Security / energy compromise in wireless sensor network

Adel Elgaber

SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UFC UNIVERSITÉ DE FRANCHE-COMTÉ

Security/Energy Compromise in Wireless Sensor Networks

ADEL ELGABER

SPIM

Thèse de Doctorat

UFC

école doctorale **sciences pour l'ingénieur et microtechniques**

UNIVERSITÉ DE FRANCHE-COMTÉ

N° | X | X | X

THÈSE présentée par

ADEL ELGABER

pour obtenir le

Grade de Docteur de
l'Université de Franche-Comté

Spécialité : **Informatique**

Security/Energy Compromise in Wireless Sensor Networks

Unité de Recherche :
Laboratoire Femto-ST
Département d'Informatique des Systèmes Complexes

Soutenue publiquement le 26 août 2014 devant le Jury composé de :

MICHAEL KRAJECKI	Rapporteur	Professeur à l'Université de Reims Champagne-Ardennes
PASCAL LORENZ	Rapporteur	Professeur à l'Université de Haute Alsace
FLORENT NOLOT	Rapporteur	Maître de Conférences HDR à l'Université de Reims Champagne-Ardennes
HERVÉ GUYENNET	Directeur de thèse	Professeur à l'Université de Franche-Comté
JULIEN BERNARD	Encadrant de thèse	Maître de Conférences à l'Université de Franche-Comté

ACKNOWLEDGEMENTS

This thesis has done by support many people to me to be in best position. I would like to thank all people who are close to me in this period of my work and in my life.
First of all I would like to thank my god to give me all the best to finish my work . Really, I remember that moment that give me strong support and give me big emotion in my heart to France and French people when my supervisor, my co-director and LIFC (Laboratoire d'Informatique de l'Université de Franche-Comté) told me (we are with you for any help) when my country was in Civil war. I would like to express my gratitude to my supervisor, Professor Hervé Guyennet (Professor in university of Franche-Comté) , I deeply appreciate the help of my co-director researcher Julien Bernard (Lecturer in university of Franche-Comté). I would like to express my deepest gratitude for his excellent guidance and patience.
Second, I would like to thank my wife Slisal Basma who support and help me to work in best situations. Parental satisfaction is best way to do anything in the life. I would like to express my deepest gratitude to my parents Tbra and Dae Elgaber for their patience and care in all my life.

CONTENTS

1

INTRODUCTION

These days, wireless networks develop very quickly and they are part of our everyday life for sensing and processing information about us or about our environment. Life is very complex and most of new technologies is working to help us to make our life as easy as possible. The collection of information about us and our life is very important to new technologies, we study and analyze the information in order to take the right decision to improve our life. For getting some information about our environment, we need some devices that can give us the values of data for some nature Phenomenon. These devices have special characteristics that can do its work beyond limited resources.

Wireless sensor network(WSN) is a network of (low-size and low-complex) devices denoted as nodes that can sense the environment and communicate the information gathered from the monitored field through wireless links. the data is forwarded via multiple hops to a sink node in same network.

The main application areas for WSNs are categorized according to the type of information measured or carried by the network. for example, medical application is important in our life, WSNs solve the problem of emergency, by fix some sensor node on body of patient that can give us reading of pressure, heartbeat, and etc to avoid the danger state, also wireless sensor networks help the patients who are in long distance from hospital. In this case the doctor can give the medicine depending on data that sensed by sensor node of patient. Military applications are very closely related to the concept of wireless sensor networks. The sensor node is so small and when it is in the area of enemy, it will be so useful to air-force for attack the target with high accuracy, also by sensor nodes can give the army precise information on Battlefield surveillance.

Most of sensor nodes deploy in unsecured area, and all transmitted data are broadcasting, data transmitted is very important in some applications. To protect our data and to avoid change data we choose cryptography algorithm that can give us level of security without effect on power of sensor. But for doing this method we must respect all resources of sensor node to keep it live. By increasing level of sensor work, energy consumption will effect, battery power will effect life of sensor. Sensor node has some advantages like small, monitor something long time but also has some constraints that give it special work beyond limited conditions to be working long life, these constraints like energy consumption, level of security, and it use wireless networks to exchange data through the network or with another networks. For protect our information we should select level of security that give the sensor suitable life more than to its security in some applications.

The nature of information is changing. So the communication between sources of information and processing devices is important. The best way to exchange data is wireless

communication because the targets may be moving. Most processing devices are mobile and powered by battery that has a limited life. The main disadvantage of wireless networks is that it can be easily attacked. So to get an optimal system to improve our life, the devices need to have a high level of power, and secure communications.

For getting optimal system of wireless sensor network, we need to know most of challengers that may be make some constraints on our proposal. WSNs mote is powered by battery and in normal case all battery has limited life, so to make sensor node live long time we must study all factors that may be effect power of sensor. Function of node always reading data, process and transmission, but the WSN usually deploy in danger area that are so difficult to change its sensor's battery. One of conditions of optimal system of WSN is security, we need to protect our data through network by applying or choosing best cryptography algorithms of WSN. To do or apply this solution we should know How much selected algorithm will effect the power of node?.

Sensor node has limited memory size (approximately view kilobytes), to apply any type of operations like computation we should know How much this operation will need memory size to be done completely?. The resources of sensor node(mote) are limited and its function is so precise to send data in time in some applications, data transmission consumes high value of energy of node. To send data in time we need to calculate or to know approximately How much data transmission will consumes energy for one time transmission?. To decided How we can save the power of sensor node. Optimal system of WSN needs many things that can consider its resources, similarly for efficiency any system, it will be solve most of problems that building on them. sensor node deploy in some area that has special characteristics like forest, building, and Battlefield, so the main target of building the system is using sensor node's resources to kept sensor live, to get this result function of application must respect all constraints of sensor. For example, high level of security will effect the power of sensor that will cause problem of sensor's life. Cryptography algorithm has many computation operations will effect the memory size of sensor that will cause problem of transmission time.

Our work focuses on a way to make a balance between a high level of security and a low level of energy consumption.

Most sensor nodes communicate through a wireless network, and they are used for some applications that have different jobs. For getting the data about some natural phenomenon, the sensor node consumes some energy to sense and store this data. The amount of energy changes depending on the type of application: for example, sensing temperature each hour is different than monitoring fire when it happens.

Most applications has important information, and wireless communication has a problem of security. So to solve this point, we encrypt sent data through the network. The encryption consumes more energy that effects the lifetime of the sensor node to increasing the consumed energy and decreasing the lifetime of sensor node.

Most sensor nodes have limited memory, and each sensor works for different applications. Some applications focus on speed of arrived information, and others focus on accurate data. Using encryption, the memory of the sensor node is effected, and the sent message is delayed. Due to the limited size of memory, the packet size is changed depending on the method for sending data, and this also changes the level of consumed energy.

With the compression of the original data, we can get the encrypted data in a memory with limited size and control the packet size of the message, but that may effect the arrived

time and the energy consumption.

In our work, we look at the characteristics of different sensor nodes (TelosB, MicaZ, WSN430) to study which one is the best for different applications, with a focus on how we can send a message from the sensor node to a base station as a case study. We choose the Senslab platform to calculate the energy consumed for sending a encrypted message. Senslab consists of lots of sensor nodes of four sites with fixed and mobile nodes.

For encrypting a message, we try to find a suitable algorithm depending on the wanted level of security and the consumed energy. We work on the different encryption algorithms. We choose two algorithms to explain our target that is getting an optimal system that can control the level of security (sending secure data for important applications) and level of energy consumption (getting a long life). So the study and implementation of the code of each algorithm on sensor of the Senslab platform is the first step to calculate the amount of consumed energy on the sensor node. The second step is choosing best compression algorithms to help us solve the problem of memory size.

After doing the experiments, we analyze the results to find general models of energy. The models are calculating the consumed energy for encryption and transmission of a message as one operation, then calculate the consumed energy for compression, encryption and transmission as one operation. These models give us a general vision on the value of consumed energy for sending message from one point to another point in a network depending on different factors.

Our target is getting optimal system with a good balance. With our general model, we can increase and decrease the level of security depending on the type of application and lifetime of sensor.

Depending on the selected application, we choose one encryption algorithm and one compression algorithm. For choosing an algorithm, we examine the conditions of applications. For example, if the application is for a military purpose, its first goal is to protect the information, and its second goal is to keep the sensor live as long as possible. If we worry about the high level security to protect the information, we can choose encryption algorithm that has a good level of security, but not a long lifetime.

When we implement the encryption algorithm, we face a challenge of memory size, as we know the sensor node has limited size. Choosing the application and its conditions to exchange data through the network, we notice that encryption has an effect on memory size. But increasing the data storage size on total memory of sensor does not guaranty that there are no missed data. It means we need to make a control between all these operations to protect sensed data when the memory is full, but in some case, the applications need to send data direct in minimum time delay. So to solve this issue, we propose to compress the sensed data and then encrypt.

For compressing sensed data, we can choose the algorithm that has low energy consumption, but the conditions of application may vary and the algorithm depends on the type of compression (lossy or lossless) and the ratio of compression.

This proposal also needs to make control in time-delay, because the speed of arrived data is so important in some application. By changing the generated data rate, we can improve the time-delay.

Our work is built on the fact that each user may have one or multiple goals when using

the sensors. The first step is to set the function of the application, then choose the conditions of arrived data. There are some balance between all these conditions, for example a high level of security affects the time delay. To solve these challenges, we propose some solutions. We use the Senslab platform to get real results, on a real sensor node, we test various encryption and compression algorithms, then get the results for different configurations. We find an energy consumption model for encryption, compression and communication, and apply it to a linear network. With this model, we can see the maximum and minimum level of security we can get. To improve our proposal, we generate random networks, because the nature of topology of sensor will be random. For this work we use our own simulation software to build and test the model on random network. Finally, we work on the generated data, as we know the packet has a fixed size and we take into account the memory of the node and the time for the different operations, including transmission.

This thesis has contributions in both practical and theoretical results:

1. The main contribution of this thesis is a relationship between the level of security and the energy consumption of the sensor node. This relationship is built on different factors that impact on both energy and security. It allows us to know the lifetime of a sensor node under a selected level of security.

2. We show that improving the security of a sensor node is possible without consuming extra energy. This is done by combining encryption with a compression algorithm that reduce the amount of data to be encrypted and sent. In some configurations, these savings are greater than the energy needed for compression.

3. The importance of sensed data is the first condition when choosing the level of security. But this level will be restricted by the memory size of a sensor node and the time delay of sent messages.

4. We are able to calculate the consumed energy on a random wireless network. This gives us a precise vision on the maximum level of security that we can get.

After this general introduction, we present in Chapter 2 everything needed for explain this thesis. We define the wireless sensor network to discuss all the sensor node and its components to know the characteristic of node, the operating system of sensor node that have special resources we should respect them, the challengers and applications of wireless sensor network as full network, senslab platform that. We discuss data compression to explain important classification of compression methods, to find best compression techniques, to select compression algorithms that we can use for wireless sensor networks. We present data cryptography to see which best way of encryption that give us clear idea to choose proper cryptography algorithm, then present the most challengers of apply cryptography in wireless sensor network. And finally we present some of details about data communication to explain way of data transmission through the wireless sensor network, choose the energy model for communication.

In Chapter 3, we present our practical work that was done by apply our proposal code on senslab platform. We have done the experiments by choosing cryptography algorithms to study and check the values of energy consumption in each selected algorithm, We found the energy model of encryption algorithms depending on our results of our experiments. Next step is find the model of size memory of sensor in case of apply cryptography algorithm, similarly, we find the model of time delay of data transmission. We calculate the

values of energy consumption with compression algorithm. And finally we find the model of memory size and time delay for sensor node when apply compression algorithm.

In Chapter 4, we discuss the anlysis of calculate the maximum level of security that we can obtain it, we study and anylsis our scenarios with three main operations that apply on one sensor node to find values of energy consumption in each case, We explain how the linear network will be working and find the model of calculate the maximum number of rounds of cryptography algorithm. And finally we check and find proper level of security in random network in different mixed cryptography and compression algorithms.

