Practical Evaluation of a Secure Key-Distribution and Storage Scheme for Wireless Sensor Networks Using TinyOS

Juliano F. Kazienko, Igor G. Ribeiro, Igor M. Moraes, and Célio V. N. Albuquerque

Instituto de Computação – Universidade Federal Fluminense (UFF)
Rua Passo da Pátria, 156 – 24.210-240 – Niterói – RJ – Brasil
{kazienko,igor,celio}@ic.uff.br, igorcompuff@bcc.ic.uff.br

Abstract

TinyOS is a major platform broadly used to carry out experiments related to Wireless Sensor Networks (WSNs). A number of researchers claim that cryptographic mechanisms demand plenty of resources from sensors. In this context, an important issue is to develop lightweight encryption mechanisms capable of running in resource-constrained sensors. The main contribution of this work is to carry out an experimental evaluation of a secure key distribution and storage scheme in a WSN using simulation and practical experiments. Through simulation, we verify that this scheme introduces very low processing overhead, in the order of nanoseconds, when compared to existing approaches. Additionally, practical measurements indicate that the scheme can be deployed by off-the-shelf sensor platforms, such as MicaZ and TelosB. The performance metrics considered are the processing time of encryption and decryption functions, the application memory requirements and the power consumption. We have also evaluated several functionalities of the scheme on a real testbed. In summary, this work demonstrates the practical feasibility of implementing such scheme in real sensor networks.

Keywords: Wireless Sensor Networks; Security; Key Storage; Key Distribution; TOSSIM; TinyOS.

1. INTRODUCTION

Wireless Sensor Networks (WSNs) demand precise mechanisms to evaluate the feasibility of proposed solutions. WSNs are a particular case of wireless ad hoc networks. They are composed of sensor nodes, also known as motes, typically with constrained resources that use multihop communication to reach powerful sink nodes. These networks have been applied to wildlife monitoring, detection of nuclear material, underwater communication as well as in many other applications [1,2,3,6,12].

The TinyOS platform has been broadly used by the research community to evaluate WSN experiments. An interesting feature of this platform is that it enables the development of an application-specific operating system that runs on sensors as well as on a simulated environment [10,16,17].

Among challenging topics in WSNs, security certainly drives many issues. However, it is frequently claimed that standard security mechanisms are prohibitive in WSNs because they implicate in the extensive use of scarce resources, such as processing power, battery capacity, limited memory, and low bandwidth. Moreover, sensors typically operate in unattended way. Therefore, it becomes very important to protect sensitive data stored in sensors, as well as cryptographic keys [2,11]. Thus, it is important to propose and evaluate security mechanisms for sensor networks. Besides, we must show that such proposals are feasible to run in such resource-constrained platform. In order to accomplish these goals, implementation and practical experiments are required.

The main contribution of this work is to carry out an experimental evaluation of the secure key-distribution and storage scheme for WSNs previously proposed by Kazienko and Albuquerque [14,15]. In order to demonstrate that their scheme provides a lightweight key encryption mechanism, we perform simulation and practical experiments. Simulations are carried out in order to verify the processing overhead of their scheme when compared to the proposal of Oliveira and Barros [20] and to a system without cryptography. Additionally, we carry out practical experiments with the off-the-shelf sensors platforms TelosB and MicaZ, considering performance metrics such as

the processing time of encryption and decryption functions, application memory requirements, and power consumption. We have also evaluated several functionalities of this scheme. Hence, this work demonstrates the practical feasibility of implementing their scheme in real sensor networks.

The remainder of this work is organized as follows. Section 2 addresses the related works. Section 3 briefly describes the scheme proposed by Kazienko and Albuquerque. In Section 4, the methodology is presented. Section 5 describes the experiments carried out. Section 6 shows the simulation results and the comparison to related work. Section 7 presents practical experiments accomplished using TelosB and MicaZ motes. Finally, in Section 8, the conclusion and future works are presented.

2. RELATED WORKS

It is possible to find a large number of symmetric key distribution mechanisms in the scientific literature. In a broad sense, there are three well-known strategies to perform this task, using: Public-key cryptosystems, Key Distribution Center (KDC) and Key predistribution [22].

The first one invariably uses asymmetric keys in order to distribute symmetrical secret keys. The main disadvantage of such strategy is related to the energy consumption and to the memory usage since—in sensors—those resources are scarce. However, it is possible to find works that consider their use to be feasible in WSNs [21]. The second one concerns the well-known Key Distribution Center, in which a central node shares secret keys with all network nodes. Symmetric keys are protected by secret keys during distribution. Although efficient, KDC presents a single point of failure. The third class of strategies is key predistribution. In this paradigm, keys are distributed among sensors before the network becomes operational. In most predistribution schemes, a group of keys is generated, in which a portion of that group is loaded in each sensor. After this phase, sensors use their keys to establish secure channels with each other, as presented by Eschenauer and Gligor [8]. In such work, the authors propose a key-management scheme for distributed sensor networks and recognize the need for secure key storage in sensors. However, they do not propose a mechanism to solve such problem.

Martina *et al.* [19] propose the use of Hardware Security Module (HSM) to protect private keys against attacks related to logical and physical tampering with or even related to the extraction of sensitive information from the protected area. Even though HSM has been applied to a Public Key Infrastructure in the aforementioned work, its use is particularly applicable in devices that use symmetrical cryptography with the purpose of establishing a safe communication. Such module can concentrate critical functions and data, such as cryptographic keys, for instance.

Recently, several works have highlighted the benefits of increasing throughput and reliability reducing the number of transmissions in computer networks due to the application of network coding techniques [9,13]. Those benefits become even more important in sensor network environments where resources are scarce. Network coding consists of a combination of packets to be transmitted. Such technique has security properties intrinsically related since more than one message is codified in only one. A particular kind of combination based on bitwise exclusive-OR operations has been used by well-known cipher families for encryption and decryption of messages, for example, the *Vernam* cipher.

