using a MAC is included, which is confirmed by our results. However, as mentioned before, OCB has the disadvantage of requiring a unique IV. In addition, it takes up more

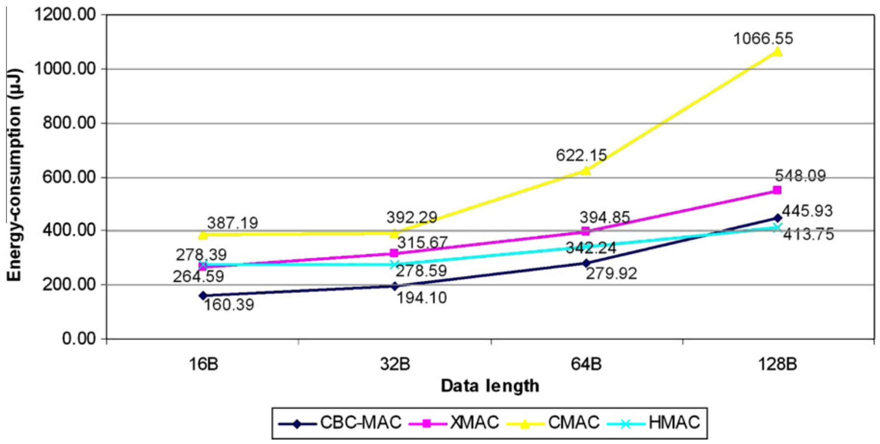


Fig. 9. Energy consumption of MAC algorithms based on AES-128 on a MicaZ sensor mote.

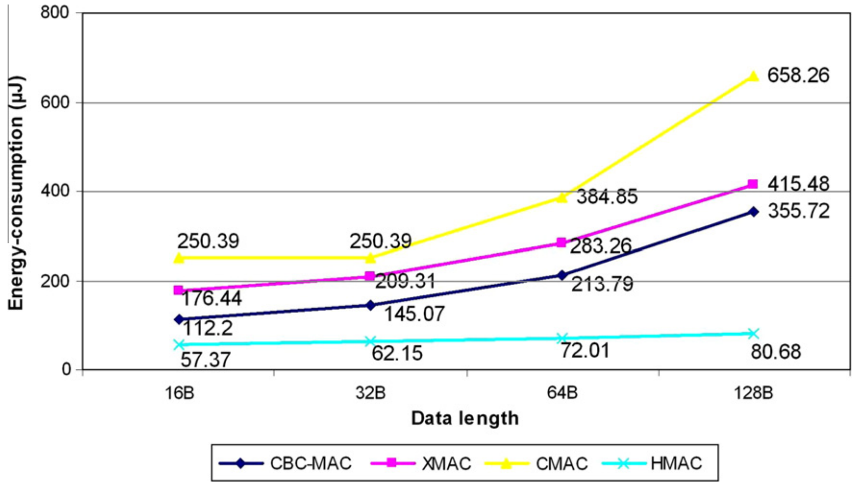


Fig. 10. Energy consumption of MAC algorithms based on AES-128 on a TelosB sensor mote.

memory that CFB and consumes 10 times more energy during decryption on a TelosB mote.

Finally, we have studied several MAC algorithms. We have shown that the memory requirements of the block-cipher-based MACs are very small compared to those of a hash-based MAC. However, the hash-based MAC has a shorter computation time and therefore consumes less energy. In addition, when using XXTEA, HMAC could be a better match than a block cipher MAC, because XXTEA cannot use different operation modes and has not yet been proven to be secure when combined with a block-cipher MAC. Admittedly, HMAC uses a significant amount of memory but this requirement could be reduced if a more memory-efficient algorithm is used.

8. Conclusion

We have analyzed the cost of using security in WSNs. In order to provide a thorough evaluation, we have measured the requirements of not only encryption algorithms but also those of using operation modes and MAC algorithms. We have extensively studied block ciphers and how changing their parameters, where applicable, affects both the energy consumption and the level of security. We have also analyzed the advantages and disadvantages of different modes of operation and MAC algorithms and tried to suggest combinations that provide a sufficient level of security while being suitable for use in a WSN environment.

For future work we plan to study more MAC algorithms such as UMAC and Poly1305-AES. We are also interested in looking into how nonce for IV could be generated without performance degradation. If this can be done, modes of operation providing both confidentiality and authentication, such as OCB, might prove to be very suitable for WSNs.

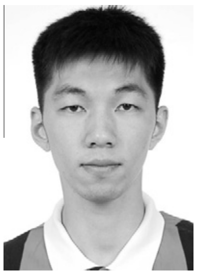
Acknowledgements

This research work was supported by KOSEF WCU Project R33-2009-000-10110-0.