In Chapter 5, We describe our proposal of getting optimal system of data transmission with proper level of security, available memory size, and in available best time. We present our strategies to find algorithm that we use to calculate the values of our conditions in our system. We analysis and check the energy of transmission in each situations of data sending. At end we find the proper level of memory size and time delay of sensor for each case of our proposal.

In Chapter 6, At end of this thesis, we present the main point that we have obtain about our idea. We present some points to improve our work.

2

Overview of wireless sensor networks

Contents

In this chapter, we give an overview of wireless sensor networks and some of its applications. In section 2.1, we first describe a sensor node and its components. Then we present wireless sensor networks, i.e. when all the nodes communicate together to achieve the same goal. Finally, we show the Senslab platform, which is an experimental platform that we used for our experiments. In section 2.2, we describe some common compression techniques and especially the techniques that are used on wireless sensor nodes. In section 2.3, we present some cryptographic algorithms and their specialization on wireless sensor nodes. In section 2.4, we describe the communication models in wireless sensor networks, and especially the energy models.

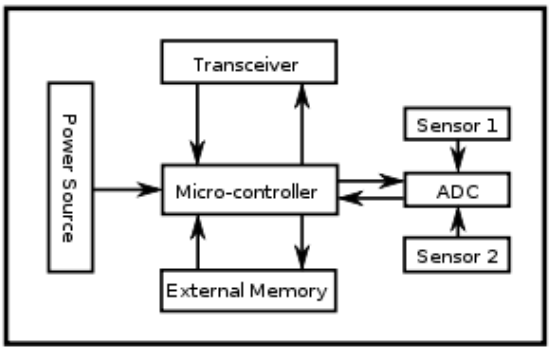

Figure 2.1: A Generic Sensor Node Diagram

2.1/ Wireless Sensor Networks (WSN)

In this section, we study most of sides of sensor networks. We start from sensor node (or mote) to know its components that are used for sensing, processing and storing data. Then we explain the operating systems of sensor node. By collecting a set of nodes, we form a full network of sensors that can exchange data between nodes and base station. The Senslab platform is used to run real code on real sensor nodes, we describe the method of work on this platform.

2.1.1/ Node Definition

A *sensor node* in a wireless sensor network collects information and communicates with other nodes in a network to tranfer the information to a *base station*. A sensor node is also called a mote. A node consists in several components: micro-controller, transceiver, external memory, power source and one or more sensors. Figure 2.1 shows a generic diagram for a sensor node.

2.1.2/ Architecture of a Node

Each sensor node consists in several components that we detail here.

2.1.2.1/ Power

Generally, a node is powered with batteries, and more precisely with AA batteries. AA cells have an operating range between 2.1 to 3.6V DC, this voltage must be at least 2.7V, and each mote needs two AA battery. A node may also be powered by the host computer with its USB port (2.7V to 3.3V).

In all cases, the input voltage should be maximum 3.6V because increasing the values of voltage can damage the micro-controller, the radio, or other components.

	MicaZ (Fig. 2.2)	TelosB (Fig. 2.3)	WSN430 (Fig. 2.4)
Micro-controller	Atmega128L 8bits	MSP430 16bits	MSP430 16bits
Frequency (MHz)	16	1.1	1.1
EPROM/ROM (KB)	4	16	48
RAM (KB)	4	10	10
Transceiver	CC2420	CC2420	CC1101/CC2420
Protocols	802.15.4/ZigBee	802.15.4/ZigBee	802.15.4
TX rate (kbps)	250	250	250–500
Size (inches)	2.25 x 1.25 x 0.25	2.55 x 1.24 x 0.24	?
Weight (g)	23	18	?
Battery	2 x AA batteries	2 x AA batteries	PoliFlex Cell
Serial	UART	UART	2 x UART

Table 2.1: Examples of Sensor Nodes

2.1.2.2/ MICRO-CONTROLLER

Each node has a processing unit that support a low energy consumption. It works in different modes depending on the situation of sensor node. It has two main modes: active mode and sleep mode. The sleep mode has different states for saving more or less energy.

2.1.2.3/ RADIO

For sending and receiving data, a node uses a radio chip, also called transceiver. Generally, it's a TI CC2420 radio for wireless communications because the CC2420 can operate at low power. The CC2420 is controlled by the micro-controller. The radio may be shut off by the micro-controller for low power duty cycled operations.

2.1.2.4/ ANTENNA

Most sensor nodes have two kinds of antenna, internal and external antenna. An antenna may attain a 50 meter range indoors and up to 125 meter range outdoors.

2.1.2.5/ EXAMPLES OF SENSOR NODE

Common nodes are MicaZ (figure 2.2) and TelosB (figure 2.3), and each one has its own characteristics [31]. We added WSN430 (figure 2.4) that is used in Senslab in the comparison of table 2.1.

2.1.3/ OPERATING SYSTEM

Sensor nodes use specialized operating systems (OS) that have some conditions about energy management and hardware. Table 2.2 shows the OS support for each sensor nodes.

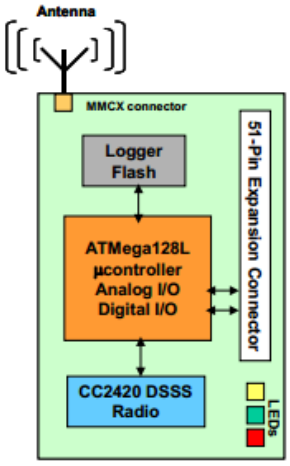

Figure 2.2: Block Diagram of MicaZ

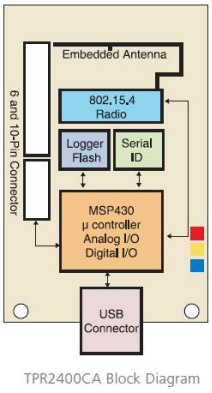

Figure 2.3: Block Diagram of TelosB

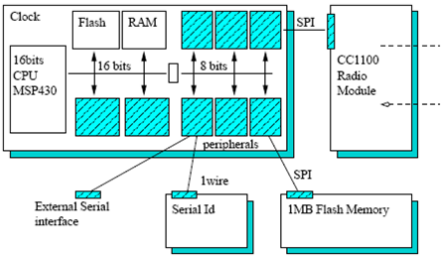

Figure 2.4: Block Diagram of WSN430

	MicaZ	TelosB	WSN430
TinyOS	✓	✓	✓
Contiki		✓	✓
MantisOS	✓	✓	
SOS	✓	✓	
Nano-RK	✓		
FreeRTOS			✓

Table 2.2: Operating System Support of Sensor Nodes

2.1.3.1/ TinyOS

TinyOS is an open source operating system that was developed at University of California in Berkeley and designed for low-power wireless devices that consist of micro-controllers with a few kilobytes of RAM and a small size of code space. It is used for sensor networks, personal area networks and smart buildings. The programs of TinyOS are written in the NesC language, organized in different components, and uses an event-driven programming model.

The operating system contains a set of components, and these components consist of three elements: Commands, Events, and Tasks. All these elements are C functions but there are some rules they follow in term of who can call them and when they get call. An event is used to signal the completion of a request. A command request a component to do something. A task is not executed immediately, but are executed later by the scheduler. Components control which commands can be sent and which event can work through the interface. All components that are used are known at compile time, so the operating system uses less than 400 bytes of program memory.

The main challenge in sensor network is energy-efficiency performance, the replacement of battery is costly and often difficult in inaccessible area. At each node, much energy is consumed by radio communication but to minimize energy consumption or save some energy, the sensor changes its state between sleep and active mode. Most of the lifetime of a sensor is spent sleeping but depending on the application and type of data that is sent through the network, the state of sensor will change.

In sleep mode the current is the lowest, in contrast most of operation will be done in active mode like transmission. So power consumption or energy consumption should be low, because most wireless sensor nodes run on battery power. The micro-controller has different modes of power in sleep state for keeping the power as low as possible.

For managing the power, TinyOS has three things that control the work. First, by calling the command `StdControl.stop`, all the equipment can stop itself. Second, for identifying the processor's status, TinyOS will check the input/output pins and control register of the processor. Third, by the Low Power Mode, the timer can work in lowest power cost mode [61, 35].

2.1.3.2/ Contiki

Contiki is an open source operating systems for memory-constrained systems with a particular focus on low-power wireless devices like sensor nodes. It was developed by Dunkels et al. The main features of Contiki are dynamic loading and unloading of code at run time. Contiki only needs about 10 kilobytes of RAM and 30 kilobytes of ROM. A running Contiki system consists of two parts, the operating system core and the loaded programs. The core consists of different things: the kernel, libraries, drivers and program loader. The loader executes initialization function of loaded program.

As for energy consumption, Contiki provides a set of mechanisms for reducing the power consumption of the system on which it runs. The default mechanism for getting low-power operation of the radio is called ContikiMAC. Nodes can be running in low-power mode, by putting the micro-controller or peripheral hardware in sleep modes [61, 76].

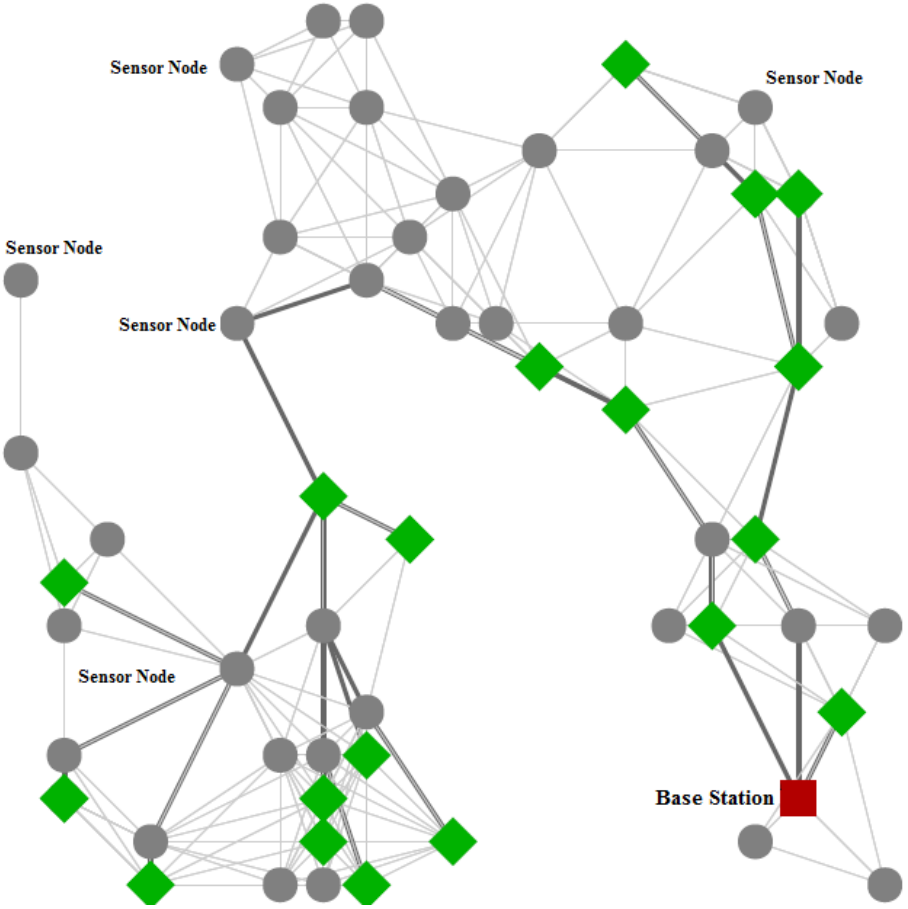

Figure 2.5: An example of a Wireless Sensor Network topology

2.1.4/ Wireless Sensor Network

In this section, we present some principles about WSNs, what is the wireless sensor network?. We discuss some applications of wireless sensor networks. We present special challengers in WSNs. And last we present the model of wireless sensor network that we are using.