Oliveira and Barros [20] proposed a system based on predistribution, mobility and distribution of keys using network coding. In such scheme, the task of key distribution in a WSN is accomplished by a mobile node. This system, however, has a limitation: it allows an attacker to discover all keys used in the system when accessing the data in any sensor along with the mobile node memory access. On the other hand, the system proposed by Kazienko and Albuquerque [14] deals with the problem of security in sensor networks from a broader perspective. The work of Kazienko and Albuquerque [14] considers the need for secure key storage in sensor nodes quite important since they are great in number and their content is, in thesis, more subject to capturing and tampering attempts. Thus, besides security in key distribution it is also intended—as explained in Section 3—to provide a larger resiliency and robustness to the system in such a way that the sensors' capturing and tampering will not result in any key discovery.

3. SYSTEM'S DESCRIPTION

In their previous work [14], Kazienko and Albuquerque have introduced a secure key distribution and storage scheme for WSNs. That system is depicted in Figure 1. The main contribution is a mechanism to solve the stored key exposure problem. This is pointed out in the key management research area as relevant and open issue, especially for WSNs [11,22].

Sensors typically operate unattended and a secure key storage is necessary in order to protect key material stored in a given sensor. Additionally, the authors have proved mathematically that their proposal increases the system security [14]. In such scheme, a special node called Mobile Node (MN) is used during the key distribution phase. Their main conclusion is that the probability of system key discovery remains the same even when an attacker gains access to the memory content of the MN and of a regular sensor of the network, simultaneously. Such proof is solely mathematically. The present research effort intends to extend this validation in order to demonstrate the scheme's feasibility through a practical evaluation.

The scheme is basically composed of two phases. The first one is a predistribution phase, in which a set of keys is generated and loaded into the sensors. The second one is the operational phase, where pairwise key distribution based on network coding takes place [9,13]. MN accomplishes a combination of two keys using an exclusive-OR operation in the form $K_i(A) \oplus R \oplus K_i(B) \oplus R$, which results in $K_i(A) \oplus K_i(B)$. Such information is sent through the wireless medium and received by the sensors which, in turn, recover the key from each other in an encrypted way. Such encryption is accomplished with a pre-stored secret Z_i present within a tamper proof area called Cryptographic Module (CM) [7,18,19] of sensor *i*. Afterwards, that same secret Z_i is used to decrypt the key and, finally, encrypt messages that will be sent.

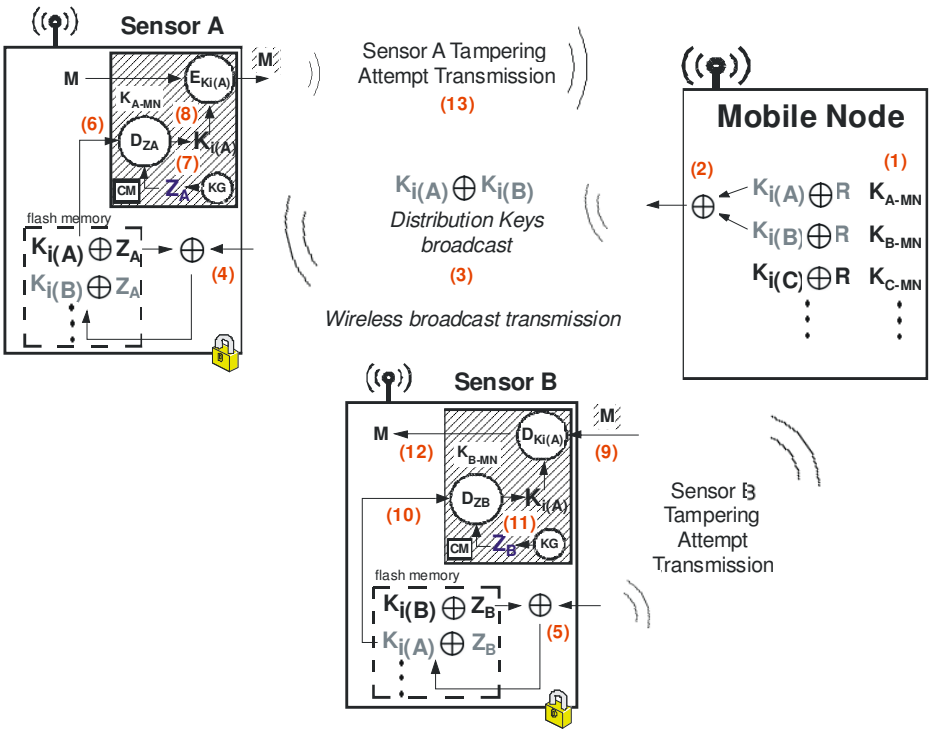

Figure 1. Kazienko and Albuquerque scheme for pairwise symmetric key distribution and secure storage by network sensor nodes. A secret key *Zi* is a parameter of the Cryptographic Module (CM). Such secret is used to encrypt all other sensor keys stored into its flash memory.

Differently from Kazienko and Albuquerque's scheme, the key distribution system proposed by Oliveira and Barros [20]—presented in Figure 2—does not accomplish the secure key storage. This is a problem because the keys used in the system may be discovered if an attacker captures a regular sensor of the network and the MN, simultaneously. This problem is solved by the scheme of Kazienko and Albuquerque. In this work, we compare both schemes through simulation. Additionally, we evaluate a system without cryptography that sends messages and stores keys as plain text.

Beyond the confidentiality, it is important to provide authentication. With this purpose, an authentication protocol of sensors avoids the distribution of keys to fake nodes. Due to this fact, such authentication should take place before the key distribution phase in order to verify the node's legitimacy. Kazienko and Albuquerque also define an authentication protocol [14], which derives from a Key Distribution Center. The MN shares different keys with each node of the network. During the authentication phase, it uses a nonce-based challenge-response scheme. Each sensor replies the encrypted version of the nonces with its own key and then MN verifies it and authenticates the sensor.

