Supporting Multilevel Security in Wireless Sensor Networks

Jongdeog Lee
University of Virginia
jl9eh@cs.virginia.edu

Krasimira Kapitanova
University of Virginia
krasi@cs.virginia.edu

Sang H. Son
University of Virginia
son@cs.virginia.edu

ABSTRACT

In this paper we introduce the concept of multilevel security (MLS) to the field of wireless sensor networks (WSNs). As WSNs become more widely-used in military, commercial, and home environments securing the data in the network grows to be a very important issue. There are many applications, however, which require more than just protecting the data from the outside world. Just as in everyday life not every user has access to all the data, in WSNs it is sometimes necessary to secure some of the information even from "inside men". Therefore, a multilevel system that would be able to accommodate the different sensitivity levels of the data as well as the different clearance levels and *need-to-know* of the users is needed.

To provide a better understanding of the cost introduced by security we studied three symmetric-key encryption algorithms - AES, RC5, and Skipjack. We measured their memory and power consumption on both MicaZ and TelosB sensor nodes. Using those results we have built the basis for an MLS environment to be used in WSNs. The MLS component we propose is simple, flexible, and adjustable to the WSN's requirements and constraints.

Categories and Subject Descriptors

C.2.0 [Computer-Communication Networks]: Security and protection; C.2.1 [Network Architecture and Design]: Wireless communication-performance measures.

Keywords

Multilevel security, RC5, AES, Skipjack, Measurements

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 information to unauthorized people, since the leaked data can be used for malicious purposes that could lead to serious problems. Even a very useful and convenient system might not be appealing to the users if it is not secure. However, security in general is stigmatized to be very expensive. This problem is even more serious in WSNs due to the limited resources of the sensor nodes. Nevertheless, since security's importance is increasing, researchers in WSNs have turned their attention in that direction and have implemented several security mechanisms to be used in sensor networks. Two example security mechanisms are TinySec [13] and TinyECC [15].

One of the limitations of the existing security mechanisms is that they only support single-level security. However, users may have different security clearances and the data transmitted over the radio or stored in a mote may have different sensitivity. Current mechanisms do not support such requirements. Moreover, there are applications where a multilevel security (MLS) environment is absolutely necessary to have an effective system. Users may not want to use high-cost security for trivial data such as temperature and humidity. However, highly sensitive data such as enemy coordinates needs to be protected by strong encryption. In other words, providing different algorithms that can secure the information with respect to its sensitivity would certainly reduce security cost. In addition, an MLS environment makes the system more flexible. By leveraging the security level, people can share information only with legitimate users who have valid clearance and *need-to-know*. Using MLS will allow reliable communication between different security levels. An MLS component can be applied to both military and urban sensor networks. Following are two example scenarios where using MLS would be useful - military environments and smart homes.

Assume that sensor nodes are deployed around a mili-

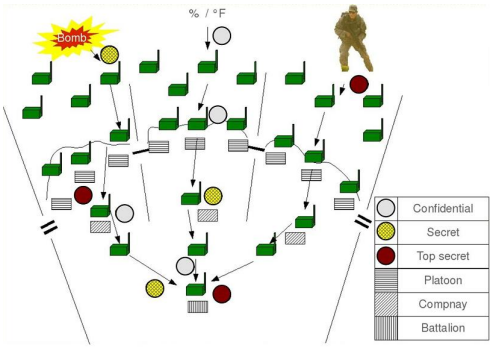

Figure 1: Military surveillance WSN

tary base for surveillance purposes as shown in Figure 1. If enemy troops appear and a sensor node correctly detects that, this data will be sent to the base station. The information needs to be protected for two reasons. First, we do not want the enemy to know that our forces have noticed its infiltration. Second, the battalion headquarters (higher level of military unit) may not want companies, platoons, and squads (lower level of military unit) to learn immediately about the invasion since this might cause disorder and chaos. A platoon leader with little experience may forget to report to a higher unit and try to handle the situation himself, which is a serious mistake and can cause the whole operation to fail. Not only is this a real story but also a frequent situation in the military. To avoid that from happening, the invasion information should be kept secret so that only authorized users can see it and give a command to the lower units after making a decision. This clear separation of information will help the higher units to have complete control over the lower units. Less sensitive information such as small explosions can be shown to companies and battalions, but not to squads or platoons. Further, non-sensitive data such as temperature and humidity data can be shared with every unit. The need-to-know principle should also be applied. Even if some battalion has sufficient security clearance, if it does not need to use some data, there is no reason why this data should be shared with it.

MLS can also be used for a smart home application. There are many sensor nodes deployed in a smart home and if someone is within radio range they can easily overhear the information sent among the nodes. The owner of the house should be allowed to read every message. However, visitors are not supposed to read messages containing information such as the owner's blood pressure or heart beat. If a doctor or a nurse visits a house to check on a patient, they only need access to the information regarding the patient's health. In addition, a doctor could be allowed to access more information than a nurse if the doctor has a higher clearance.

Because of the wireless property of WSNs (i.e. everybody might overhear a message sent over the radio) encryption is absolutely necessary for achieving security. There are many different data encryption algorithms that could be used in WSNs. They are generally divided into three major categories: symmetric-key algorithms, asymmetric-key algorithms, and hash algorithms. However, according to Potlapally et al. [18], using asymmetric-key cryptography in WSNs is not power-efficient. On the other hand, cryptographic hash functions are mainly used for message integrity verification. As a result, symmetric-key cryptography turns out to be the best approach. We have studied three widely used symmetric-key algorithms: AES, RC5, and Skipjack. We chose them since RC5 and Skipjack are implemented in TinySec, while one of the most broadly used radios, CC2420, provides a hardware AES encryption. Since power is one of the most scarce and therefore valued resources of a wireless mote, knowing how much power each of those algorithms uses will be beneficial to a large number of WSNs designers. Further, understanding how the different algorithm parameters (key size, word size, and number of rounds) influence the power consumption would allow designers to choose the best combination of parameters to meet the rest of their system's requirements such as lifetime.

This paper's contributions are twofold. First, we introduce the principles of MLS to WSNs and show its basic design. The proposed MLS component is flexible and can be changed depending on the WSN's requirements and the motes' capabilities. Different encryption algorithms can be combined to build the MLS or, if necessary, a single algorithm but with different parameters can be used instead. The second contribution is that we have extensively studied RC5, AES, and Skipjack and their power consumption and memory requirements for MicaZ and TelosB, which are two of the most popular sensor nodes at the moment. This knowledge will allow WSN's designers and users to determine which algorithms are most appropriate for their system's requirements.