2.1.4.1/ Definition of a Wireless Sensor Network

A wireless sensor network (WSN) is an important technologies at present, it is a collection of wireless nodes organized into a cooperative network for lots of purpose [87]. Sensor node can be used to detect or monitor a variety of physical parameters like light, sound, humility, temperature. Every node has some attributes such as position, speed, and direction [3, 19].

Each node has set of functions, sense information, process sensed data and transmit the processed data to the sink or base station (BS), all nodes self-organize and communicate to each other [11, 42], they forms a distributed topology as shown in figure 2.5.

We can distinguish three roles for nodes in a wireless sensor network.

- **Base station:** This node is also called *gateway*, it's the target for the data in the network. The node with a red square on figure 2.5 is a base station.
- **Sensing node:** These nodes are the ones that are sensing and producing some data. They are the source of the messages in the network. The nodes with green squares on figure 2.5 are sensing nodes.
- **Router nodes:** These nodes are responsible for transmitting the data from sensing nodes to the base station. The nodes with grey disks on figure 2.5 are router nodes.

2.1.4.2/ APPLICATION TYPES

Wireless communications need a high level of security because most wireless networks are easy to attack from unauthorized persons. On the other hand, it has lot of applications in various fields like medicine, habitat monitoring, military, and its small size allows it to be easily deployed in the monitoring area without being noticed. These application are divided into four classes based on the data delivery and communication patterns between the sensor node and the sink [85, 70, 4].

- **The continuous model:** The sensor nodes periodically report to the sink a data analysis. Example: a sensor monitoring a field for agriculture that sends temperatures and light levels [42].
- **The event-driven model:** The sensor nodes send information only when the event of interest is detected. Example: a sensor detecting forest fires (it will send information when the temperature increasing over a given temperature) [42].
- **The observer-initiated model:** The base station sends a clear request to specific sensors to get an answer or a reaction. Example: some events at certain geographic area [42].
- **The hybrid model:** This model consists in a combination of several models in same network [42].

2.1.4.3/ CHALLENGES IN WIRELESS SENSOR NETWORKS

A wireless sensor network faces many challenges like higher unreliability of sensor nodes, energy consumption, computation and storage constraints [32], since sensor nodes are energy- and size- limited, how to make wireless sensor networks work as long as possible is one of the main challenges in the design of wireless sensor networks[11]. The lifetime of sensors is limited, it depends on the type of battery, type of application, radio transmissions, and data storage [9].

Therefore, the power analysis is the most important point in the development of a sensor network application. The energy consumption of a sensor node can change depending on the application algorithms and operating system [2, 84, 27, 18].

With the analysis of critical operations on limited power source, it gives the designer and users a clear idea about the lifetime of sensor nodes [25, 14]. Moreover the limited memory can cause a problem to store a large set of data, and also the processor can cause problem with the time for computation [15, 24].

Figure 2.6: Unreliable Channel

As we said the battery has limited power which are difficult or impossible to replace in most situation, and the main challenge is to maximize the lifetime of the sensor [73]. A sensor consumes most of its energy in communication with other sensors. To minimize the energy, one can use data compression, but it can cause an issue in data accuracy [82, 63].

2.1.4.4/ THEORETICAL MODEL OF A WIRELESS SENSOR NETWORK

To modelize a wireless sensor network, we use the graph representation [69]. A node is represented by a vertex and a communication link is represented by an edge between two nodes. The edge can be directed if the nodes are homogeneous, or undirected if the nodes are heterogeneous.

The communication links between nodes can be determined with several strategies. In our simulations, we used Unit Disk Graphs [69]. In this type of undirected graphs, a range is chosen for the nodes. For every node i, all the nodes that are in the range of i, i.e. that are at a distance less than the chosen range, are linked to node i. They form the *neighborhood* of i.

In information theory, a communication is modelised with two actors, the source and the destination, and a message that go through an unreliable channel, as shown in figure (2.6. Coding theory is responsible to ensure the integrity of the message. Compression is responsible to make the communication efficient (in terms of number of transmitted bits). Cryptography is responsible to make the message safe.

In our case, we consider the transmission link to be unsafe. Due to the wireless nature of communications, an attacker can easily listen to every communication between two nodes. This makes the communication channel unsafe by nature and that is the main problem we address. Moreover, communication is the biggest energy consumer on a wireless node. So with compression, we target efficiency in terms of transmitted bits and as a consequence, efficiency in terms of consumed energy. On the contrary, we consider that the channel has no transmission errors, i.e. errors due to a physical failure. Our point is not fault tolerance.

2.1.5/ SENSLAB

Senslab [64, 22] is a Wireless Sensor Network experimental platform. It is deployed on four sites in France (Lille, Strasbourg, Rennes, Grenoble), it consists in 1024 sensors (256 sensors for each site) with fixed and mobile nodes. All these sensors can be accessed by

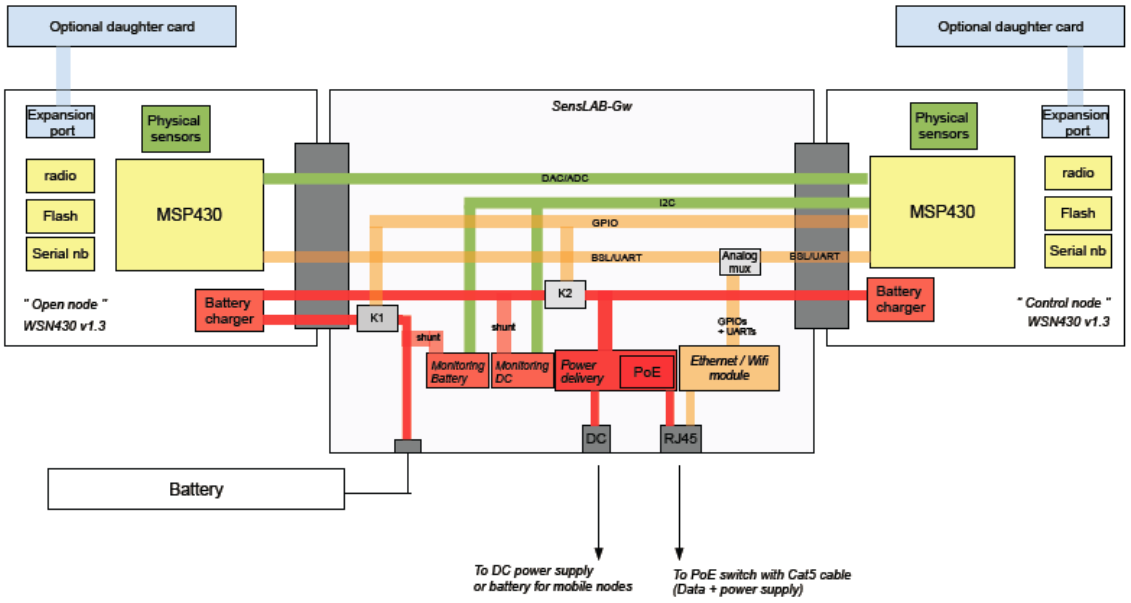

Figure 2.7: Senslab node logical architecture

the user, and each sensor can measure electric power consumption, radio activity, light and temperature.

Thanks to the Senslab web portal, each user can launch his experiment. The website offers an interface between the user and nodes on platform and a control to give the available nodes to only one user at any point in time. A virtual machine (VM) is responsible for the connection between the user and the running experiment nodes.

The Senslab platform is using the WSN430 sensor board that consists in memory chip, MSP430 microcontroller, CC2420 radio chip, DS1722 temperature sensor and TSL2550 luminosity sensor.

2.1.5.1/ TESTBED DESCRIPTION

A Senslab node consists in two WSN430 nodes and a gateway. Figure 2.7 shows the logical architecture of a Senslab node and figure 2.8 shows a photo a a Senslab node.

The open node contains a standard sensor node, it has a MSP430 micro-controller, radio interface and a battery. It is the part which allows the user to execute the experiments. The control node is responsible on measurement features during the experiments, it allows the open node to be started up and programmed. The gateway is responsible for the communication between the open node and the control node and supplies the energy to them. It also controls the communication between the Senslab node and the platform. The aim of this board set is to offer essential Senslab features, such as:

- Automated firmware deployment on open node.
- Accurate power monitoring of open nodes: battery and DC power supply
- Radio environment monitoring control: RSSI measures and noise injection

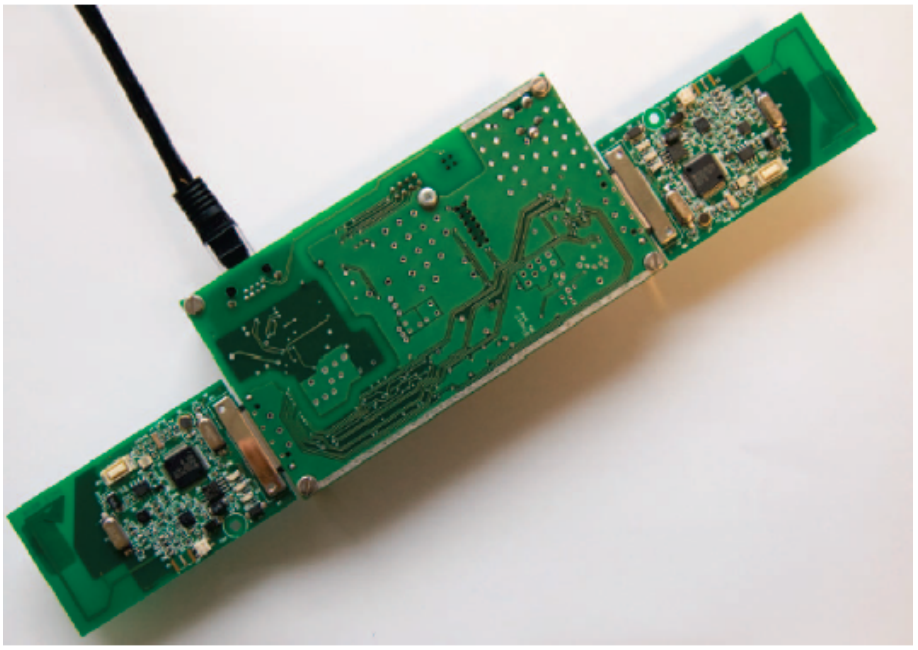

Figure 2.8: Photo of a Senslab node

- Configurable sensor polling on control node: temperature, light, acoustic activity
- Fixed (Ethernet) as well as mobile (Wifi) communication with node handler
- Remote software update ability for controlling nodes and gateway.

2.1.5.2/ Workflow with Senslab

The services of the testbed are divided into two parts. First the web portal that is used for submitting the experiments. Second, the virtual machine (VM) that allows the user to interact with the nodes during the experiments. Monitoring data is collected in files with comma-separated values (CSV) for each experiment. These files are accessible in the virtual machine for subsequent analysis.

Each experiment has a special profile that explains its conditions. A profile has many arguments like power supply or the sensor/radio/power monitoring performed by the control node during the experiment. When the experiment has started the user can connect with the selected nodes through the web portal or his VM, and make control over this node to see the run of program.

2.2/ Data compression

In this section, we give an overview of data compression. We present some compression techniques that are used in some algorithms, also we choose common compression algorithms that we apply in wireless sensor networks.

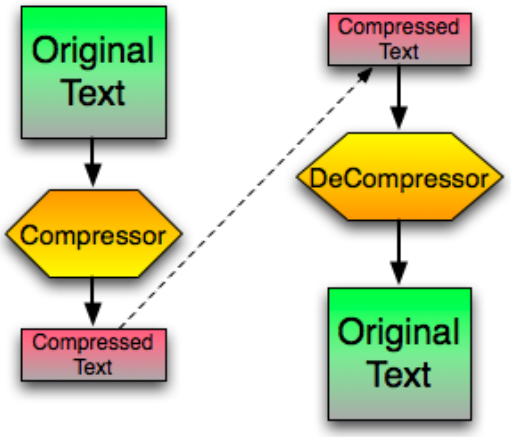

Figure 2.9: The compression principle

2.2.1/ OVERVIEW OF DATA COMPRESSION

Data compression is a common requirement for most of the computerized applications. There are number of data compression algorithms, which are dedicated to compress different data formats. It is the operation that reduces the amount of original data when we want transmit or store a huge file which needs lots of resources [58].