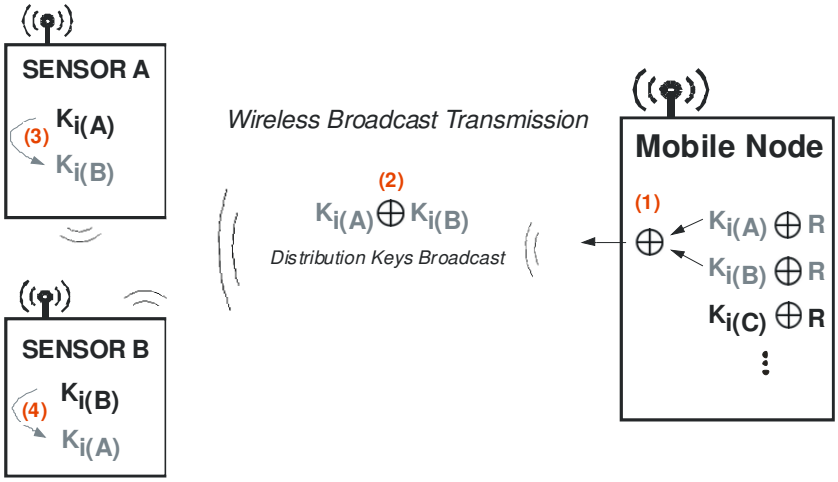

Figure 2. Oliveira and Barros scheme [20] for pairwise cryptographic key distribution in a WSN. The keys are stored as plain text into sensors A and B. If the MN and a regular sensor of the network are captured, it is easy to derive the information *R* that is used to encrypt all system's keys compromising the security.

4. METHODOLOGY

In this work, the following tools, environments, and platforms are used to develop the experiments with sensor nodes:

- *TinyOS* [17] is a component-based operating system specifically developed for WSNs. TinyOS is written in nesC language and it is free and open source;
- *nesC* (network embedded systems C) [10] is a dialect of C language. It is an event-driven programming language used to create sensor applications to run on TinyOS;
- *TOSSIM* [16] is the TinyOS Simulator. More specifically, it is a library of TinyOS. TOSSIM is widely used by researchers because the same code developed for simulation runs on the real sensor hardware. This feature makes the development of WSN applications faster. TOSSIM is a discrete event simulator and works by translating hardware interrupts into discrete events;
- *XubuntOS* is a Linux distribution with the TinyOS programming framework embedded. This distribution allows the development of TinyOS applications because a given code may be compiled to a specific sensor platform.

4.1 The *nesC* Programming Language

The nesC programming language is an extension of C language designed to generate optimized code for embedded systems like sensor nodes in a WSN. All applications written in nesC are developed through the construction and reuse of pieces of code called components. The components are divided into two categories: modules and configurations. The module contains the implementation of the algorithms themselves through declarations of variables, functions, etc. The settings are used to connect components to assemble larger and more complex components. Besides the modules and configurations, there are also interfaces that are composed of function signatures—without implementation—and can be provided or used by the components. The functions contained in the interfaces can represent actions or events where actions must be implemented by the component that provides the interface while the events should be implemented by the component that uses it. The interfaces implemented by a component are the only means of accessing that component [10].

4.2 The TinyOS Operating System

The TinyOS operating system, developed in nesC, is designed to assist the development of applications for WSNs. Its philosophy differs radically from traditional operating systems since it does not reside in the sensors independently from applications. In fact, when a program is developed, it uses components that represent abstractions of real components of the hardware to perform the desired activities. Thus, TinyOS can be considered a framework for developing applications for sensors [17].

The development of a system aided by TinyOS consists in creating components using other components already available in the framework. For example, in the sensor application development accomplished in this work, we developed a component that exchanges encrypted messages among sensors. The logic behind this application is developed but how sensors send packets to the real network interface is abstracted through communication-specific components of TinyOS.

4.3 The TOSSIM Simulator

TOSSIM is a simulator for WSNs that comes along with the TinyOS environment [16]. In order to run simulations, one must first define the desired network topology to be simulated and the channel noise model. The topology provides the identifiers for each node, the links among network nodes, and the signal strength of each of these links. The noise model is used to simulate the noise in the environments where a WSN should operate. After the establishment of the topology, it is necessary to build a Python script setting out guidelines on how the execution of the simulation must proceed.

Debug information may be obtained about the application being simulated with the use of the *dbg* command. The *dbg* command prints out certain data in an output device such as screen or a text file. The output device used is defined by creating a channel in the Python script. The channel has a name and it should be passed as an argument to the function *dbg* in the program written in nesC.

5. EXPERIMENTS

In this section, the components of a sensor and the simulation scenario are described.

5.1 Sensor's Components

Nowadays, available sensors consist of a set of embedded hardware with limited resources. A typical sensor node is composed of four components:

- A *power unit*, responsible for supplying energy to other components;
- A *sensing unit*, that actually contains the sensor, for instance, of light, humidity, temperature, etc;
- A *computing unit*, composed of RAM and flash memories and a processor that typically uses a set of analog-to-digital converters (ADCs) to obtain data from sensors and communications protocols;
- A *communication unit*, used to send and receive radio signals.

Figure 3 presents the general schematic of a typical sensor node hardware [1,2].