The rest of the paper is organized as follows. We discuss the related work in Section 2. Section 3 elaborates on the MLS concept and the problems it attempts to solve. Section 4 describes the encryption algorithms we have chosen to study in more detail and Section 5 discusses their security strength. The experimental setup we built to measure the power consumption of RC5, AES, and Skipjack is presented in Section 6, which is followed by the results from our experiments in Section 7. Section 8 contains the MLS component we propose. Section 9 concludes the paper and discusses some of the open problems of MLS in WSNs.

2. RELATED WORK

To the best of our knowledge there has been no previous work on Multilevel security (MLS) in WSNs. We believe that our paper is the first one to introduce the MLS concept to the field of sensor networks. Security in general, however, is not a new topic in WSNs. TinySec [13], a link layer security architecture for WSNs, is provided as a library for TinyOS 1.x. It contains two block ciphers, RC5 and Skipjack, as encryption algorithms, and Cipher Block Chaining (CBC) supports encryption and decryption of messages longer than the size of a block. Message integrity and authentication are guaranteed by CBC Message Authentication Code (CBC-MAC). Unfortunately, TinySec introduces a 10 percent overhead with respect to energy, latency, and bandwidth. In addition, since TinySec needs to modify the radio stack, it does not work well on other radios except CC1000. CC1000, however, is not widely used anymore, which is why TinySec is not much used either. TinySec is not even included in TinyOS 2.x, which is the next version of TinyOS 1.x. Admittedly, the Security group in the Computer Science Department at the University of Cambridge has managed to successfully port TinySec on CC2420 [19]. Unfortunately, they could not solve the flexibility problem so nontrivial effort is still needed to use TinySec with another radio. Moreover, they support only a single level of security and therefore cannot leverage security strength at the application level. The security level can only be changed at compile time by specifying a different encryption algorithm in the make file.

Another software package, TinyECC [15], implements Elliptic Curve Cryptography (ECC) which turns out to be one of the efficient types of public-key cryptography (PKC) in the context of WSNs. Liu at al [15] provide 3 different levels of security strength by using different key sizes. However, users can specify the key length at compile time only, which means that multilevel encryption at the application level is not supported. Further, since public-key encryption algorithms consume a lot of power, they are not very suitable for WSNs. Therefore, TinyECC is mainly used for less frequent events such as key distribution in TinySA [9].

There are several papers that measure how running encryption algorithms influences the power consumption of wireless sensors. Ganesan et al. [8] measure the energy consumption with respect to different encryption algorithms so that designers can easily predict the performance of the system. They study 5 encryption algorithms and measure the computational overhead they cause on 6 different platforms. However, they are not studying Atmega128 and MSP430, which are the MCUs used in MicaZ and TelosB respectively.

Guimaraes et al. [10] evaluate the energy cost of security algorithms on Mica2 using TinySec. Their results show the impact on the MCU and memory usage for a single mote. However, the motes that they are using are Mica2 with a CC1000 radio. Further, they do not consider AES, and they fail to mention the parameters of the RC5 algorithm they study. In addition, according to their results, Skipjack consumes more power than RC5, which is not what we have found.

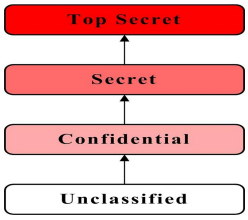

Figure 2: The hierarchy of security levels

Law et al. [14] studied the influence of block ciphers on WSNs and compared the CPU cycles and memory used by the different encryption algorithms. However, they do not specify the platform they use in their experiments. In addition, the presented power consumption results are not sufficient to provide a good reference to WSNs' designers.

Chang et al. [6, 5] used a setup similar to ours to measure the power consumption introduced by hash functions and symmetric-key algorithms on Mica2 motes with a CC1000 radio and Ember sensors with a EM2420 radio. They use a PicoScope 3206 oscilloscope to sample the voltage drop across two registers and speculate about the power consumption by observing the voltage drop. The results they provide, however, cannot be used as a reference for MicaZ and TelosB motes.

None of the previous work studies how the different algorithm parameters - word size, key size, and number of rounds, influence the power consumption. When measuring the power consumption introduced by a specific encryption algorithm, they only use a single representative of that algorithm, i.e. pick a single set of parameters, for example RC5-16/16/20. Since different papers study different algorithm representatives, the results they provide are not comparable and we cannot draw conclusions about the algorithms based on them.

3. MULTILEVEL SECURITY

3.1 Principles and problems

Multilevel security (MLS) allows users with different security clearances to reliably communicate with each other without compromising the information they share. MLS classifies messages with respect to information sensitivity, and only permits users who have a valid security

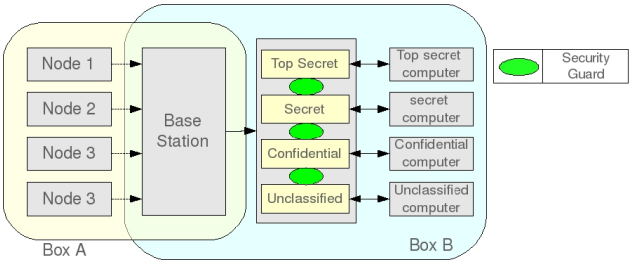

Figure 3: Multilevel security in WSN

clearance to access the data. *Security level* is a general term for either clearance level or classification level. A well-know hierarchical security level with respect to security strength is Unclassified, Confidential, Secret, and Top secret as can be seen in Figure 2. We have used this hierarchy as a basic model in the rest of the paper.

A major problem for MLS is *sanitization*, which tries to make sure that no security restrictions are violated when sharing information. There are two ways to violate those restrictions, namely *read-up* and *write-down*. Read-up occurs when lower security clearance users have access to higher security messages. Write-up occurs when higher security clearance users write high security messages to a place which can be accessed by low security level users. The Bell-LaPadula model is usually used to control the access to data [1]. However, this model has been proven to not fully satisfy the requirements of MLS [16]. Sanitization is a very hard problem to solve in both theory and practice due to macro functions and Trojan viruses. An example is when a user with a Top secret clearance accesses a Unclassified file which has been infected by a Trojan virus. The virus could then copy all the Top secret data the user can access to a Unclassified user's computer. This leak of valuable information is a clear violation of the MLS principles. Other problems, such as system assurance and downgrading, also remain unsolved in MLS.