References

[1] Chris Karlof, Naveen Sastry, David Wagner, TinySec: a link layer security architecture for wireless sensor networks, in: SenSys 2004, pp. 162–175.
[2] An Liu, Peng Ning, TinyECC: a configurable library for elliptic curve cryptography in wireless sensor networks, in: IPSN 2008, pp. 245–256.
[3] Guido Bertoni, Luca Breveglieri, Matteo Venturi, Power aware design of an elliptic curve coprocessor for 8 bit platforms, in: PERCOMW 2006, pp. 337.
[4] David Malan, Matt Welsh, Michael Smith, A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography, in: IEEE SECON 2004, pp. 71–80.
[5] Krzysztof Piotrowski, Peter Langendoerfer, Steffen Peter, How public key cryptography influences wireless sensor node lifetime, SASN 2006, pp. 169–176.
[6] Arvinderpal S. Wander, Nils Gura, Hans Eberle, Vipul Gupta, Sheueling Chang Shantz, Energy analysis of public-key cryptography for wireless sensor networks, in: PERCOM 2005, pp. 324–328.
[7] Nachiketh R. Potlapally, Srivaths Ravi, Anand Raghunathan, Niraj K. Jha, A study of the energy consumption characteristics of cryptographic algorithms and security protocols, in: IEEE TMC 2005, pp. 128–143.
[8] Chih-Chun Chang, Sead Muftic, David J. Nagel, Measurement of energy costs of security in wireless sensor nodes, in: ICCCN 2007, pp. 95–102.
[9] Yee Wei Law, Jeroen Doumen, Pieter Hartel, Survey and benchmark of block ciphers for wireless sensor networks, ACM Transactions on Sensor Networks (2006) 65–93.
[10] Devesh Jinwala, Dhiren Patel, Kankar Dasgupta, Optimizing the block cipher and modes of operations overhead at the link layer security framework in the wireless sensor networks, in: ICISS 2008, pp. 258–272.
[11] Prasanth Ganesan, Ramnath Venugopalan, Pushkin Peddabachagari, Alexander Dean, Frank Mueller, Mihail Sichitiu, Analyzing and modeling encryption overhead for sensor network nodes, in: WSNA 2003, pp. 151–159.
[12] G. Guimaraes, E. Souto, D. Sadok, J. Kelner, Evaluation of security mechanisms in wireless sensor networks, in: Systems Communications Proceedings 2005, pp. 428–433.
[13] Chih-Chun Chang, David J. Nagel, Sead Muftic, Balancing security and energy consumption in wireless sensor networks, Mobile Ad-Hoc and Sensor Networks (2007) 469–480.
[14] Gaurav Jolly, Mustafa C. Kusçu, Pallavi Kokate, Mohamed Younis, A low-energy key management protocol for wireless sensor networks, in: ISCC 2003, pp. 335–340.
[15] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, J.D. Tygar, SPINS: security protocols for sensor networks, Wireless Networks (2001) 189–199.
[16] Sencun Zhu, Sanjeev Setia, Sushil Jajodia, LEAP+: efficient security mechanisms for large-scale distributed sensor networks, ACM Transactions on Sensor Networks (2006) 500–528.

[17] David W. Carman, Peter S. Kruus, Brian J. Matt, Constraints and approaches for distributed sensor network security, NAI Labs, The Security Research Division, 2000, number 00-010.
[18] Niels Ferguson, Richard Schroeppel, Doug Whiting, A simple algebraic representation of Rijndael, Selected Areas in Cryptography, LNCS 2001, pp. 103–111.
[19] Eli Biham, Alex Biryukov, Adi Shamir, Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials, Journal of Cryptology 5 (2005) 291–311.
[20] RSA Laboratories, 2009. <http://www.rsa.com/rsalabs>.
[21] Tinyness: An overview of TEA and related ciphers, 2004. http://www.users.cs.york.ac.uk/matthew/TEA/.
[22] B.S. Kaliski Jr., Y.L. Yin, On the security of the RC5 encryption algorithm, RSA Laboratories, September 2006, TR-602, Version 1.0.
[23] Jongdeog Lee, Krasimira Kapitanova, Sang H. Son, Supporting multilevel security in wireless sensor networks, University of Virginia Department of Computer Sciences, October 2008, CS-2008-11.
[24] CC2420 Data Sheet, 2007. <http://focus.ti.com/lit/ds/symlink/cc2420.pdf>.
[25] Mihir Bellare, Joe Kilian, Phillip Rogaway, Survey and benchmark of block ciphers for wireless sensor networks, in: JCSS 2006, pp. 362–399.
[26] John Black, Phillip Rogaway, CBC MACs for arbitrary-length messages: the three-key constructions, Advances in Cryptology, Lecture Notes in Computer Science 2000, pp. 197–215.
[27] Tetsu Iwata, Kaoru Kurosawa, OMAC: one-key CBC MA, fast software encryption, LNCS 2887 (2003) 129–153.
[28] Bruce Schneier, 2005. <http://www.schneier.com/blog/archives/2005/02/sha1_broken.html>.



Jongdeog Lee is an instructor at the Department of Electronics and Information Science of Korea Military Academy. He received the B.S. degree in computer science from Korea Military, M.S. degree from University of Virginia. His research interests include wireless sensor networks and information security.

Krasimira Kapitanova is currently working towards a Ph.D. in Computer Science at the University of Virginia. She received her B.S. degree in Computer Science and Technologies from Technical University Sofia, Bulgaria, and an M.C.S. degree from the University of Virginia. Her research interests include formal event description in sensor networks, QoS management, and information management and security.

Sang Hyuk Son is a Professor at the Department of Computer Science of University of Virginia. He received the B.S. degree in electronics engineering from Seoul National University, M.S. degree from KAIST, and the Ph.D. in computer science from University of Maryland, College Park. He has been a Visiting Professor at KAIST and Sogang University in Korea, City University of Hong Kong, Ecole Centrale de Lille in France, and Linkoping University and University of Skovde in Sweden.
Prof. Son is on the executive board of the IEEE Technical Committee on Real-Time Systems, for which he served as the Chair during 2007–2008. He has served as an Associate Editor of IEEE Transactions on Parallel and Distributed Systems, and is currently serving as an Associate Editor for IEEE Transactions on Computers, Real-Time Systems Journal, and Journal of Computing Science and Engineering. His research interests include real-time and embedded systems, database and data services, QoS management, wireless sensor networks, and information security. He has written or co-authored over 260 papers and edited/authored four books in these areas. He has served as the Program Chair and General Chair of several real-time and sensor network conferences, including IEEE Real-Time Systems Symposium, IEEE Symposium on Object and Component-Oriented Real-Time Distributed Computing, and International Conference on Networked Sensing Systems.