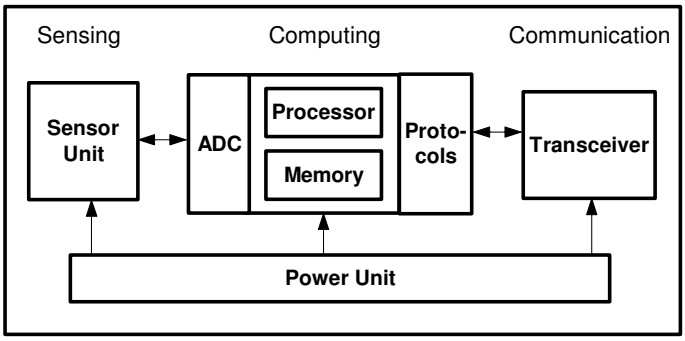

Figure 3. General schematic of a typical sensor node hardware. The main components are the power, sensing, computing, and communication units.

With the purpose of mapping to the real world architecture aforementioned, one kind of mote used in this work is the off-the-shelf sensor TelosB (from UC Berkeley, Crossbow). It consists of 16-bit Texas Inst. MSP430 F1611 processor, 10 kB of RAM memory, 48 kB of flash memory, transmission rate of 250 kb/s by using a Bluetooth radio and it is programmable through a USB interface [5]. Another kind of mote also used in this work is the MicaZ that consists of a processor Atmel ATmega 128L MPR2400, 4 kB of RAM memory, 128 kB of flash memory, and transmission rate of 250 kb/s with Bluetooth radio [4]. Both motes are equipped with the radio model TI CC2420.

In our simulation experiments, the same code plays the tasks of key distribution and of sending/receiving messages. Thus, just one mote plays the role of key distribution while the other motes are regular motes, that is, they only send and receive encrypted messages. On other hand, in the practical experiments presented in Section 7, we split the code in key distribution code and regular mote code in order to avoid the loading of unnecessary code in the sensors.

As a last observation, it is important to highlight that the TinyOS Simulator considers only the MicaZ architecture in its compilation process. Therefore, our simulation experiments use such architecture. Besides, such feature facilitates the loading of simulated code mainly into MicaZ real sensors. In the practical experiments, we also consider one TelosB mote.

5.2 Simulation Scenario

In this work, we have implemented and modeled three WSN communication scenarios. The *first scenario* is the one that considers the system introduced by Kazienko and Albuquerque [14], i. e., a system with secure key distribution and storage. As the key storage is encrypted, it is necessary to decrypt the key before its use. The *second scenario* considers the scheme proposed by Oliveira and Barros [20]. In such communication scheme there is a secure key distribution, but not secure key storage. The keys are stored in sensor's RAM as plain text and, therefore, there is no need to do any decryption of keys prior to sending messages. The *third scenario* does not use any cryptography. There is no key distribution and the messages are sent through the wireless medium as plain text.

In Figure 4, our experimental evaluating scenario is depicted. We have implemented the operational phase of the Kazienko and Albuquerque scheme, shown in Figure 1, with some modifications. Firstly, all sensor's keys are loaded in RAM memory instead of in sensor's flash memory. Additionally, since CM is not present in our sensors we assume that the secret Z_i is stored in memory. Z_i is a sensitive information used to encrypt all other keys of the sensor. In this first scenario, such information is stored into the sensor's RAM memory only for experimental purposes.

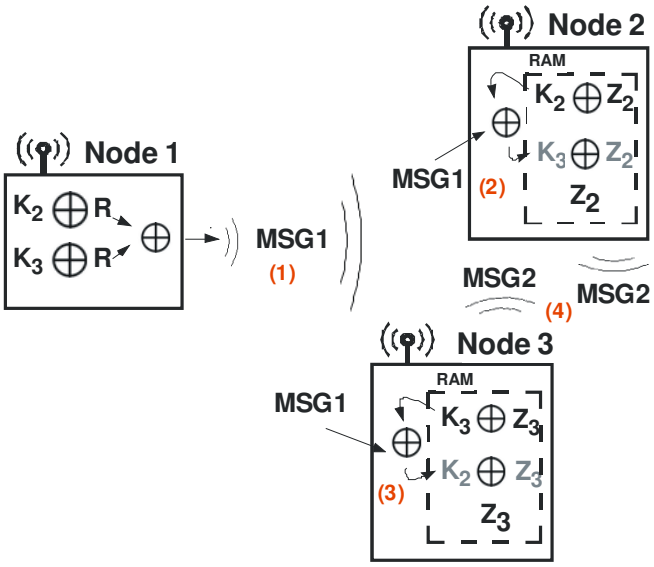

Figure 4. Experimental evaluating scenario based on [14]. The operational phase of such scheme was modeled and implemented. All keys are stored into the RAM memory of the sensor.

The evaluation scenario is composed of three sensor nodes. The TOSSIM simulation environment is configured to support bidirectional links between all three nodes. There is no mobility in such evaluation scenario.

In Figure 4, step (1), Node 1—that models the Mobile Node—performs the key distribution. It accomplishes the sending of MSG1 that is composed of the key identifiers and the XOR operation between these keys stored by the MN, one from Node 2 and another one from Node 3. In steps (2) and (3), sensor nodes receive MSG1 and recover the keys from each other, but in a encrypted way with the information Z_i. In step (4), all communication between the Nodes 2 and 3 are encrypted. In this step sensors exchange 8-byte MSG2 messages. These messages are logged into message sent and received files. Afterwards, such logs are analyzed. In our implementation, every message is encrypted with the target's key.

6. SIMULATION RESULTS

This section presents our evaluation results through simulation. An experimental evaluation was accomplished to measure the processing overhead of both schemes: the one proposed by Kazienko and Albuquerque [14] and the one proposed by Oliveira and Barros [20]. Our goal is to verify the level of overhead caused by the additional encryption needed to solve the key exposure problem by using the Kazienko and Albuquerque scheme. Table 1 shows the TOSSIM simulation parameters.

Table 1. Simulation parameters.

Parameters	Value
Number of Events	10,000,000
Number of Simulations Runs	100
Confidence level	95%
Number of motes	3 motes
Message Size	8 bytes

To start with, it is essential to point out that the comparisons accomplished in this section are based on the three scenarios defined in Subsection 5.2.