One practical approach to implementing MLS is using multiple independent levels of security (MILS) [11]. MILS controls and physically separates information flow with respect to different security levels. Security monitors are positioned between the different security layers to make sure that invalid data flow, such as flow from a higher security layer to a lower security layer, does not occur.

3.2 Multilevel security in WSNs

Figure 3 shows a typical architecture of WSNs. Note that box B contains wired communication as opposed to box A, which represents wireless communication. MILS architecture, as mentioned in section 3.1, could be applied for box B to provide security. The situation in box A is more complicated. Any node can overhear a message if it happens to be in its radio range. This makes applying MILS to box A impossible since we cannot physically separate the wireless communication. Admittedly, we could use different radio channels for the different security levels. However, this would require a mote with a Top secret clearance, for example, to listen to all four channels for Top secret, Secret, Confidential, and Unclassified messages. Constantly switching the radio channels will require additional energy and will also significantly slow down the communication. Therefore, encryption is the best way to accomplish MLS in WSNs.

Note that we do not assume that attackers can physically access the motes. If that was the case, they can retrieve the encryption key by reading the mote's memory since the keys are stored there. In situations like that, unless the key is protected by specialized hardware, security cannot be guaranteed. Therefore, we only consider the cases when an attacker can remotely access the motes. Under this assumption, we believe that not only the transmitted data but also the one stored on the motes should be protected.

Storage: The sensed data is not always immediately transmitted. In some cases it is stored on the motes for a certain amount of time which makes it vulneravle if it is not encrypted. Therefore, data should be classified and correspondingly encrypted as soon as possible after it is sensed. In this way unauthorized users will not be able to read it even if they manage to gain access.

Transmission: The collected data is sent to the base station over the radio which makes snooping a possible attack. However, if the data is properly secured according to its security level, attackers and lower clearance users cannot read it since they do not have a valid key to decrypt it.

Another important concern is how to apply the encryption schemes to MLS. The first thing we need to consider is the strength of the security algorithms. The strongest encryption algorithms will be used to protect Top secret data while weaker algorithms will be used for Secret and Confidential information. No encryption will be used for Unclassified information. Unfortunately, the price we have to pay for security is not ignorable. It is especially noticeable in WSNs, because the motes have very limited processing, memory, and power resources. For example, a MicaZ mode, which is currently one of the most popular platforms, has a 8MHz MCU, 4KB RAM, 128KB ROM, and uses two AA batteries for power supply. Compared to a desktop computer or a laptop, this is an incredibly slow CPU, small memory, and low power supply. Therefore, encryption cost in WSNs should be carefully calculated.

The security cost in a WSN has three dimensions:

Security level	Security strength	Cost
Top secret	High	High
Secret	Medium	Medium
Confidential	Low	Low
Unclassified	None	None

Table 1: Cost of security in MLS

power consumption, latency and memory usage.

- Power consumption: Using encryption results in consuming more power which in turn decreases the lifetime of the network. Since usually most sensor nodes have limited power supply, the additional power consumption could be a serious problem.

- Latency: The extra computations the MCU has to perform due to encryption increase the execution time. Depending on the complexity of the cryptography algorithms, this latency might be significant.

- Memory: Many encryption algorithms have substantial memory needs. However, in some applications might not be able to spare half of the available memory just to provide security. Therefore, we need to make sure that using encryption is not burdening the rest of the system.

In order to efficiently use security in their systems, designers of WSNs should carefully consider the price they will have to pay for that. Table 1 presents the general correlation between security strength and cost. Since this information is general and is not specific for either the security algorithms that are used or the platform that they execute on, we did measurements that would give us more details on the actual cost of security for MicaZ and TelosB motes.

4. THE ENCRYPTION ALGORITHMS

Here we give a more detailed overview of the symmetric -key algorithms we studied in this paper. All of the algorithms are block ciphers and have been used outside the WSNs community for many years now.

4.1 AES

The Advanced Encryption Standard (AES) algorithm, also known as Rijndael, is a block cipher adopted as an encryption standard by the U.S. government. AES is one of the most popular symmetric-key cryptography algorithms. Unlike the Data Encryption Standard (DES), which was its predecessor, AES is a substitution-permutation network, not a Feistel network. AES is fast in both software and hardware and is relatively easy to implement. It 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. It operates on a 4Œ4 array of bytes, called *state*.

The cipher is specified in terms of repetitions of processing steps that are applied to make up rounds of keyed transformations between the input plaintext and the final output ciphertext. A set of reverse rounds are applied to transform the ciphertext back into the original plaintext using the same encryption key. The transformation functions used in the decryption phase are more complex than those used in the encryption phase.

4.2 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. Using above 16 rounds is believed to be sufficient protection against security attacks [18]. Algorithms with less than 8 rounds are considered to have low security and algorithms with 8 to 16 rounds provide medium security.

Since in RC5 we can change all three parameters, this allows us to study the influence each of them has on the power consumption. This could give us detailed information on how we should alter the encryption algorithm if we have to meet some non-security requirements. An example scenario is a WSN deployed in a comparatively inaccessible place, which has a lifetime requirement of a year. Since the transmitted data is confidential, we have chosen to use RC5 to encrypt the data. However, eleven months into the project, it turns out that the WSN cannot meet its lifetime requirement since the motes have used up more power that we had initially anticipated. If we still want to keep the same data transmission frequency, what we can do is decrease the security level. For that purpose we will have to change one or more of the algorithm's parameters (key size, number of rounds, and word size). What we wanted to accomplish is provide sufficient information to determine which parameter(s) to change and by how much so that we can meet the lifetime requirement of the WSN while still providing a sufficient level of security.

4.3 Skipjack

Skipjack is a block cipher that was developed by the U.S. 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. As we can see, all of Skipjack's parameters are constants.

5. STRENGTH OF THE ENCRYPTION ALGORITHMS

5.1 AES

According to [7], the most efficient key-recovery attack for Rijndael is exhaustive key search. Obtaining

information from given plaintext-ciphertext pairs about other plaintext-ciphertext pairs cannot be done more efficiently than by determining the key using brute force. The expected effort of exhaustive key search depends on the length of the encryption key and is:

- For a 16-byte key - 2^{127} applications of Rijndael.
- For a 24-byte key - 2^{191} applications of Rijndael.
- For a 32-byte key - 2^{255} applications of Rijndael.

The rationale for this is that a considerable safety margin is taken with respect to all known attacks.