When data compression is used in a data transmission application, speed is the primary goal. Speed of transmission depends upon the number of bits sent, the time required for the encoder to generate the coded message and the time required for the decoder to recover the original ensemble. In a data storage application, the degree of compression is the primary concern. In data transmission applications, the important thing is the speed of transmission that depends on the number of sent bits. So by doing compression, the number of bits is decreased. The time for generating the coded message and the time required for decoder to obtain the original message may be of interest in the application. In data storage applications, the level of compression is the primary concern to know how the amount of data decreases [58, 5].

We define the *compression ratio*, noted α, to be the reduction in size relative to the uncompressed size. If the uncompressed size is u and the compressed size is c, then:

$$\alpha = 1 - \frac{c}{u}$$

When $\alpha = 1 = 100\%$, no compression occurs. When $\alpha = 0.3 = 30\%$, this means that the size decreased by 70%.

Compression can be classified into lossy or lossless [67]. The compression ratio depends on the actual type of information. Decompression is the opposite operation, in order to retrieve the original data. Figure 2.9 shows the principle of compression and decompression of a text.

2.2.2/ CLASSIFICATION OF DATA COMPRESSION METHODS

Data Compression methods depend on the studies of human perception, In addition the natural of data(all contents are important or some of them). There are two classes of compression: lossless and lossy compression [67].

2.2.2.1/ LOSSLESS COMPRESSION

Lossless data compression is one of common methods in data compression of all types of data, in this type the original data will be perfectly reconstructed from compressed data that received on decoder, it is used in case of important data that used in many applications need same original data, some example of lossless data compression is used in some image file format in medical field[58]. steps of doing lossless compression divided into two things in sequence. First is generates a statistical model for the input data, statistical model is generated by two methods. Adaptive models dynamically update the model as the data is compressed. Static model stored compression model of input with compressed data. Second step uses one of these models to map input data to bit sequences, for producing bit sequences we use one of encoding algorithms. Various lossless data compression algorithms have been proposed and used. Some of the main techniques in use are the Huffman Coding, Run Length Encoding, Arithmetic Encoding.

2.2.2.2/ LOSSY COMPRESSION

Lossy data compression is way of compress data in low size of bits as possible, it reduces the original size of compressed data when decompress it, it means some information of original data will be lost. It use for many applications that has small size of memory, most of applications that are using this type of compression are about (audio, video and internet telephone)[67, 58].

2.2.3/ DATA COMPRESSION TECHNIQUES

The goal of data compression is to reduce the number of bits required to store any type of data (text, binary, sound, etc) by using statistical properties of data. The reduction is done by a computational process to compress and decompress. Most data compression techniques that are used in most compression algorithms depend on repeated bytes in the full stream and use some methods that calculate the numbers of repeated letters (or bits) [71]. There are several families of compression techniques, they are fundamentally different in their solution to the problem.

Today complex compression systems use one or more techniques from each family to achieve the greatest possible reduction of data, and decreasing the size of actual data is done by different models. Some of main techniques are Huffman Coding, Run Length Encoding, Arithmetic Encoding and Dictionary Based Encoding [54, 75].

2.2.3.1/ HUFFMAN CODING

A Huffman Coding is a lossless data compression algorithm, it is high efficiently with compression ratio from 20% to 90%. Huffman Encoding Algorithms use the probability distribution of the alphabet of the source to develop the code words for symbols. The frequency distribution of all the characters of the source is calculated in order to calculate the probability distribution. According to the probabilities, the code words are assigned. Shorter code words for higher probabilities and longer code words for smaller probabilities are assigned. For this task a binary tree is created using the symbols as leaves according to their probabilities and paths of those are taken as the code words [54].

Huffman Encoding divided into Two families. First static Huffman Algorithms, it calculates the frequencies first and then generate a common tree for both the compression and decompression processes. Details of this tree should be saved or transferred with the compressed file. Second Adaptive Huffman Algorithms, it develops the tree while calculating the frequencies and there will be two trees in both the processes. In this approach, a tree is generated with the flag symbol in the beginning and is updated as the next symbol is read.

2.2.3.2/ ARITHMETIC ENCODING

Arithmetic coding is a lossless data compression, it typically has a better compression ratio than Huffman coding. In this method, a code word is not used to represent a symbol of the text. Instead it uses a fraction to represent the entire source message. The occurrence probabilities and the cumulative probabilities of a set of symbols in the source message are taken into account. The cumulative probability range is used in both compression and decompression processes. In the encoding process, the cumulative probabilities are calculated and the range is created in the beginning. While reading the source character by character, the corresponding range of the character within the cumulative probability range is selected. Then the selected range is divided into sub parts according to the probabilities of the alphabet. Then the next character is read and the corresponding sub range is selected. In this way, characters are read repeatedly until the end of the message is encountered. Finally a number should be taken from the final sub range as the output of the encoding process. This will be a fraction in that sub range. Therefore, the entire source message can be represented using a fraction. To decode the encoded message, the number of characters of the source message and the probability/frequency distribution are needed [6, 54].

2.2.3.3/ RUN LENGTH ENCODING (RLE)

RLE algorithm is one of simplest data compression algorithm. The technique has simple steps: if a data item i occurs n times in an input stream, they are replaced by a couple (n, i). It is a lossless compression technique [12].

More precisely, RLE works by reducing the physical size of a repeating string of characters, this repeating string is called a run. It is typically encoded into two bytes. The first byte represents the number of characters in the run and is called the run count. In practice, an encoded run may contain 1 to 128 or 256 characters, the run count usually contains the number of characters minus one (a value in the range of 0 to 127 or 255). The second

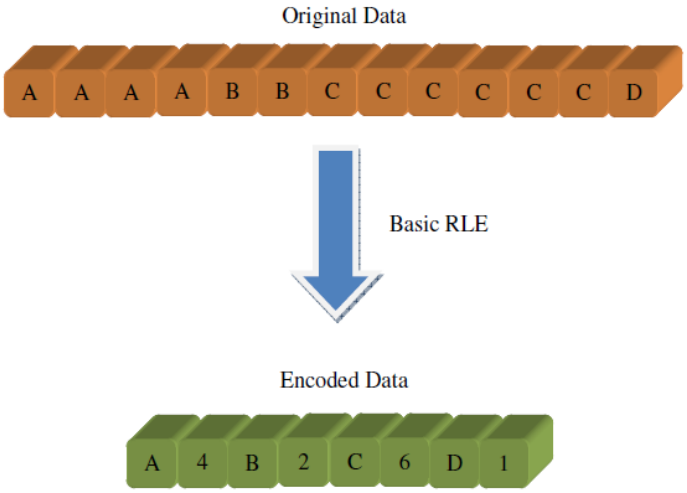

Figure 2.10: The Principle of Run-Length Encoding (RLE)

byte is the value of the character in the run, which is in the range of 0 to 255, and is called the run value. Figure 2.10 show an example of run-length encoding [6].

Algorithm 1 shows the compression with RLE of an input buffer `input` of length `length` in an output buffer `output`. This algorihm is the naive version of the RLE algorithm. It can be improved with many tweaks. For example, for non repeated sequence of characters, the algorithm can encode the length of the non-repeated sequence with a negative number and put the sequence. This saves much space because there is only one count for the whole sequence instead of one per character. This kind of optimization makes the algorithm more difficult to understand and to read, so it has been discarded from the presented algorithm.

Algorithm 2 shows the decompression with RLE of an input buffer `input` of length `length` in an output buffer `output`. The decompression must match the compression, especially in the case of optimizations. Like compression, the version with optimizations is not presented here for simplicity.

For example, if we have a string of 15 `A` characters, it normally requires 15 bytes to store: `AAAAAAAAAAAAAAA`. With RLE, the same string would require only two bytes: one for 15 and one for `A`. The $(15, \texttt{A})$ code generated to represent the character string is called a RLE packet. The first byte, 15, is the run count and contains the number of repetitions, the second byte, `A`, is the run value and contains the actual repeated value in the run. A new packet is generated each time the run character changes, or each time the number of characters in the run exceeds the maximum count.

2.2.4/ Data Compression for Sensor Networks

Wireless sensor networks are resource constraint: limited power supply, bandwidth for communication, processing speed, and memory space. One of best way of getting practical values of those resource is applying data compression on sensor data. The most important advantage of data compression is to reduce the time for data transmission via the communication channel. Also to decrease the total size of original [79].

Algorithm 1 Run-Length Encoding Compression

```
function RLECOMPRESS(output, input, length)
    o ← 0
    i ← 1
    count ← 1
    value ← input[0]
    while i < length do
        if input[i] = value then
            count ← count + 1
        else
            output[o] ← count
            output[o+1] ← value
            o ← o + 2
            count ← 1
            value ← input[i]
        end if
        i ← i + 1
    end while
    output[o] ← count
    output[o+1] ← value
end function
```

Algorithm 2 Run-Length Encoding Decompression

```
function RLEUNCOMPRESS(output, input, length)
    o ← 0
    i ← 0
    while i < length do
        count ← input[i]
        value ← input[i+1]
        x ← 0
        while x < count do
            output[o] ← value
            o ← o + 1
            x ← x + 1
        end while
        i ← i + 1
    end while
end function
```

Usually, processing data consumes much less power than transmitting data in wireless medium, so it is effective to apply data compression before transmitting data for reducing total power consumption by a sensor node [44].

Most existing compression algorithms are not applicable for sensor nodes because of their limited resource. Some of compression algorithms, which have been specifically designed for wireless sensor networks, are studied and presented in our work. The data compression is one effective method to utilize limited resources of wireless sensor networks [40, 57].

2.2.4.1/ The Lempel Ziv Welch algorithm (S-LZW)

LZW is a lossless data compression algorithm based on a dictionary. This algorithm has no transmission overhead and is computationally simple. Since both the sender and the receiver have the initial dictionary (table) and all new dictionary entries are created based on existing dictionary entries, the recipient can recreate the dictionary on the fly as data is received[83].

To adapt the algorithm to sensor nodes, the dictionary can be stored as a hash table with the base entries being the initial dictionary, then strings can be represented as branches from that table so they are easy to locate. To improve LZW for a sensor node (S-LZW), one must balance three major inter-related points: the dictionary size, the size of the data to be compressed, and the protocol to follow when the dictionary fills[45].

2.2.4.2/ The K-RLE algorithm

The idea of the K-RLE [17] is to get more compression with less energy consumption thanks to a lossy and simple compression algorithm. This algorithm can be used when a small variation in the data is not significant.

We assume K is a number. If a data item $d \simeq d + K \simeq d - K$ occur n consecutive times in the input stream, we replace the n occurrences with the single pair (n, d). The parameter K is the precision, it means that the user can not distinguish two values in the range $[d - K, d + K]$ either because the instrument measurement noise is greater than that or because the difference is not significant. So the algorithm is lossless at the user level. If $K = 0$, K-RLE is RLE. K has the same unit as the dataset values.

Algorithm 3 shows the compression with K-RLE. It is quite the same as RLE except the condition for considering that the current input is close to the running value. The decompression algorithm can be exactly the same.

2.3/ Data Cryptography

By using cryptography techniques we can hide original information that we exchange between nodes through network. we present two types of encrypt the data, also study the advantage of some algorithms that we use to protect our data when it is sending. A wireless sensor network is non secure, we select some algorithms that can use to encrypt data through wireless sensor network.