In such scenarios, the simulation generates on average 182,000 message transmissions, from those 165,000 messages are received, resulting therefore on average of 10% lost messages. We believe that such loss percentage is due to the interference and collisions that are present in the wireless medium model.

Figure 5 depicts simulation results obtained from 100 simulation runs. In each simulation, the average time required for sending packets in the three scenarios previously described is compared and sorted in decreasing order. Regarding the *first scenario*, that employs the Kazienko and Albuquerque scheme, such time is given by the sum of the following intervals: sender's encryption delay (decryption of key and encryption of message), transmission delay, propagation delay and receiver's decryption delay (decryption of key and decryption of message). Regarding the *second scenario* that employs the Oliveira and Barros scheme, there is no decryption of keys in encryption and decryption delays. Regarding the *third scenario* that does not employ a key distribution scheme, there is no delay related with encryption functions.

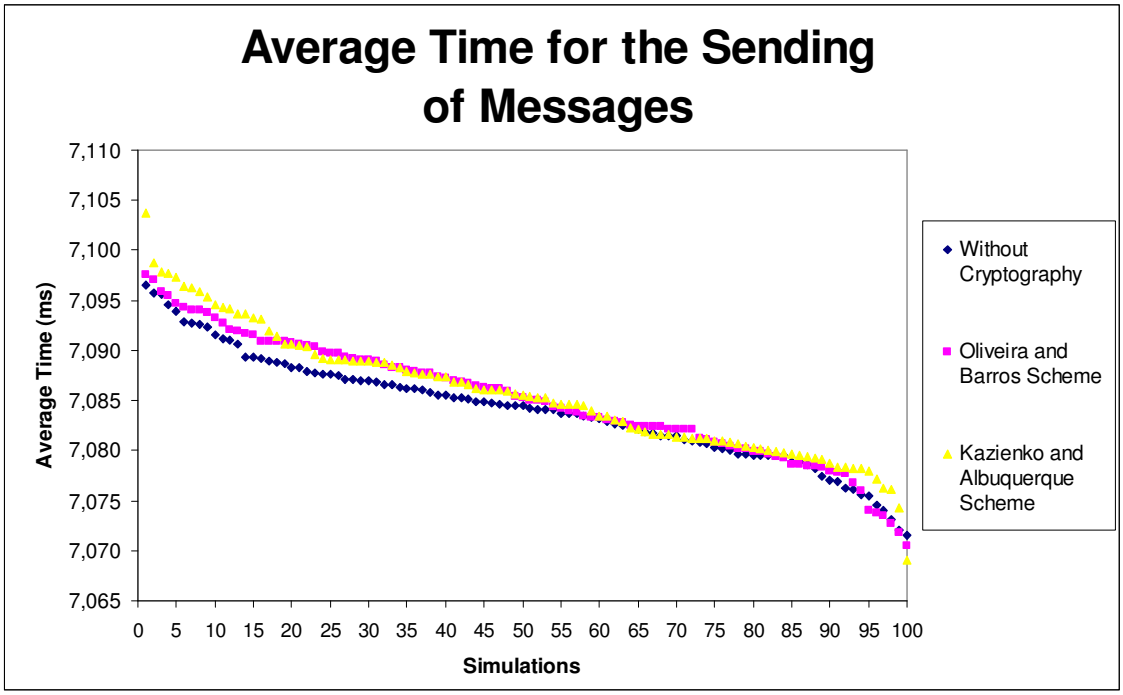

Figure 5. This graphic shows the average time for the sending of messages among systems for 100 simulations. Each curve represents the average time spent by each scheme to send all messages. The results are sorted in decreasing order.

Simulation results indicate a very low difference between the scheme proposed by Oliveira and Barros and the scheme introduced by Kazienko and Albuquerque. This difference is in the order of just 250ns on average. This result is shown afterwards in Figure 6.

Another remark regarding Figure 5 is that the difference between the largest and the smallest sample with Kazienko and Albuquerque scheme is 34μs. On the other hand, the difference between the largest and the smallest sample of the scheme of Oliveira and Barros is just 27μs. Also, the difference between the largest and the smallest sample provided by the system without cryptography is 24μs. In fact, most of the variation in transmission time is due to the CSMA-based medium access control affecting all evaluated systems, as observed in Figure 5.

Figure 6 shows a data compilation from the 100 simulation runs. This figure presents the average time demanded for the system without cryptography (i), the scheme of Oliveira and Barros [20] (ii) and the Kazienko and Albuquerque's scheme (iii) to send messages from Node 2 to Node 3 and vice-versa. We calculated the confidence interval for a 95% confidence level. Error bars are plotted as vertical lines at each point.

We observe a very low time difference between (ii) and (iii) schemes of just 250 nanoseconds on average. Such value is obtained by the difference between 7.085432ms (from Kazienko and Albuquerque's bar) and 7.085182ms (from the Oliveira and Barros's bar). Also, a very low difference of 1μs is observed between (i) and (iii) approaches obtained by the difference between 7.084357ms (from the without cryptography's bar) and 7.085432ms (from Kazienko and Albuquerque's bar).