5.2 RC5

There are two types of attacks against RC5 - differential and linear cryptanalysis. RC5 has appeared to be extremely resistant to linear attacks [12]. A differential attack described by A. Biryukov and E. Kushilevitz in 1998 remains the best published result. A summary of the data requirements for this attack with a varying number of rounds is provided in Table 2 for RC5 with a 64-bit block size. The second row in the table has been derived from the first row using the fact that a differential attack with `m` chosen plaintexts can be converted into one with approximately $2^{w}(2m)^{1/2}$ known plaintexts where the block size is `2w` [3]. In chosen plaintext cryptanalysis [17] it is assumed that the cryptanalyst can enter specially chosen text into the enciphering device and get a cryptogram created under the control of the secret key. In contrast, in known plaintext cryptanalysis [17] it is assumed that the cryptanalyst knows the ciphertext and a portion of the original text, and in special cases knows the correspondence between the ciphertext and the original text.

Number of rounds	4	6	8	10	12	14	16	18
Chosen plaintext	2^{7}	2^{16}	2^{28}	2^{36}	2^{44}	2^{52}	2^{61}	$>$
Known plaintext	2^{36}	2^{41}	2^{47}	2^{51}	2^{55}	2^{59}	2^{63}	$>$

Table 2: Plaintext requirements for the currently best-known Differential attack on RC5 (64-bit block size)

While most of the data requirements are impractical anyway, ">" is used to denote when the attack is infeasible even at a theoretical level. This is when the plaintext requirements are greater than $\mathbf{2^{2w}}$, which is the maximum number of possible 2w-bit plaintexts [3].

5.3 Skipjack

E. Biham and A. Shamir discovered an attack against 16 of the 32 rounds of Skipjack, and with A. Biryukov [2] extended this to 31 of the 32 rounds using impossible differential cryptanalysis. Truncated differentials and later a complementation slide attack were published against all 32 rounds of the Skipjack cipher. It was found, however, that the attacks are flawed. Biham, Shamir and Biryukov's attack continues to be the best cryptanalysis of Skipjack known to the public.

Brickell et al. [4] wrote in 1993 that Skipjack will not be broken by exhaustive search in the following 30-40 years. They also took into account the speed by which processing power increases, so we can believe that Skipjack has at least 15-25 years left before it is broken by exhaustive search. The authors also say that Skipjack cannot be attacked with a shortcut method either.

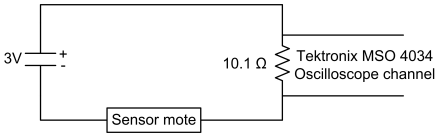

Figure 4: Experimental setup

5.4 Comparison

Since all three algorithms are so different there is no straightforward way to compare their strength. However, there are a couple of conclusions that can be drawn. The first one is that if we look at Skipjack and AES, we can claim that AES is stronger than Skipjack because the strength of both algorithms is determined by the key size. The key used in AES (128, 192 or 256 bits) is always longer than the key used in Skipjack (80 bits). The second conclusion is that we cannot make the same claim for RC5. Since the strength of RC5 is not determined by the key length but by the number of rounds, we cannot compare it to any of the other two algorithms. The fact that all three parameters - block size, key size, and number of rounds, can be arbitrarily changed makes the comparison even harder.

6. EXPERIMENTAL SETUP

The circuit we built for our experiments is shown in Figure 4. We used a Tektronix MSO 4034 oscilloscope to measure the power 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 sensors have 10KB of RAM, 48 KB ROM and also use a CC2420 radio. The oscilloscope helped us determine the time it took for the three different encryption algorithm phases - key setup, encryption and decryption, to execute. It also allowed us to measure the voltage drop in the circuit.

We use the oscilloscope to determine the power consumption of any of the three algorithm phases. We connect one of the pins of the mote to the oscilloscope and use it to signal when a phase begins and finishes ex-

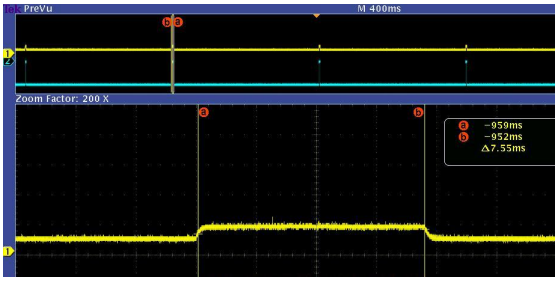

Figure 5: Example oscilloscope display

ecuting. HAProfiling.h and MSP430GeneralOC.nc interfaces are used to control the pins for Atmega128 and MSP430 respectively. The oscilloscope registers the changes in the pin's level and displays that on the screen. An example oscilloscope display is captured in Figure 5.

We calculated the power consumed by the mote using the equation P = I x V_b, where I is the current in the circuit, and V_b is the voltage of the battery. According to the MicaZ datasheet the mote's processor runs at 3V and this is the value we have set for V_b. Since $I = V/R$, to find the current we need the voltage and the resistance. We are using a 10.1Ω resistor and we can determine that the voltage is V = 86 mV using the oscilloscope. Therefore, the current is I = V/R = 86 mV/10.1Ω = 8.52 mA. Now we can calculate the power P = I x V_b= 8.52mA x 3V = 25.54 μJ. 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 power each algorithm phase consumes.

Since TelosB is more energy-efficient than MicaZ, its power consumption is lower. The voltage in TelosB is 25.1 mV when the MCU is working and 4.8 mV when the MCU is idle. We can see that when the MCU is idle the TelosB power management puts it into sleep mode and turns down the voltage. When we calculate the current we get $I_{MCU-active}$ = V/R = 25.1mV/10.1Ω = 2.49mA and $I_{MCU-idle}$= V/R = 4.8 mV/10.1Ω = 0.48 mA. Calculating the power gives us: P= $I_{MCU-active}$ x V_b= 2.49 mA x3V = 7.47 μJ, which is almost a third of the power consumed by a MicaZ mote.

To verify that the numbers we calculated were realistic we validated our results against the information in 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 1.8mA and 0.005mA in sleep mode. The values we measured are 2.49mA and 0.48mA for active and idle mode respectively. A difference of about 0.50-0.60mA for both platforms is not significant especially if we consider that there might be some slight current fluctuations in the circuit we have built, we have some differences due to the resistors we are using, and, in addition, our measurement tools are not perfect.

Since we are measuring the power consumption of software encryption, this only involves the MCU. Therefore, the MCU is the only active hardware during our experiments. The time needed for turning the MCU's pin on and off is very short and does not affect the total execution time. Using this setup we can measure the exact time each encryption phase takes with very low error.