Algorithm 3 Run-Length Encoding Compression

```
function RLECOMPRESS(output, input, length, K)
    o ← 0
    i ← 1
    count ← 1
    value ← input[0]
    while i < length do
        if | input[i] - value | ≤ K then
            count ← count + 1
        else
            output[o] ← count
            output[o+1] ← value
            o ← o + 2
            count ← 1
            value ← input[i]
        end if
        i ← i + 1
    end while
    output[o] ← count
    output[o+1] ← value
end function
```

2.3.1/ OVERVIEW OF DATA CRYPTOGRAPHY

Cryptography has a long history starting by the Egyptians some 4000 years ago and these days, cryptography is improving and developing for getting high security in the transmission of data through the networks. Cryptography is changing the original data that is transmitted through the network or between some nodes by using mathematical techniques to protect the actual content of transmitted message from attackers [28, 13, 80].

To explain the principle of cryptography, we assume there are two persons (points), A and B, as in figure 2.11 . We need to make a communication between them, and we have transmission media to send and receive the message. The problem is how to make sure this media is secure for sending and receiving this message, because this media is open for all. So we do some steps to hide the original content of the message, all these steps it called cryptography. For doing all last steps, we need to follow many cryptography algorithms that have been tested.

The transmission media can be wired or wireless. For wireless sensor networks, we use wireless communications and this type is physically insecure. So for getting more security for transmission data, we use some cryptography algorithms. All these algorithms have a special procedure but most of them depends upon encrypting the data by using keys. Some wireless points has security hardware and others has security software that we can modify as we want [60, 52].

Cryptography is a method of hiding data that can transmitted through the network and only two sides (sender, receiver) can read and process, it is a science of protecting information by encoding it into an unreadable format. Although the goal of cryptography is to hide information from unauthorized individuals, most algorithms can be broken and

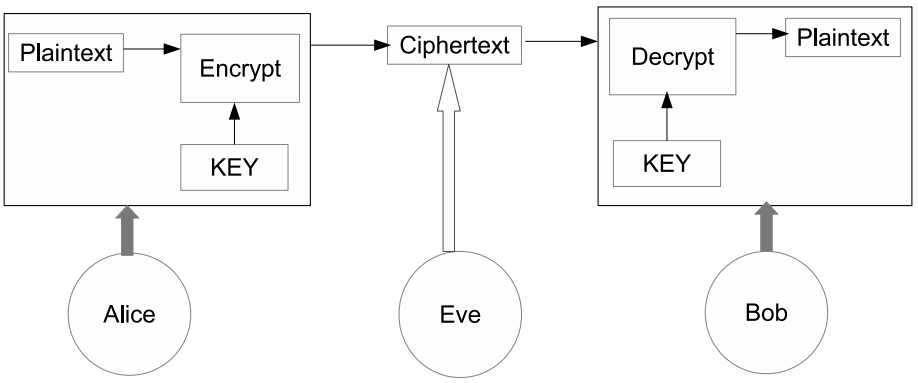

Figure 2.11: Principle of cryptography

the information can be read and changed if the attacker has enough time and resources. So the actual goal of cryptography is need hard and intensive work to obtain the original message from the attacker side.

Cryptography is the art of achieving security by encoding messages to make them non-readable. Cryptography is the study of hiding information that enables you to store sensitive information and also transmit it across insecure networks but it cannot be read by anyone except the intended recipient[43].

Encryption is a transforming original data (plaintext) to get new form that appears to be unreadable (ciphertext). In normal case the data is stored on computer, it is usually protected by logical and physical access controls, but when this data is sending through network, it will be in a vulnerable state For doing the encryption and decryption process, we need to use an encryption algorithm which determines how simple or complex the process will be. Most algorithms are complex mathematical formulas that are applied in a specific sequence to the plaintext, most encryption methods use a secret value called a key (usually a long string of bits) which works with the algorithm to encrypt and decrypt the text.

The algorithm is a set of mathematical and logical rules, it consist of many and different operations apply on original message. Mostly the mechanism of doing encryption for a message is known but the secret part of using encryption algorithm is the key, the key is any value that contains large sequence of random bits.

By cryptosystems, transmission can get more confidentiality, authenticity, integrity, and non-repudiation services. Confidentiality means that unauthorized (person or machine) cannot access information. Authenticity refers to validating the source of the message to ensure the sender is properly identified. Integrity provides assurance that the message was not modified during transmission. Non-repudiation means that a sender cannot deny sending the message at a later date, and the receiver cannot deny receiving it, most of application as military and financial agencies are very concerned about keeping information confidential, so they would choose encryption mechanisms that provide a high security.

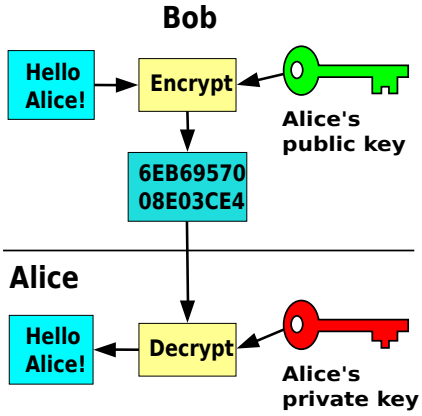

Figure 2.12: Principle of asymmetric cryptographic

2.3.2/ ASYMMETRIC CRYPTOGRAPHY

In this type of cryptography, there are two keys that used for the encryption system: a private key and a public key. The message is encrypted with the public key that is known to all and then decrypted with the private key. Both keys are related to each other by a mathematical equation and it is impossible to get the private key from the public key [47].

We call M the message, e the public key, and d the private key. The public key defines an encryption transformation E_e, while the private key defines the associated decryption transformation D_d. B (point) wish to send a message M to A (point) and has an authentic copy of A's public key e. B obtains the ciphertext $C = E_e(M)$ and transmits it to A. For getting the original message M at A (point), A computes $M = D_d(C)$ as in figure 2.12.

The advantages of asymmetric cryptography are:

- The private key is guaranteeing high security.
- The private key and public key pair may remain unchanged for a long time, and many sessions (even several years).
- In a large network, the number of keys are smaller than in the symmetric-key scenario.

The disadvantages of asymmetric cryptography are:

- Public key encryption is slow compared to symmetric encryption.
- Key sizes are typically much larger than those required for symmetric-key encryption.
- No one can be absolutely sure that a public key belongs to the person it specifies, so everyone must verify that their public keys belong to them.
- It requires a lot of computer supplies compared to symmetric-key encryption.

2.3.2.1/ RSA

The RSA algorithm was published by Rivest, Shamir and Adelman in 1978. It is the best known symmetric cryptographic algorithm. It uses a pair of keys that are generally 1024, 2048 or 4096 bit long. The fundamental idea of RSA is to find two functions D and E so that $D \circ E = Id$ and it is difficult to find D, the decryption function, from E, the encryption function. As a consequence, the function E can be public and allows people to use it to encrypt messages, the only one who can decrypt the ciphertext is the one who has the function D.

In RSA, every computations are modulo n which is the product of two prime numbers p and q. More precisely, here is the protocol to build a pair of key:

1. Choose p and q two prime numbers
2. Compute $n = p \times q$
3. Compute $\varphi(n) = (p-1)(q-1)$. φ is the Euler's totient.
4. Choose an integer e that is prime with $\varphi(n)$ and strictly lower than $\varphi(n)$
5. Compute d, the modular inverse of e modulo $\varphi(n)$, i.e. the number so that $e \times d = 1 \mod \varphi(n)$. This computation is efficient with the Extended Euclidean algorithm.

Then, (e, n) is the public key and (d, n) is the private key. The encryption function on a message m is defined as:

$$E(m) = m^e \mod n$$

And the decryption function on a ciphetext c is defined as:

$$D(c) = c^d \mod n$$

The main drawback of RSA is that it needs huge computations. Indeed, the security of RSA relies on the difficulty to find p and q with n only. This is called the factorization problem. This problem is difficult for big numbers, such as 1024 bit numbers. Those big numbers do not fit in a processor register so they have to be handled by a special library. Computing modular additions, modular multiplications, modular exponentiations with big numbers requires carefully chosen algorithms to be efficient. But they need much memory and processing time which are not compatible with most sensor nodes.

2.3.2.2/ Elliptic Curve Cryptography (ECC)

An elliptic curve is an algebraic structure that has been used in cryptography since 1986. Any elliptic curve can be written as a plane algebraic curve defined by an equation of the form:

$$y^2 = x^3 + ax + b$$

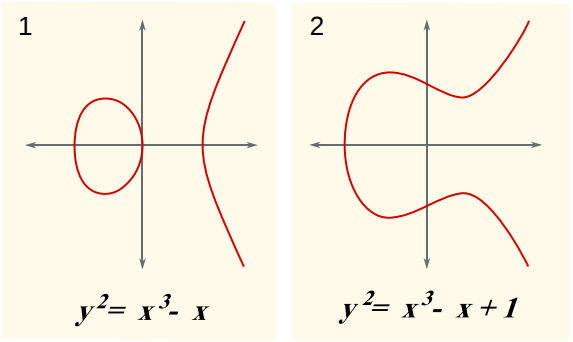

Figure 2.13: Examples of Elliptic Curves

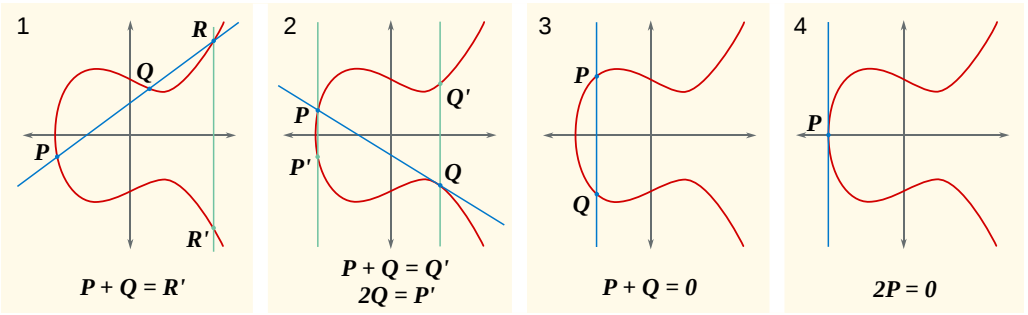

Figure 2.14: Addition on an Elliptic Curve

The set of points of the elliptic curve is all the points that follow the equation and the *point at infinity*, noted O. Figure 2.13 shows two examples of elliptic curves which represent the two types of elliptic curves: curves in one part and curves in two parts.

$E(K)$ is the additive group of all points plus the point at infinity. O is the neutral element of the addition. The opposite of a point $P = (x, y)$ is $-P = (x, -y)$. In elliptic curve cryptography, x and y are part of a finite group $K = \mathbb{F}_q$.

Then, the addition for this group is defined as follows:

- For two points P and Q that are different, if the two points are not opposite, then the PQ line crosses the curve on a third point R. Then by definition, $P + Q + R = O$. This means that the result of the addition $P + Q$ is the opposite of R, which is the symmetric point to R relative to the x-axis.
- For a point P, $P + P$ is computed taking the tangent to the curve at the point P. This line crosses the curve on a second point R and the addition of $P + P$ is the opposite of R.
- For two points P and Q that are opposite, then $P + Q + O = O$. The PQ line is parallel to the y-axis and the result is O.

Figure 2.14 shows the different cases for the addition on an elliptic curve.