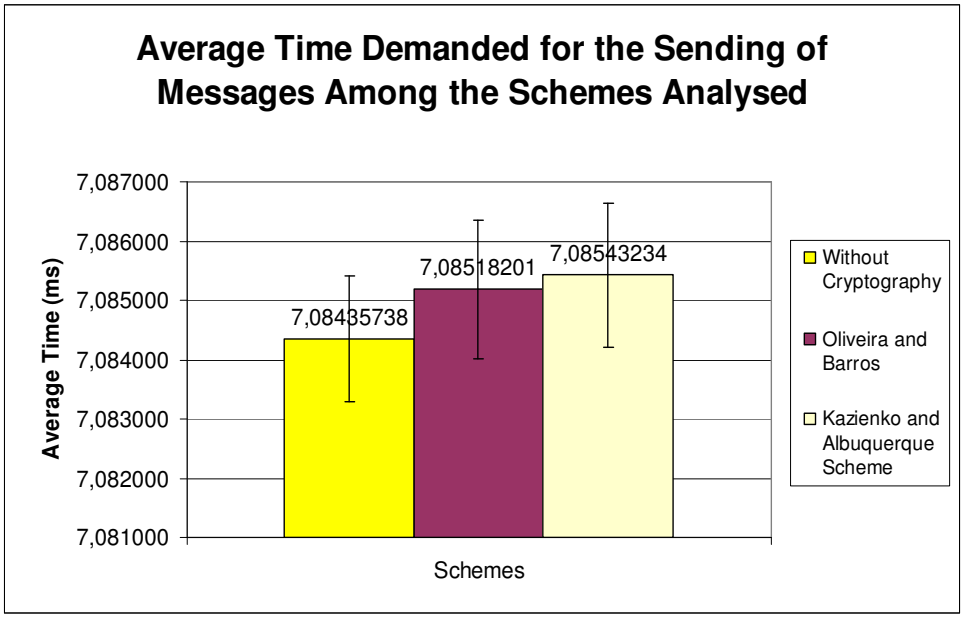

Figure 6. This graphic depicts a compilation from the 100 simulation runs shown in the Figure 5 and compares a system without cryptography (i), the Oliveira and Barros scheme (ii) and the Kazienko and Albuquerque scheme (iii). The difference between (ii) and (iii) is very low of just 250ns on average. Also, the difference between (i) and (iii) remains very low in the order of 1μs.

It is fair to state that Kazienko and Albuquerque scheme introduces processing overhead due to the message encryption algorithms performed by sensors. However, such encryption functions are lightweight, somewhat simple and fast since they consist of XOR operations over bit sequences. We claim such aspect has contributed to a very low processing overhead in the order of nanoseconds on average, as presented in Figure 6. The processing overhead reflects on the energy consumption. This factor should be considered in sensor networks due to their major constraints.

Additionally, the processing overhead caused by Kazienko and Albuquerque scheme remains very low even when compared with the system without cryptography. On average, the difference between these two approaches is in the order of 1μs. Thus, we argue the use of this scheme is feasible since its processing overhead is very low and comes with the added benefit of providing security for a sensor application.

In addition, we carry out measurements in order to verify the processing time required by sensors to perform encryption and decryption functions with their mechanism. The sender sensor has to decrypt the secret *Z* and encrypt

the message. This is the *sender processing time*. On the other hand, the receiver sensor has to decrypt the secret Z and decrypt the message. Such time is the *receiver processing time*. For 182,942 samples collected, in both cases, the processing time is zero considering a precision of nanoseconds.

Another important concern in WSNs is power consumption. Therefore, in this simulation experiment, we also carry out measurements with the aim to estimate the usage of energy by the considered approaches.

In order to estimate the total power consumption of the three motes per simulation run, we only consider the sending and receiving events. In our scenario, the power consumption caused by processing is negligible compared to the energy consumption of radio transmission and reception.

First of all, we need to quantify what is the energy drained by sending and receiving modes. According to the MicaZ architecture data sheet [4], radio reception drains 19.7mA and radio transmission at the highest power available drains 17.4mA. In addition, a regular MicaZ mote is fed by two AA batteries with 1.5V.

Table 2 shows the number of sending and receiving events on average as well as the estimated power consumption per simulation run, considering the three schemes investigated in our study.

Table 2. Number of events of sending and receiving generated on average per simulation run.

	Without Cryptography	**Oliveira and Barros**	**Kazienko and Albuquerque**
Sending events	182,907	182,891	182,952
Receiving events	165,453	165,403	165,310
Estimated power consumption	19,159.8J	19,156.17J	19,154.41J

Analyzing the obtained results in Table 2, we claim that the Kazienko and Albuquerque scheme is feasible from the power consumption point of view since the estimated power consumption is quite similar in all approaches.

7. PRACTICAL EXPERIMENTS

Our practical experiments are performed on four off-the-shelf sensors: one TelosB [5] and three MicaZ [4] motes. In these experiments, only the scheme of Kazienko and Albuquerque is implemented. Different from simulation experiments, the application code is split in order to avoid unnecessary code to be loaded to sensors. The *key distribution code*, called Code D, is loaded into the TelosB mote and the *regular mote code*, called Code R, is loaded into all MicaZ motes. We have documented memory size requirements of our D and R codes. The footprint of such codes, considering flash memory and RAM, is shown in Table 3.

Table 3. Footprint of codes.

	Flash Memory	**RAM Memory**	**Loaded in Mote**
Code D	11.2kB	318B	TelosB
Code R	22.0kB	664B	MicaZ

The results presented in Table 3 confirm that both codes D and R fit well within the flash memory available in TelosB and MicaZ motes. The available memory in such kind of motes is detailed in Subsection 5.1.

In order to perform practical experiments, we built an experimentation scenario. The MicaZ motes are deployed about 15 meters from each other in an indoor environment. Mote 1 is the TelosB distribution node, loaded with code D. This sensor plays the role of MN, as described in Section 2. The other three sensors—the MicaZ motes—are loaded with code R. These motes are regular sensors that begin to send and receive encrypted messages only when the key distribution takes place. These sensors are called Motes 2, 3 and 4. In our scenario, between Motes 2 and 3 there is a wall, while between Motes 3 and 4 there are two walls.