7. MEASUREMENTS

In this section we present the results of our experiments. The experiments were performed on both MicaZ and TelosB motes using the setup described above. In addition to measuring the power consumption of AES, RC5, and Skipjack, we also measured the RAM and ROM memory usage of the algorithms. Those results are presented in Table 3.

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

Table 3: Encryption algorithm memory usage on MicaZ and TelosB

We can see that AES takes up a significant amount of both RAM and ROM. For a MicaZ mote the RAM requirements are about half of the available RAM. However, we believe that this number can be decreased if the algorithm is further memory-optimized. Chang at al [5] succeeded to decrease the RAM size by 1KB using their "Divide and Conquer" technique. In contrast to AES, RC5 and Skipjack are not so memory-heavy.

7.1 AES

7.1.1 Software implementation

The only variable parameter in AES is the key size. We have measured the execution times and calculated the power consumption of AES-128, AES-192 and AES-256 where the number in the name of the algorithm corresponds to the size of the key in bits. We measured separately the execution times for key setup, encryption, and decryption.

The values we measured on a MicaZ mote are shown in table 4. As we can see, all three phases - key setup, encryption, and decryption, are influenced by the key size. The bigger the size of the key, the longer it takes

for all three phases to execute. Something to note is that for all three algorithms - AES-128, AES-192, and AES-256, the encryption time is about 43 percent 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 were quite different. Here decryption took much longer than encryption - more than 10 times. The measurements and corresponding calculations are presented in Table 5. As we can see, TelosB is slower compared to MicaZ. However, since its power consumption is less, the final result is that both the key setup and encryption phases require less power to execute. This is not the case with the decryption phase, though. Since it takes up so much longer to execute, even though TelosB consumes less energy, the end result is that decryption on TelosB costs more power than decryption on MicaZ. As a result, the total power consumption of TelosB when executing AES is higher than that of MicaZ. This is an interesting observation since, as we shall see in the following subsections, TelosB motes are usually more power-efficient than MicaZ motes.

7.1.2 Hardware implementation

CC2420 provides a hardware implementation of AES-128. Our sensors are running in *stand-alone mode*, which means that the radio is only encrypting the messages without actually transmitting them. The hardware AES -128 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.

When executing the hardware AES-128 the current is higher compared to that when only the MCU is active, since now we are also using the radio. For MicaZ the current is $I = 26.14mA$ and for TelosB - $I = 21.19mA$. The results for the hardware AES encryption for both MicaZ and TelosB are presented in Table 6. This is by far the cheapest encryption when either time or power consumption is concerned. However, the constraint we have to face is that this type of encryption is available only for CC2420 or CC2520 radios and unfortunately not every platform comes equipped with them. Sensors using CC1100 or CC2500 are not able to take advantage of the hardware AES encryption.

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

Table 6: Hardware AES encryption for MicaZ and TelosB

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

7.2 RC5

Since we can vary all three parameters of RC5 - word size, number of rounds, and key size, we used different combinations to fully understand the influence of those parameters on the power 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 also executed RC5 using a 16-bit word.
- The number of rounds has a proportional effect on the security of RC5[1]. We wanted to see how the number of rounds influences the power consumption. For this reason we measured the power consumption with 4, 8, 12, 16, and 18 rounds.
- We used different values for the key size. We chose the values that are used in the AES algorithm - 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 measured the time necessary to perform the key setup, encryption and decryption functions. Then we calculated the power that was consumed during each of those functions as well as the total power. We ran the experiments on both MicaZ and TelosB nodes.

The execution times of all three phases of RC5 were longer when we ran RC5 on TelosB. However, the total power consumption was smaller. The explanation behind this result is that the current through a TelosB node (2.49mA) is much smaller than the current through a MicaZ node (8.52 mA). Since the execution time of RC5 on TelosB was less than twice that on MicaZ, the power consumption on TelosB was smaller. Based on this result we can make the conclusion that it is more power-efficient to run RC5 on TelosB nodes.

7.2.1 Changing the word size

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 power consumption on MicaZ and TelosB nodes for RC5 when

Key size (bits)	Key setup (ms)	Encryption (ms)	Decryption (ms)	Key setup power (µJ)	Encryption power (µJ)	Decryption power (µJ)	Total power (µJ)
128	2.44	1.53	3.52	62.32	39.08	89.90	191.29
192	2.68	1.82	4.25	68.45	46.48	108.55	223.48
256	3.01	2.11	4.98	76.88	53.89	127.19	257.95

Table 4: AES: Influence of the key size on the execution time on MicaZ

Key size (bits)	Key setup (ms)	Encryption (ms)	Decryption (ms)	Key setup power (µJ)	Encryption power (µJ)	Decryption power (µJ)	Total power (µJ)
128	3.58	3.77	43.20	26.74	28.16	322.70	377.61
192	3.97	4.60	51.00	29.66	34.36	380.97	444.99
256	6.66	5.58	66.00	49.75	41.68	493.02	584.45

Table 5: AES: Influence of the key size on the execution time on TelosB

the key size and the number of rounds were kept constant (rounds = 4, key size = 128 bits) are presented in Figure 6.

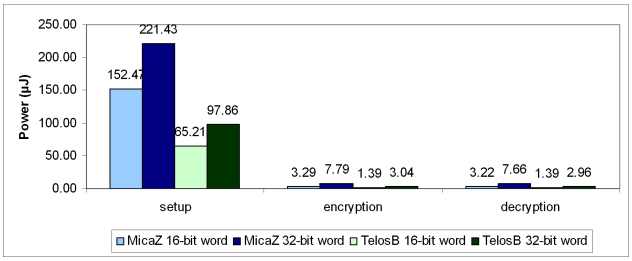

Figure 6: RC5: Influence of the word size on the power consumption on MicaZ and TelosB sensor nodes.

As we can see from the values in Figure 6, increasing the word size from 16 to 32 bits more than doubles the execution time for the encryption and decryption functions on both architectures. This ratio remains constant even if we change the key size or the number of rounds. Since the key setup is also longer, when using 32-bit words we can conclude that a smaller word size leads to faster execution for all three cryptographic phases. If we divide a single 32-word in half we will get two 16-bit words. The key setup will occur only once, so summing up the time for key setup, encryption and decryption for the 32-bit word will give us a larger number than the sum of the key setup and twice the encryption and decryption of the 16-bit words. This means that decreasing the word size decreases the power consumed in RC5. The same conclusions should also apply for the case when 64-bit words are used. This result is not surprising since a longer block naturally needs more time for encryption and decryption. The increase in the 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 array `S`. Since the elements of that array `S` are words, the longer the words are, the bigger the array will be and the longer its setup will take.