Elliptic curve cryptography is based on the difficult problem of discrete logarithm. Before defining this problem, for $k \in \mathbb{N}$, we define $kP = P + P + \ldots + P$ (k times). Then, given an elliptic curve E on a finite field, a point $P \in E$ of order n, i.e. so that $nP = O$, and a point

Bits in RSA key	Bits in ECC key
1024	160
2048	224

Table 2.3: Comparison of key length for RSA and ECC with the same level of security

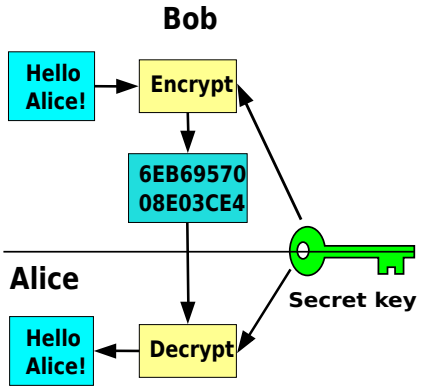

Figure 2.15: Principle of symmetric cryptographic

$Q \in \langle P \rangle = \{kP, 0 \leq k \leq n-1\}$, the problem of the discrete logarithm is finding $k \in [0, n-1]$ so that $Q = kP$.

Elliptic curve cryptography is used for example with the Diffie-Helman protocol for key exchange or for the ElGamal cryptosystem.

The main advantage of ECC over RSA is that it needs keys with fewer bits for the same level of security. Table 2.3 shows a comparison of key length for RSA and ECC with the same level of security.

2.3.3/ SYMMETRIC CRYPTOGRAPHY

In this type of cryptographic, only one key is used for encryption and decryption as in figure 2.15. We call M the message, K the key, C the ciphertext. Then, encryption is defined as $E_K(M) = C$ and decryption is defined as $D_K(C) = M$. The decryption function D is the inverse function of the encryption function E. That means: $D_K(E_K(M)) = M$.

The advantages of symmetric cryptography are:

- This type of encryption is simple and easy to do. All the users have to do is specify and share the secret key and then begin to encrypt and decrypt messages.
- Symmetric key encryption is much faster than asymmetric key encryption. Keys for symmetric-key ciphers are relatively short.
- It does not require a lot of computer resources compared to public cryptography.
- To prevent an attack, a different secret key is used for communication with every different party. If a key is known, only the messages between a particular pair

of sender and receiver are affected. Communications with other people are still secure.

The disadvantages of symmetric cryptography are:

- The key must remain secret at both ends.
- In large networks, it needs many keys, a new shared key has to be generated for communication with every different party. This creates a problem with managing and ensuring the security of all these keys.
- In communication between points A and B, the key may be changed frequently, and perhaps for each communication session. [46]

A block cipher is an encryption method which breaks up the plaintext message to be transmitted into blocks of a fixed length, and encrypts one block at time. Two important classes of block ciphers are substitution ciphers and transposition ciphers.

The block cipher is one of the methods of hiding information. It must divide the plaintext into equally-sized blocks, each block consist of group of characters. The most popular block size is 8 characters, or 64 bits. If the total number of characters in the plaintext is not divisible by the block size (i.e. a complete last block can not be made), then extra characters are generally added to the end of the plaintext until a complete last block can be formed. Once the plaintext is arranged in blocks, then each block is transformed into another equally sized block [37].

For example, if the block size were eight characters, each block would be transformed into a different eight-character ciphertext block. A block cipher can use a number of other cryptographic techniques to transform each block, A block ciphers can be either symmetric-key or public-key, block cipher is a function which maps n-bit plaintext blocks to n-bit cipher text blocks, n is called the block length. It may be viewed as a simple substitution cipher with large character size.

A stream ciphers use a different method, we can see it as a block cipher with a length of block equal to one, it encrypts the message letter by letter. With this advantage, we can avoid the different size of blocks, and also give us probability to encrypt at least one symbol at a time to avoid problems of memory size [46, 43].

A stream cipher is an algorithm that encrypts data one bit at a time. Since stream cipher encrypt data bit by bit, they require that the data be in binary form. Stream ciphers are generally faster than block ciphers in hardware, and have less complex hardware circuitry. It is more appropriate in some telecommunications applications, when buffering is limited. but also may have advantage in situations where transmission errors are highly probable, Stream ciphers can be either symmetric-key or public-key [46].

2.3.3.1/ DATA ENCRYPTION STANDARD (DES)

DES is a block encryption algorithm. A plaintext consist of 64 bits as one block for encryption and for decryption also get 64 bits as one block ciphertext. Also it is a symmetric algorithm, because it use the same key for encryption and decryption. It uses a 64 bit key, 56 bits make up the true key, and 8 bits are used for parity.

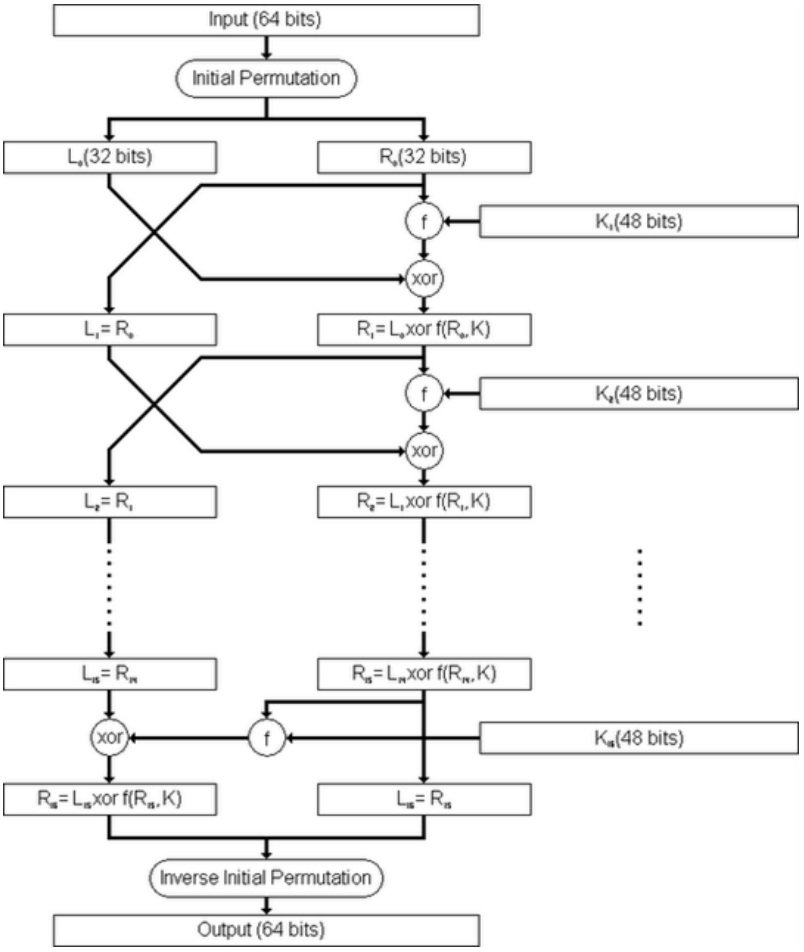

Figure 2.16: The Data Encryption Standard (DES) Algorithm

The DES algorithm works by follow special procedure, it divides the full message into lots of blocks and encrypts one block at a time, each block is 64 bits and is divided in half and each character is encrypted one at a time as in figure 2.16. It has 16 rounds of transposition and substitution functions for each character. The output of encryption is a 64-bit block of ciphertext.

2.3.3.2/ ALGORITHM AES

The algorithm is flexible in supporting any combination of data and key size of 128, 192, and 256 bits, it uses a key (cipher key) whose length can be 128, 192, or 256 bits and is denoted AES-128, AES192, AES-256, respectively. AES allows a 128 bit data length that can be divided into four blocks, These blocks operate on array of bytes and organized as a 4 x 4 matrix. For full encryption, the data is passed through N_r rounds (N_r = 10, 12, 14). It is a symmetric algorithm because the same key is used for encryption and decryption. And it is a block cipher that encrypts a 128-bit block (plaintext) to a 128-bit block (ciphertext), or decrypts a 128-bit block (ciphertext) to a 128-bit block (plaintext).

The encryption procedure consists of several steps. After an initial `AddRoundKey`, a round function is applied to the data block (consisting of `ByteSub`, `ShiftRows`, `MixColumns` and

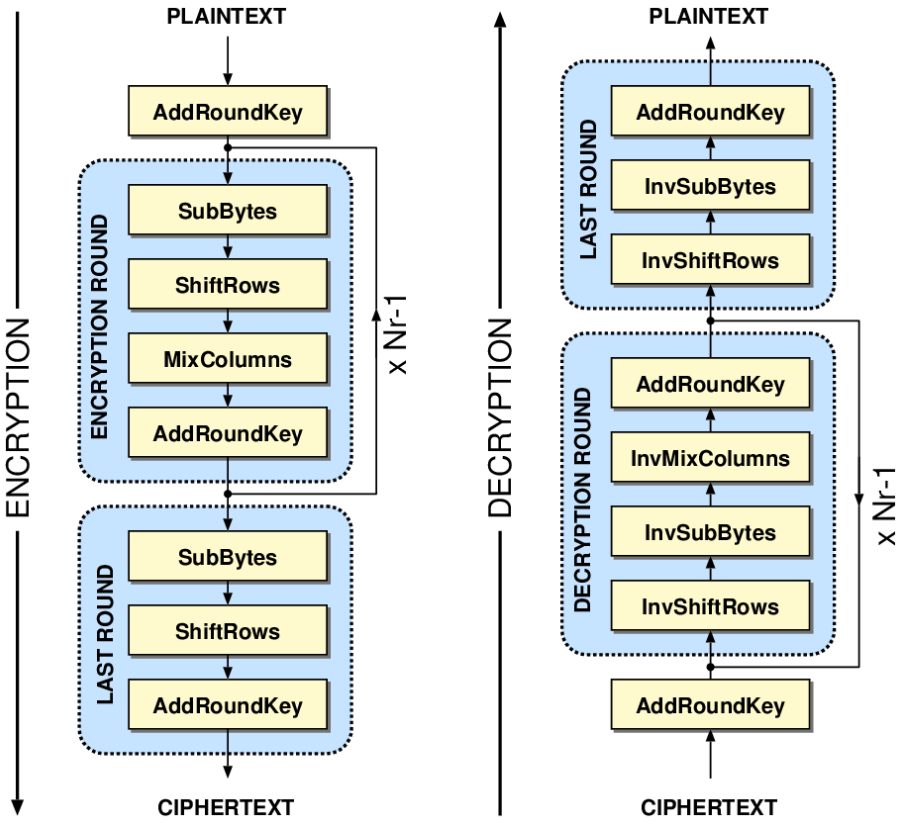

Figure 2.17: The Advanced Encryption Standard (AES) Algorithm

`AddRoundKey` transformation, respectively) as in figure 2.17. It is performed iteratively (N_r times) depending on the key length. The decryption structure has exactly the same sequence of transformations as the one in the encryption structure, the transformations `InvByteSub`, the `InvShiftRows`, the `InvMixColumns`, and the `AddRoundKey` allow the form of the key schedules to be identical for encryption and decryption [86, 56].

2.3.3.3/ ALGORITHM RC5

RC5 is a symmetric block cipher with a variable block size, a variable number of rounds and a variable-length key. This provides the opportunity for great flexibility in both performance characteristics and the level of security [23]. In [77], RC5 is exactly designated as RC5-$w/r/b$, where the variable parameters w, r, and b respectively denote the word size (in bits), the number of rounds, and the length of secret key (in bytes). The standard values of w is 16, 32 and 64, the standard values of r and b range from 0 to 255. The parameter of RC5-32/12/16 is commonly chosen.

There are three components in RC5: key expansion, encryption, and decryption. These components consist of three primitive operations (and their inverse): words addition , bitwise XOR, and data-dependent left rotation. In the key-expansion routine, the user-provided secret key is expanded to fill a key table whose size depends on the number of rounds. The key table is then used in both encryption and decryption.

2.3.4/ CRYPTOGRAPHY IN WIRELESS SENSOR NETWORK

A wireless sensor network is a big challenger for saving the energy, because it consists in a large number of sensors that have limited computation, and communication capabilities. Under these resources, it is important to minimize the amount of consumption of energy to improve sensor lifetime [49].

A wireless sensor network is building in some dangerous places, especially in situations where it is not possible to replace sensor node batteries.