Initially, Mote 1 accomplishes the key distribution of $K2 \oplus K3$ and $K3 \oplus K4$ in sequence, where K2, K3 and K4 are previously known keys from Motes 2, 3 and 4, respectively. We have used the sensors' LEDs in order to indicate the success of some tasks. For example, the key distribution messages are sent in an interval of 10 seconds and when it takes place all sensor LEDs toggle. For the regular sensors, we used the orange LED to indicate the sending and the green LED to indicate the receiving of messages. It is important to highlight that the green LED blinks only

when a message (previously known to the application) is received and correctly decrypted. Figure 7 depicts the network behavior when the Mote 1 is placed in the middle of the Motes 2 and 3.

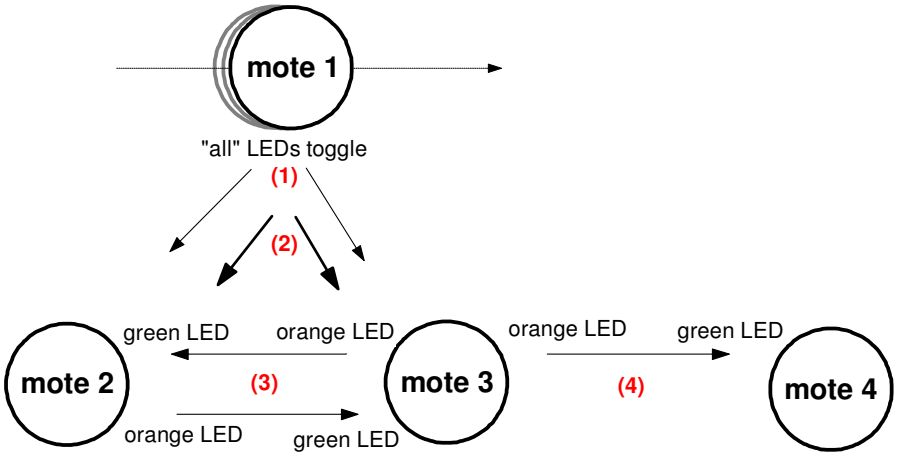

Figure 7. This diagram depicts the network situation when Mote 1 is placed in the middle of Mote 2 and Mote 3; and out of the range of Mote 4. Motes 2 and 3 receive the XOR transmission and recover the keys from each other. Just at this moment, they begin to exchange messages between themselves.

In Figure 7, step (1), the key distribution transmission of K2 ⊕ K3 is accomplished. In a similar way, in step (2), it is accomplished the transmission of K3 ⊕ K4. Steps (1) and (2) take place in sequence with the periodicity of 10 seconds, as mentioned. In step (3), Motes 2 and 3 receive such transmissions and recover the keys. Motes 2 and 3 recover the keys from each other and then they start to exchange encrypted messages. Moreover, although Mote 4 is out of the range of Mote 1, Mote 3 received the transmission of K3 ⊕ K4 and recovered the key of Mote 4. Thus, in the step (4), the Mote 3 sends encrypted messages to Mote 4. In this experiment, we have defined that the messages are encrypted with the target's key. Figure 8 depicts the mobility of Mote 1 towards the range of Motes 3 and 4.

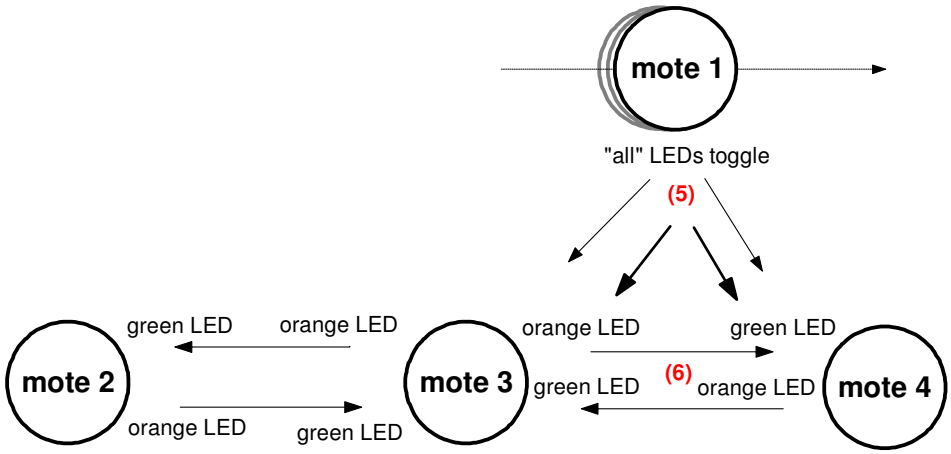

Figure 8. This diagram depicts the network situation when Mote 1 is placed in the middle of Mote 3 and Mote 4; and out of the range of Mote 2. Motes 3 and 4 receive the XOR transmission and recover the keys. After that, Mote 4 is able to recover the key of Mote 3 and then to send encrypted messages to this mote.

In Figure 8, step (5), the same key distribution messages are transmitted, as explained in the steps (1) and (2) from Figure 7. However, the MN is carried to the range of Motes 3 and 4; and out of the range of Mote 2. Since Mote 4 receives the transmission of K3 ⊕ K4, it can recover the key of Mote 3 and starts sending encrypted messages to this mote, as it is possible to see in the step (6).

After the mobility of the Node 1 presented in Figures 7 and 8, the orange and the green lights of all Motes 2, 3 and 4 remained blinking throughout the duration of the experiment. Such result indicated the messages kept being encrypted and decrypted correctly. This validates the functionality of the scheme and its feasibility to run on real sensor platforms.

8. CONCLUSION AND FUTURE WORKS

Currently, experimental evaluation in wireless sensor networks is broadly supported by TinyOS. In this work, we argue in favor of the practical feasibility of implementing the scheme proposed by Kazienko and Albuquerque using TinyOS.

Simulation results reveal a low and negligible processing overhead when compared to existing approaches. Additionally, the power consumption observed is quite similar among all approaches. Moreover, our experimental evaluation was composed of practical experiments. Two classes of the application were developed: the key distribution code and the regular mote code. In both cases, the applications fit well within the TelosB and MicaZ sensors memory size. Furthermore, when such codes were loaded into such sensors, we observed that the pairwise key distribution and the encrypted message exchanging were being successfully accomplished.