7.2.2 *Changing the number of rounds*

Changing the number of rounds influences the execution times 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 Figure 7. Note that we have used a logarithmic scale for the y-axis in order to better visualize the changes in the algorithms' power consumption.

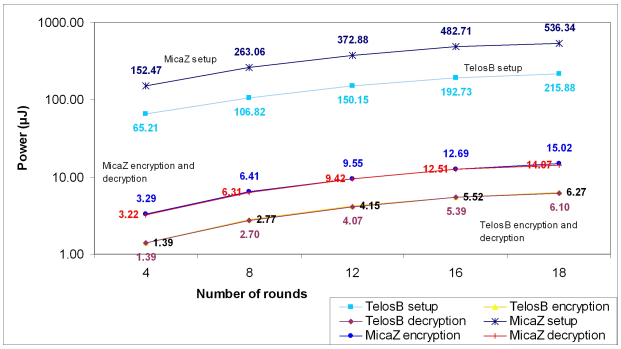

Figure 7: RC5: Influence of the number of rounds on the power consumption on MicaZ and TelosB sensor node.

We can see that 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 power required for one round would be 27.46mW for key setup and 0.79mW for encryption and decryption. From these results we can conclude that the power consumption increases linearly with the number of rounds. The more encryption rounds we perform, the more power will be used by the MCU to do the work.

The reason why the key setup phase is influenced by changing the number of rounds comes again from RC5's nature. As we mentioned previously in Section 7.2.1, 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[0...t-1] consists of $t = 2 \times r + 1$ words. Therefore, the higher the number of rounds is, the bigger the expanded key table would be, making the key expansion take more time.

7.2.3 Changing the key size

Varying the key size only influences the key setup phase. The time used for encryption and decryption remains constant. The values we measured for both MicaZ and TelosB when 18 rounds were used with a word size of 32 bits are presented in Figure 8.

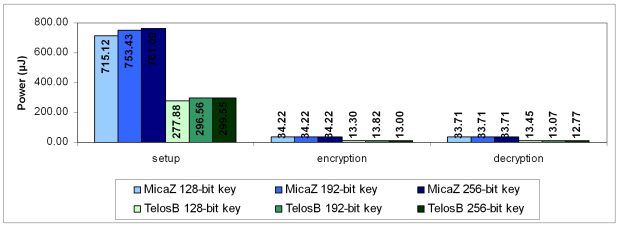

Figure 8: RC5: Influence of the key size on the power consumption on MicaZ and TelosB sensor node.

The values in those tables reveal a case when the difference between the 128-bit key and the 256-bit key execution is about 2ms. In other cases, for example, when the word size is 16 bits and we are using 12 encryption rounds, this difference is only 0.2 ms (for more information check Appendix A). From those results we can see that even doubling the key size does not lead to considerable additional power consumption changes. Therefore we can conclude that the key length has almost no impact on the execution time of any of the three algorithm phases and therefore, on the power consumption in general.

7.3 Skipjack

All parameters in Skipjack are constants so we did not have much to measure. Further, since the key is determined from the very beginning, there is no key setup phase. The result for both MicaZ and TelosB are presented in Table 7.

Platform	encrypt (ms)	decrypt (ms)	encrypt power (µJ)	decrypt power (µJ)	total power (µJ)
MicaZ	0.22	0.22	5.52	5.52	11.04
TelosB	0.35	0.35	2.63	2.63	5.26

Table 7: Power consumption of Skipjack

8. MLS DESIGN

Based on the results in Section 7 we propose a multi-level security (MLS) component which can use different cryptography algorithms depending on the information sensitivity and the available power resources. This component is flexible and allows users to change it to better comply with their system's constraints. For example, the security hierarchy can have a different number of levels than what we have shown and users can choose suitable algorithms for each level to meet the WSN's requirements. Different encryption algorithms can also be included. Adding new algorithms should not be hard since we are working with a software implementation. As long as the algorithms use the same interface, incorporating them into the MLS component should be straightforward.

Different WSNs systems have different requirements. Therefore, when building a system, designers have to comply with different constraints. As we have mentioned in previous sections, security introduces power consumption cost, latency cost, and memory usage cost. Power consumption and latency are linearly dependent since the more time the algorithm needs to execute, the more power will be consumed by the MCU. The fact that power consumption is proportional to latency follows from the equation $W = P \times T = I \times V \times T$. P is fixed because the current and voltage are also fixed, so W depends only on T. We have looked at two different scenarios. The first one is when the designer's main concern is power consumption. The second one is when memory usage is the main concern.

8.1 Based on power consumption and latency

From the measurements we have performed we can see that the power consumed by the setup phase of RC5 is significantly higher than that of AES. However, the encryption and decryption phases of RC5 are executed faster and therefore require less power. Thus, we can conclude that the algorithm choice depends on how frequently the key setup phase will need to be executed. If an application requires the key to be changed very frequently, RC5 would be less power-efficient than AES

even if RC5 outperforms AES for the encryption and decryption phases.

The overall security level also needs to be considered when designing the system. In a military application, for example, even the confidential data needs to be highly secure since the enemy or unauthorized users can benefit from inferring information from a confidential message. A smart-home application, however, does not need to use strong algorithms to protect confidential messages such as the number of times the residents of the house go to the kitchen. Although users might not want to broadcast freely this information, a low-strength algorithm will be sufficient to provide the necessary security. Admittedly, this information can be secured much better but this will require more power and will decrease the performance of the whole system. This trade-off between cost and performance is a common issue in any system design.

Table 8 suggests the MLS component for both a military and a smart-home application. Note that if we can use sensors with CC2420 or CC2520 radios, the hardware AES-128 encryption will be used at the Confidential level since it is much more cost-effective compared to the software AES-128 encryption.

Application	Confidential	Secret	Top secret
Military application	AES-128	AES-192	AES-256
Military application using a CC2420	Hardware AES-128	AES-192	AES-256
Smart-home	Skipjack	RC5 8 16	RC5 18

Table 8: Multilevel security for WSNs

8.2 Based on memory usage

If the sensor nodes have enough memory and the code size is not a concern, then Table 8 presents a suitable MLS component. However, for certain motes like MicaZ, the available memory is too small to contain all three encryption algorithms. Even if we optimize the algorithms in terms of memory and manage to store them on the mote, the remaining space will probably not be enough for the main application to run. Therefore, when there is not enough memory, it is desirable to use only one algorithm to provide all three levels of security by changing its parameters. Since AES requires significantly more memory than RC5, when the system requires the algorithms to occupy as little memory as possible, RC5 is a more appropriate choice. On the other hand, if the mote has enough memory for two algorithms after loading the main application, then users can use two different algorithms and achieve higher performance and security.