Importance of information has high priority when it is sent through the network. For protecting data, we need to apply some security procedures where security mechanisms address computing services (e.g. authentication, intrusion detection, etc.) and provide secure transaction. Since the sensor has limited battery life, power consumption is affected when applying cryptography algorithms

In some applications, sensor networks are deployed in a hostile environment, security becomes very important in this type of networks that face some malicious attacks. So to avoid this vulnerability, data that is sent over the network should be encrypted. Symmetric key scheme is more appropriate cryptography for wireless sensor networks due to its low energy consumption and simple hardware requirements, but most of them cannot provide sufficient security level (e.g. integrity, confidentiality, and authentication) as public key approach does.

Best solution to keep the information in security in wireless sensor networks is cryptography. Encryption algorithms and their use are an essential part of the secure transmission of information [43].

2.3.4.1/ SECURITY REQUIREMENTS

The security of wireless sensor networks can be divided into two categories: operational security and information security.

Operational security means the network should continue to work and do its functions if some of its components are attacked. The information security means the integrity and authenticity of information should always be protected, and confidential information should never be detected.

Confidentiality Confidentiality means keeping information secret from unauthorized parties. A sensor network should not leak sensor readings to neighboring networks. The confidentiality objective is required in sensors' environment to protect information traveling between the sensor nodes of the network or between the sensors and the base station from disclosure, since an adversary having the appropriate equipment may eavesdrop on the communication. By eavesdropping, the adversary could overhear critical information such as sensing data and routing information [8].

Authentication In a sensor network, an adversary can easily inject messages, so the receiver needs to make sure that the data used in any decision-making process originates from the correct source. As in conventional systems, authentication techniques verify the identity of the participants in a communication, distinguishing in this way legitimate users

from intruders. In the case of sensor networks, it is essential for each sensor node and base station to have the ability to verify that the data received was really sent by a trusted sender and not by an adversary that tricked legitimate nodes into accepting false data. If such a case happens and false data are supplied into the network, then its behaviour could not be predicted, and most of times the mission of wireless sensor networks will not be accomplished as expected. However, authentication for broadcast messages requires stronger trust assumptions on the network nodes [8].

Integrity Data integrity ensures the receiver that the received data is not altered in transit by an adversary. Lack of integrity could result in many problems since the consequences of using inaccurate information could be disastrous, for example, for the healthcare sector where lives are endangered. Integrity controls must be implemented to ensure that information is not altered in any unexpected way [8].

Availability Availability ensures that services and information can be accessed at the time they are required. In sensor networks there are many risks that could result in loss of availability such as sensor node capturing and denial of service attacks. The availability of a sensor and sensor network may decrease for the following reasons:

- Additional computation consumes additional energy. If no more energy exists, the data will no longer be available.
- Additional communication also consumes more energy.Besides, as communication power increases so does the chance of a communication conflict or interference.

2.3.4.2/ ATTACKS IN WSNS

A wireless sensor network is one of wireless networks type, so most of attack cases of wireless network will be effect also on wireless sensor network. Basically, attacks on wireless sensor networks can be classified into one or more of the following categories:

- Outsider vs. Insider attack: in an outsider attack, a malicious node harms the wireless sensor network without being part of it. In contrast, in an insider attack the malicious node harms the wireless sensor network as (authorized) participant of the wireless sensor network [36].
- Physical vs. Remote attack: in a physical attack an adversary physically accesses the sensor node that should be harmed by tampering or destroying the sensor's hardware. In contrast, a remote attack is implemented from a (large) distance, e.g. by emitting a high-energy signal to interrupt the communication [36].
- Passive vs. Active attack: in a passive attack an adversary just eavesdrops or monitors the communication within the wireless sensor network. In contrast, in an active attack the adversary directly influences the communication in the wireless sensor network by modifying, fabricating or suppressing data packets [36].
- Laptop-class vs. Mote-class attack: a mote-class attack is an attack against a wireless sensor network that is implemented from a mote, i.e. the attacking device is of

Algorithm	Key Size	Key exchange		Signature	
		Client	Server	Sign	Verify
RSA	1024	15.4	304	304	11.9
	2048	57.2	2302.7	2302.7	53.7
ECC	160	22.3	22.3	22.82	45.09
	224	60.4	60.4	61.54	121.98

Table 2.4: Energy cost for RSA and ECC (mJ) [81]

Algorithm	Key Size	Key exchange		Signature	
		Client	Server	Sign	Verify
RSA	1024	3.51	68.97	68.97	2.70
	2048	12.98	523.10	523.10	12.20
ECC	160	6.15	6.15	6.26	12.41
	224	16.62	16.62	16.93	33.55

Table 2.5: Energy cost for RSA and ECC (mJ) [53]

same type of hardware as the sensor nodes that should be attacked. In contrast, in a laptop-class attack, the adversary utilizes a device which is superior to the sensor nodes that should be attacked in terms of computational power and transmission power [36].

2.3.4.3/ Results about energy consumption

Wander et al [81] give some figures about the energy consumption of RSA and ECC for the same level of security while running on an Atmega128L processor. Table 2.4 (table 2 in [81]) show that RSA consumes more energy than ECC for the same level of security for both key echange and signature operations.

Piotrowski et al [53] give similar figures about energy consumption of RSA and ECC. The experiment was run on MICA family nodes (MICA2, MICAz and MICA2DOT) as well as on TelosB. Table 2.5 (table 7 in [53]) only focus on the results of TelosB nodes as they are based on MSP430. The results are quite similar to those of Wander et al, there is only a multiplicative factor of 4 to 5, probably due to the different processors. These results show again that RSA consumes more energy than ECC for the same level of security.

2.4/ Data Communication

In this section, we present general view on data communication, some definitions of principles of communication system. we present energy model of communication model.

2.4.1/ Overview of Data Communication

We present some principles about data communication.....

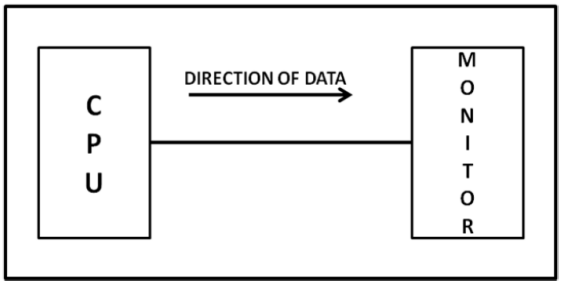

Figure 2.18: Simple data flow communication

2.4.1.1/ General definitions

Data communication is a process of exchanging data or information. In case of computer networks this exchange is done between two devices over a transmission medium, this process needs a communication system which is consist of hardware and software, the hardware part include three main components, the sender devices, receiver devices and the intermediate devices. The software part involves certain rules which specify how the devices will be communicating and when, what is the type of medium for pass the data, all these steps are called a *protocol* [26].

The effectiveness of any data communications system depends upon some points of quality communication system like:

- Delivery: the data should be delivered to the correct destination and correct user.
- Accuracy: the communication system should deliver the data accurately without any errors or lost because the data maybe get some change during transmission that maybe change the delivered data.
- Synchronization: the audio and video data must be delivered in a timely manner without any delay.

Most communication systems have main components:

- Message: it is the information to be communicated by the sender to the receiver.
- Sender: it is any device that is capable of sending the data (message).
- Receiver, it is a device that the sender wants to communicate the data (message), or to receive the data from sender.
- Transmission Medium, it is the path by which the message travels from sender to receiver, it can be wired or wireless and many subtypes in both [72].
- Protocol, it is an agreed upon set or rules used by the sender and receiver to communicate data or it is a set of rules that governs data communication

All communication systems has one of three main data flow: simple as in figure 2.18, half-duplex as in figure 2.19 and full-duplex as in figure 2.20.

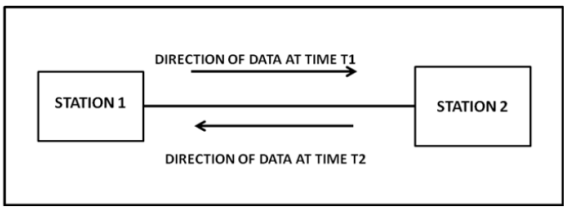

Figure 2.19: Half-duplex data flow communication

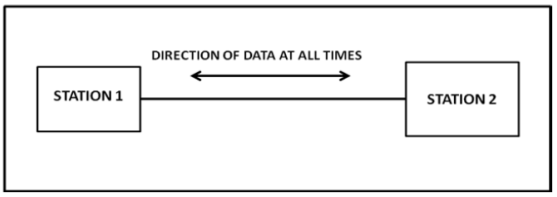

Figure 2.20: Full-duplex data flow communication

2.4.1.2/ Classification of Data Communication

Data communication can be classified according to many principles but here we will classify according to medium, there are two types of data communication (data transmission), wired and wireless transmission and each one has many advantages and disadvantages.

Wired transmission, it is the medium of pass the information between sender and receiver, wired medium like cables or optical fibre, and some example of wired communication are telephone networks, cable television and fibre-optic communication.

Wireless transmission, this type does not use the cables for passing the data through the network, but use wireless technology like radio communication, microwaves, wifi,...,etc.

2.4.2/ Energy Model for Communication in WSN

2.4.2.1/ Communication, energy and time

Communication is the most energy consuming task on a wireless node. So we have to take this step into account.

Before describing the energy model that we use in our experiments, we focus on the energy consumption in the network, i.e. once the message has been transmitted. We do not care about the energy of the routing of the message until the base station. Indeed, whatever the message and the source node, the route will be exactly the same whether the message is compressed and/or encrypted. So the routing protocol is not important for our evaluation of the energy consumption. The only parameter that can modify the energy on the network is the size of the message.

So from now on, we focus only on the energy on the source node and the size of the message at the output of the source node so that we can evaluate the total influence of compression and/or encryption on the energy consumption.

As for the delay, compressing and/or encrypting take some time. So there is a delay that

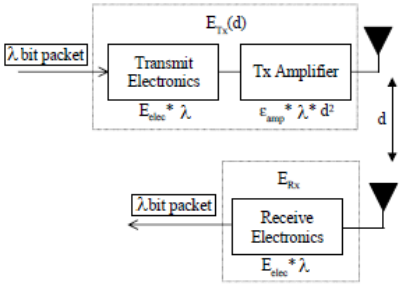

Figure 2.21: First Order Radio Model

can occur on the source node. We do not take into account the delay that can occur in the network, we assume the size of the message has a negligible influence on the delay in the network.

2.4.2.2/ The First Order Radio Model

In order to evaluate the energy for communication between several nodes, we use the First Order Radio Model [30]. The First Order Radio Model has been used first for evaluating the well-known LEACH protocol. It is based on a model of a radio transmission that is described on figure 2.21.

The assumption is that when a node sends data, it first needs to handle the λ-bit data with the transmit electronics. This first step has an energy cost of $\lambda \times E_0^{\text{trans}}$. Then, the signal is amplified according to the distance d between the sender and the receiver. This second step has an enegy cost of $\lambda \times \epsilon \times d^2$. So, for sending data, the energy model is:

$$E^{\text{trans}}(\lambda) = \lambda \times (E_0^{\text{trans}} + \epsilon \times d^2) \tag{2.1}$$

When a node receives data, the circuitry is simpler and does not depend on the distance. The energy model is:

$$E^{\text{recv}}(\lambda) = \lambda \times E_0^{\text{recv}} \tag{2.2}$$

For the numerical simulations, we choose the same numbers as [30]:

$$\begin{cases} E_0^{\text{trans}} = 50 \text{ nJ/bit} \\ E_0^{\text{recv}} = 50 \text{ nJ/bit} \\ \epsilon = 100 \text{ pJ/m}^2\text{/bit} \end{cases}$$

2.4.2.3/ Finer modelization of communication

In our experiments, we first omit the limited size of the messages and compute the energy globally. In the last part, we take into account the maximum size of the payload and the size of the header. We took some figures from TinyOS which has a header of 11 bytes

and a maximum payload size of 28 bytes by default. With this finer modelization of the communication, we can define several strategies for message optimizations in order to improve the energy consumption.