Even though our implementation does not fully model the original scheme in which the information *Z* is protected in a cryptographic module area, we claim that such hardware implementation would make the encryption and decryption of keys even faster.

For future works, we intend to extend and pursue additional practical experiments. First, we intend to increase the number of sensors in order to extend our evaluation regarding scalability. Second, we wish to implement and evaluate the feasibility of an authentication protocol, as proposed by Kazienko and Albuquerque [14].

Acknowledgments

This research effort is financed by Capes, CNPq and Faperj.

References

[1] AKYILDIZ, I. F.; SU, W.; SANKARASUBRAMANIAM, Y.; CAYIRCI, E. Wireless Sensor Netwoks: a Survey. *Computer Networks*. 38(4):393-422, 2002.

[2] BECHER, A; BENENSON, Z.; DORNSEIF, M. Tampering with Motes: Real-tampering Physical Attacks on Wireless Sensor Networks. In: *3rd International Conference on Security in Pervasive Computing (SPC)*, p.1-15, 2006.

[3] BIRYUKOV, A.; KHOVRATOVICH, D. Related-key Cryptanalysis of the full AES-192 and AES-256. *Cryptology ePrint Archive, Rep.2009/317*. <http://eprint.iacr.org/>, 2009.

[4] CROSSBOW. *MicaZ datasheet*. Available in: <http://www.xbow.com>, accessed in: Sept. 2010.

[5] CROSSBOW. *TelosB datasheet*. Available in: <http://www.xbow.com>, accessed in: Oct. 2010.

[6] CULLER, D., ESTRIN, D., SRIVASTAVA, M. Overview of Sensor Networks. *IEEE Computer Magazine*. 37(8):41-49, 2004.

[7] EREN, H.; SANDOR, L. Fringe-effect Capacitive Proximity Sensors for Tamper Proof Enclosures. In: *IEEE Sensors for Industry Conference (SIcon'05)*, p.22-26, 2005.

[8] ESCHENAUER, L.; GLIGOR, V. D. A Key-management Scheme for Distributed Sensor Networks. In: *9th ACM Conference on Computer and Communications Security*, p.41-47, 2002.

[9] FRAGOULI, C.; BOUDEC, J.-Y. L.; WIDMER, J. Network Coding: An Instant Primer. *ACM SIGCOMM Computer Communication Review*, 36(1):63-68, 2006.

[10] GAY, D.; LEVIS, P.; BEHREN, R. Von; WELSH, M.; BREWER, E.; CULLER, D. The nesC Language: A Holistic Approach to Networked Embedded Systems. In: *PLDI '03: Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*. New York: ACM, 2003, pp.1–11.

[11] HU, F.; SHARMA, N. K. Security Considerations in Ad Hoc Sensor Networks. *Ad Hoc Networks*, vol. 3, no. 1, pp. 69–89, Jan. 2005.

[12] I-NRG. *Inter-networking research group*. Available in: <http://inrg.cse.ucsc.edu>, accessed in: Sept. 2010.

[13] KATTI, S.; RAHUL, H.; HU, W.; KATABI, D.; MÉDARD, M.; CROWCROFT, J. XORs in the Air: Practical Wireless Network Coding. *IEEE/ACM Transactions on Networking*, 16(3):497-510, 2008.

[14] KAZIENKO, J. F.; ALBUQUERQUE, C. V. N. Autenticação, Distribuição de Chaves e Armazenamento Seguro em Redes de Sensores Sem Fio. In: *XXXVI Conferência Latino-americana de Informática (CLEI2010)*, Asunción, Paraguay, 2010, pp.1-14.

[15] KAZIENKO, J. F.; ALBUQUERQUE, C. V. N. Secure Secret Key Distribution and Storage in Wireless Sensor Networks. In: *Third IEEE International Symposium on Trust, Security and Privacy for Emerging Applications, in Conjunction with 10th IEEE International Conference on Computer and Information Technology (TSP/CIT'10)*. Bradford, UK: IEEE Computer Society, 2010, pp.890-895.

[16] LEVIS, P.; LEE, N.; WELSH, M.; CULLER, D. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In: *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, 2003, pp.126-137.

[17] LEVIS, P.; MADDEN, S.; POLASTRE, J.; SZEWCZYK, R.; WHITEHOUSE, K.; WOO, A.; GAY, D.; HILL, J.; WELSH, M.; BREWER, E.; CULLER, D. TinyOS: An Operating System for Sensor Networks. *TinyOS Team*. Springer-Verlag, 2004.

[18] LIM, D.; LEE, J. W.; GASSEND, B.; SUH, G. E.; DIJK, M. V.; DEVADAS, S. Extracting Secret Keys from Integrated Circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13(10):1200-1205, 2005.

[19] MARTINA, J. E.; DE SOUZA, T. C. S.; CUSTÓDIO, R. F. OpenHSM: An Open Key Life Cycle Protocol for Public Key Infrastructure's Hardware Security Modules. In: *European PKI Workshop: Theory and Practice (EuroPKI'07)*, pp.220-235, 2007.

[20] OLIVEIRA, P. F.; BARROS, J. A Network Coding Approach to Secret Key Distribution. *IEEE Transactions on Information Forensics and Security*, 3(3):414-423, 2008.

[21] WANG, H.; LI, Q. Efficient Implementation of Public Key Cryptosystems in MICAz and TelosB motes. *College of William and Mary, Tech. Rep. WM-CS-2006-07*, Oct., 2006.

[22] ZHOU, Y.; FANG, Y.; ZHANG, Y. Securing Wireless Sensor Networks: a Survey. *IEEE Communications Surveys and Tutorials*, 10(3):6-28, 2008.