9. CONCLUSIONS AND FUTURE WORK

This paper introduces the concept of MLS to the field of WSNs. We believe that with the broader use of WSNs, securing the data in the network will become more and more important. And just like in the real world where different people have access to different data according to their clearance and need-to-know, the same principles will apply for WSNs. There is no reason why all information on the network should be available to everyone. The only scenario where this would be reasonable is when there is just a single event being detected or there is only one person or a group of people with the same clearance interested in the data. In the more common case where the nodes are sensing for a number of events and the data is of interest to more than just one person, properly setting up the data access permissions will be crucial for the system's security. This is where applications such as military surveillance, smart homes, WSNs used for security, industrial networks, etc. could largely benefit from MLS. By correctly separating the information depending on its sensitivity and assigning clearance to nodes according to their need-to-know and level of trust, MLS will help improve the security of the network in general, protect the data that is being transmitted or stored on the motes, and thus prevent damage that could occur if information is leaked.

The MLS component we propose is general and does not confine the system designers in any way. They can decide how many levels of security their system will require, which encryption algorithms should be used and how the different clearances will be distributed. In addition, we also provide data on how much each algorithm costs not only in terms of memory consumption but also in terms of latency and power consumption. This information could be used to better and more precisely design a sensor network so that it can meet its requirements.

There are a number of cryptography issues that need to be considered to support MLS in WSNs. The first one is that in order to be a full-fledged software security application, an MLS component should also contain various operation modes as well as message authentication code (MAC). Operation modes are used to more securely encrypt messages longer than a single block. There are many existing types of operation modes such as Cipher-Block Chaining (CBC), Cipher Feedback (CFB), and Counter (CTR). MAC algorithms are used to guarantee message integrity and authentication. We reserve the support of those features for future work.

The second issue is key distribution. Although symmetric -key algorithms are both powerful and efficient, key distribution is a non-trivial problem. As multiple keys are used in a MLS environment, there should be a

way to distribute those keys. One option would be using TinyECC for key distribution. However, TinyECC would introduce additional memory usage.

Another issue in MLS is packet labeling. When it is transmitted, data should be labeled according to its security level. However, labeled data will leak some information such as "*node A is transmitting Top secret messages to node B*". Such information could allow attackers to speculate about the relationship between the two nodes. On the other hand, unlabeled packets will cause performance degradation since once a node receives a message it will attempt to decipher that message using all the keys it possesses. Encrypting the header of a packet is another possible way to increase security. An encrypted header will hide both the sender and the intended receiver of the message. However, if the current node used for routing does not have the required clearance it will not know where to forward the message.

Still another problem we have to consider is dynamically changing the security level. There are cases when we would want to *downgrade* a Top secret message to Confidential or to upgrade a Confidential node to Secret. For example, in a case such as our military surveillance example, we have battalion commanders with Top secret clearance making decisions based on Top secret data. Once the decision is made, it needs to be delivered to the lower level military units. However, since Top secret data has been used, the decision also has a Top secret level. If it is sent in this form, the lower level military units will not be able to read it. Therefore, the decision needs to be downgraded. This, however, is a very hard problem since downgrading may lead to leaking information when changing the security level. Moreover, we cannot afford to reset every key to redesignate the security level for every mote. Without the ability to downgrade, though, we cannot provide a dynamically changing MLS since we would only be allowed to increase the security level but not to decrease it.

10. REFERENCES

[1] D. Bell and L. LaPadula. Secure computer system unified exposition and multics interpretation. Technical Report MTR-2997, MITRE Corp., Bedford, MA, July 1975.

[2] E. Biham, A. Biryukov, and A. Shamir. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. *Journal of Cryptology*, 18(4):291–311, September 2005.

[3] E. Biham and A. Shamir. *Differential cryptanalysis of the data encryption standard*. Springer-Verlag London, UK, 1993.

[4] E. F. Brickell, D. E. Denning, S. T. Kent, D. P. Maher, and W. Tuchman. SKIPJACK review: interim report. *Building in big brother: the cryptographic policy debate*, pages 119–130, 1995.

[5] C.-C. Chang, S. Muftic, and D. J. Nagel. Measurement of energy costs of security in wireless sensor nodes. *Proceedings of 16th International Conference on Computer Communications and Networks, 2007. ICCCN 2007.*, pages 95–102, September 2007.

[6] C.-C. Chang, D. J. Nagel, and S. Muftic. Balancing security and energy consumption in wireless sensor networks. *Mobile Ad-Hoc and Sensor Networks*, 4864/2007:469–480, November 2007.

[7] J. Daemen and V. Rijmen. The block cipher rijndael. In *CARDIS '98: Proceedings of the The International Conference on Smart Card Research and Applications*, pages 277–284, London, UK, 2000. Springer-Verlag.

[8] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. Dean, F. Mueller, and M. Sichitiu. Analyzing and modeling encryption overhead for sensor network nodes. In *WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 151–159, New York, NY, USA, 2003. ACM.

[9] J. Grossschädl. TinySA: a security architecture for wireless sensor networks. In *CoNEXT '06: Proceedings of the 2006 ACM CoNEXT conference*, pages 1–2, New York, NY, USA, 2006. ACM.

[10] G. Guimaraes, E. Souto, D. Sadok, and J. Kelner. Evaluation of security mechanisms in wireless sensor networks. *Systems Communications, 2005. Proceedings*, pages 428–433, August 2005.

[11] W. S. Harrison, N. Hanebutte, P. Oman, and J. Alves-Foss. The MILS architecture for a secure global information grid. *CrossTalk 18*, 10:20–24, October 2005.

[12] B. S. K. Jr. and Y. L. Yin. On the security of the RC5 encryption algorithm. Technical Report TR-602, Version 1.0, RSA Laboratories, September 2006.

[13] C. Karlof, N. Sastry, and D. Wagner. TinySec: a link layer security architecture for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 162–175, New York, NY, USA, 2004. ACM.

[14] Y. W. Law, J. Doumen, and P. Hartel. Survey and benchmark of block ciphers for wireless sensor networks. *ACM Transactions on Sensor Networks*, 2(1):65–93, 2006.

[15] A. Liu and P. Ning. TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. *International Conference on Information Processing in Sensor Networks*, 0:245–256, 2008.

[16] J. McLean. A comment on the "basic security theorem" of bell and lapadula. *Inf. Process. Lett.*, 20(2):67–70, 1985.

[17] N. Moldovyan and A. Moldovyan. *Innovative Cryptography, Second Edition*. Charles River Media, 2007.

[18] N. R. Potlapally, S. Ravi, A. Raghunathan, and N. K. Jha. A study of the energy consumption characteristics of cryptographic algorithms and security protocols. *IEEE Transactions on Mobile Computing*, 5(2):128– 143, December 2005.

[19] Security Group at University of Cambridge. http://www.cl.cam.ac.uk/research/security/sensornets/tinysec/.

11. APPENDIX A

Word size (bit)	Number of rounds	Key size (bit)	Setup (ms)	Encryption (ms)	Decryption (ms)	Setup power (µJ)	Encryption power (µJ)	Decryption power (µJ)	Total power (µJ)
16	4	128	5.97	0.129	0.126	152.47	3.29	3.22	158.99
32	4	128	8.67	0.305	0.300	221.43	7.79	7.66	236.88
16	4	192	6.22	0.129	0.126	158.86	3.29	3.22	165.37
32	4	192	9.36	0.305	0.300	239.05	7.79	7.66	254.51
16	4	256	6.37	0.129	0.126	162.69	3.29	3.22	169.20
32	4	256	9.98	0.306	0.301	254.89	7.82	7.69	270.39
16	8	128	10.30	0.251	0.247	263.06	6.41	6.31	275.78
32	8	128	14.40	0.600	0.592	367.78	15.32	15.12	398.22
16	8	192	10.50	0.251	0.247	268.17	6.41	6.31	280.89
32	8	192	15.10	0.600	0.592	385.65	15.32	15.12	416.10
16	8	256	10.60	0.251	0.247	270.72	6.41	6.31	283.44
32	8	256	15.70	0.600	0.592	400.98	15.32	15.12	431.42
16	12	128	14.60	0.374	0.369	372.88	9.55	9.42	391.86
32	12	128	20.20	0.894	0.884	515.91	22.83	22.58	561.32
16	12	192	14.80	0.374	0.369	377.99	9.55	9.42	396.97
32	12	192	20.90	0.894	0.884	533.79	22.83	22.58	579.20
16	12	256	14.80	0.374	0.368	377.99	9.55	9.40	396.94
32	12	256	21.30	0.895	0.884	544.00	22.86	22.58	589.44
16	16	128	18.90	0.497	0.490	482.71	12.69	12.51	507.91
32	16	128	25.90	1.190	1.180	661.49	30.39	30.14	722.02
16	16	192	29.20	0.497	0.490	745.77	12.69	12.51	770.98
32	16	192	26.90	1.190	1.180	687.03	30.39	30.14	747.56
16	16	256	19.00	0.497	0.490	485.26	12.69	12.51	510.47
32	16	256	27.00	1.190	1.180	689.58	30.39	30.14	750.11
16	18	128	21.00	0.588	0.551	536.34	15.02	14.07	565.43
32	18	128	28.00	1.340	1.320	715.12	34.22	33.71	783.06
16	18	192	21.30	0.588	0.551	544.00	15.02	14.07	573.09
32	18	192	29.50	1.340	1.320	753.43	34.22	33.71	821.37
16	18	256	21.10	0.558	0.551	538.89	14.25	14.07	567.22
32	18	256	29.80	1.340	1.320	761.09	34.22	33.71	829.03

Table 9: RC5 power consumption on MicaZ

Word size (bit)	Number of rounds	Key size (bit)	Setup (ms)	Encryption (ms)	Decryption (ms)	Setup power (µJ)	Encryption power (µJ)	Decryption power (µJ)	Total power (µJ)
16	4	128	8.73	0.186	0.186	65.21	1.39	1.39	67.99
32	4	128	13.10	0.407	0.396	97.86	3.04	2.96	103.86
16	4	192	9.54	0.187	0.188	71.26	1.40	1.40	74.07
32	4	192	14.70	0.403	0.418	109.81	3.01	3.12	115.94
16	4	256	10.40	0.192	0.190	77.69	1.43	1.42	80.54
32	4	256	17.10	0.403	0.378	127.74	3.01	2.82	133.57
16	8	128	14.30	0.371	0.362	106.82	2.77	2.70	112.30
32	8	128	18.30	0.731	0.718	136.70	5.46	5.36	147.53
16	8	192	15.50	0.384	0.369	115.79	2.87	2.76	121.41
32	8	192	20.90	0.764	0.724	156.12	5.71	5.41	167.24
16	8	256	16.10	0.373	0.375	120.27	2.79	2.80	125.85
32	8	256	24.50	0.798	0.790	183.02	5.96	5.90	194.88
16	12	128	20.10	0.556	0.545	150.15	4.15	4.07	158.37
32	12	128	26.40	1.170	1.170	197.21	8.74	8.74	214.69
16	12	192	21.00	0.569	0.540	156.87	4.25	4.03	165.15
32	12	192	28.90	1.190	1.180	215.88	8.89	8.81	233.59
16	12	256	22.00	0.589	0.551	164.34	4.40	4.12	172.86
32	12	256	31.40	1.210	1.180	234.56	9.04	8.81	252.41
16	16	128	25.80	0.739	0.722	192.73	5.52	5.39	203.64
32	16	128	33.00	1.570	1.540	246.51	11.73	11.50	269.74
16	16	192	26.90	0.750	0.732	200.94	5.60	5.47	212.01
32	16	192	35.90	1.610	1.560	268.17	12.03	11.65	291.85
16	16	256	27.10	0.733	0.722	202.44	5.48	5.39	213.31
32	16	256	37.60	1.570	1.550	280.87	11.73	11.58	304.18
16	18	128	28.90	0.839	0.817	215.88	6.27	6.10	228.25
32	18	128	37.20	1.780	1.800	277.88	13.30	13.45	304.63
16	18	192	29.20	0.826	0.809	218.12	6.17	6.04	230.34
32	18	192	39.70	1.850	1.750	296.56	13.82	13.07	323.45
16	18	256	30.30	0.823	0.834	226.34	6.15	6.23	238.72
32	18	256	40.10	1.740	1.710	299.55	13.00	12.77	325.32

Table 10: RC5 power consumption on TelosB