2.5/ SUMMARY

In end of this chapter, we present most of information that we need to our work. We collect best details about wireless sensor networks that we need to work on WSN with encryption and compression algorithms.

3

MODELLING ALGORITHMS ON A SINGLE NODE

Contents

In this chapter, we present and discuss our practical work by implement the code on sensor node to know our proposal that may give us good contribution. In section 3.1, we have done our experiment on two cryptography algorithms to calculate values of energy consumption for each algorithm. In section 3.2, we find the energy model for each cryptography algorithm. also present general model for encryption algorithms. In section 3.3, we present the model of time and model of memory for cryptography algorithms. In section 3.4, we present energy model for compression algorithms. In section 3.5, we present the models of time and memory for compression algorithms.

3.1/ EXPERIMENTS ON CRYPTOGRAPHIC ALGORITHMS

For doing our experiment, we choose Senslab platform for testing our source code. The encryption algorithms that implemented on single node is DES and AES. The source

code for encryption algorithm has been written by C language that uploaded on sensor node WSN430.

For getting secured message, the sensor node face problem with energy consumption. So by doing this experiment we can find the values of energy consumption in two encryption algorithms .

3.1.1/ EXPERIMENTS ON THE DES ALGORITHM

For doing experiments of an encryption process on a sensor node, we choose the DES algorithm. Algorithm 4 shows the general procedure that was used to test DES. The SETUP call is responsible for initializing the node. Then, the node is waiting for a signal that is sent by the user with the WAIT. The cryptographic code then starts in the RUN function until the end. When the node finishes, it is put in deep sleep with the SLEEP function. This procedure allows to observe an energy peak during the RUN function so that it is easy to compute the energy.

Another procedure we used removed the WAIT call. This way is more automatic but the setup time energy must be computed separately to be substracted from every results.

Algorithm 4 General procedure to test DES

```
function MAIN
    SETUP( )
    WAIT( )
    RUN( )
    SLEEP( )
end function
```

By using Senslab platform, we can connect with platform through internet for upload and run the code, also for getting the results of experiments. As we explained, DES is a symmetric key cryptographic algorithm based on a Feistel scheme with a 56-bit key and 16 rounds. Each round consists in a fixed set of four operations called `Expansion`, `KeyMixing`, `Substitution` and `Permutation` that operates on 64-bit blocks that are divided in two 32-bit half-blocks.

In our experiments, we use a custom implementation of DES in C that can be tuned to reduce the number of rounds, in these experiments we work on Senslab platfoem, in each experiment we upload the code to senslab server, compiler and run on some selected sensors. The main target of our experiment is measuring the power consumed by the DES algorithm on a MSP430-based node of the Senslab platform, and in our experiments we try to change the number of rounds and fix other factors affected, and obtained the results of consumed power for each scenario. The code is divided into some functions, each one doing special job.

For each experiment, we used random input data of $\lambda = 64$ bits. The number of rounds ranges over the values 2, 4, 8 and 16, for starting the encryption process we get by using keyboard. Each experiment is repeated five times.

In first experiment, figure 3.1 shows an example of the result of one experiment with 2 rounds. The encryption starts at $t_1 = 92.26$ s and finishes just before $t_2 = 94.26$ s. During encryption, the node consumes a little more than 9.78 μW Then, to compute the energy,

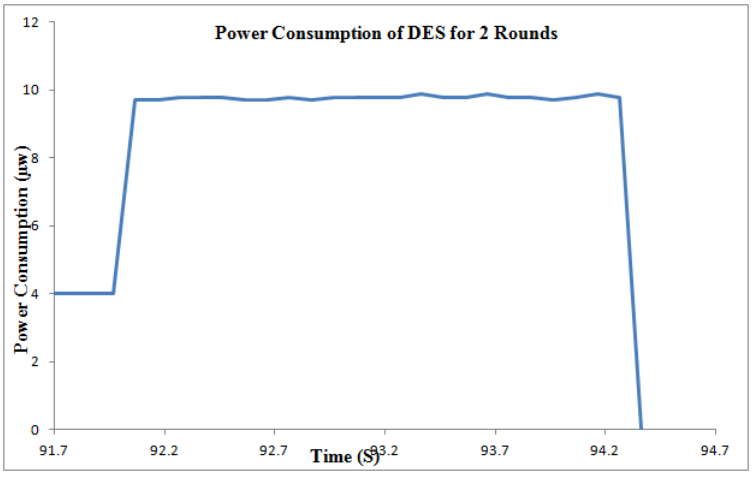

Figure 3.1: An experiment of energy consumption of DES with 2 rounds

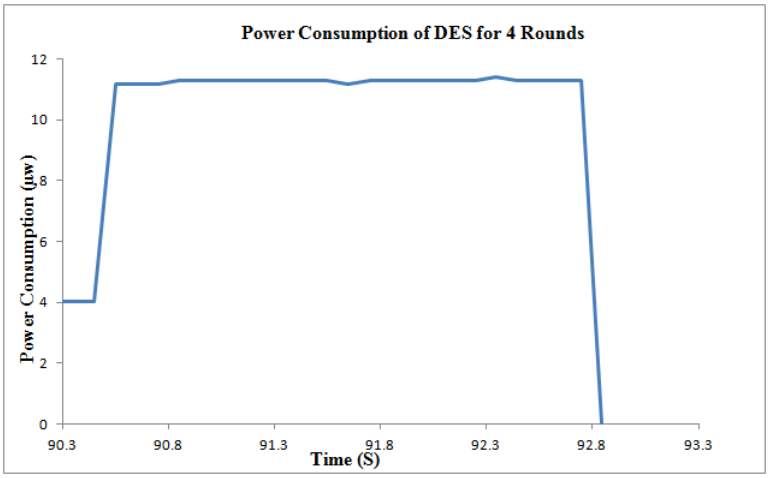

Figure 3.2: An experiment of energy consumption of DES with 4 rounds

we sum the power over time between t_1 and t_2. We use the areas of the consecutive trapezoids of height $\Delta t = 100$ ms thanks to the file of measures given by Senslab. And the computed energy consumption is 21.487 μJ.

In second experiment, figure 3.2 shows an example of the result of one experiment with 4 rounds. The encryption starts at $t_1 = 90.54$ s and finishes just before $t_2 = 92.74$ s. During encryption, the node consumes a little more than 11.28 μW. Then, the computed energy is 24.749 μJ.

In third experiment, figure 3.3 shows an example of the result of one experiment with 8 rounds. The encryption starts at $t_1 = 100.35$ s and finishes just before $t_2 = 103.74$ s. During encryption, the node consumes a little more than 9.78 μW Then, the computed energy is 33.266 μJ.

In fourth experiment, figure 3.4 shows an example of the result of one experiment with 16 rounds. The encryption starts at $t_1 = 96.36$ s and finishes just before $t_2 = 100.25$ s. During encryption, the node consumes a little more than 10.38 μW. Then, the computed energy is 40.551 μJ.

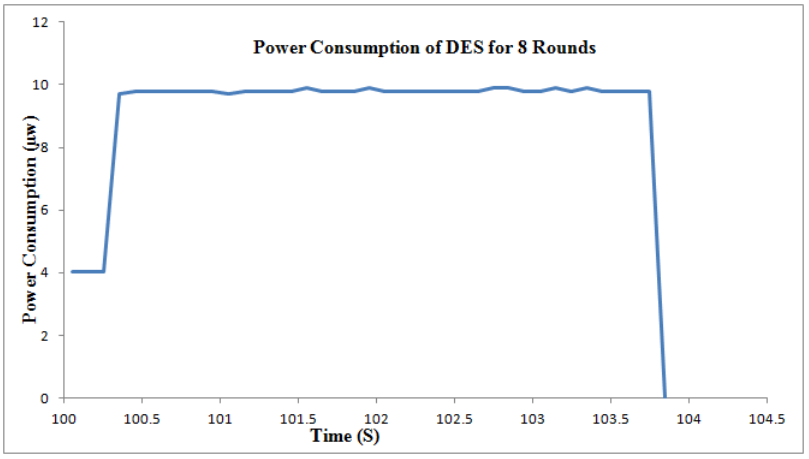

Figure 3.3: An experiment of energy consumption of DES with 8 rounds

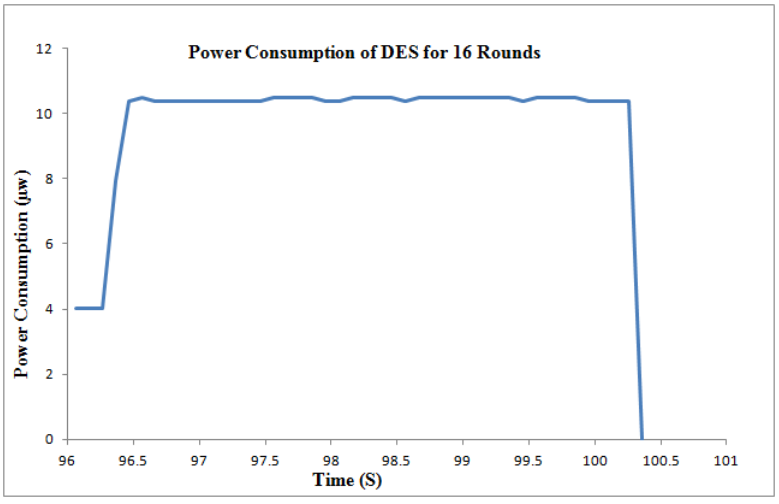

Figure 3.4: An experiment of energy consumption of DES with 16 rounds

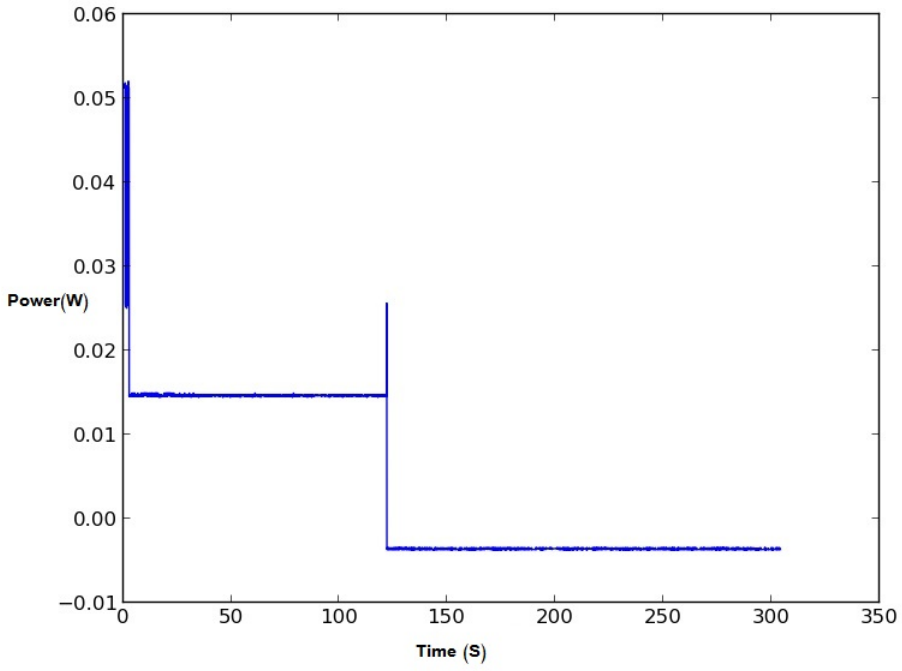

Figure 3.5: An experiment of energy consumption of AES

3.1.2/ EXPERIMENTS ON THE AES ALGORITHM

In this experiment, we work on the AES algorithm to study and check the value of consumed energy, AES has good level of security for protect our information, but maybe it has big challenge with consumed of energy in sensor networks. So to take good decision when we choose this algorithm for some application, we need to know the value of consumed energy with selected level of its security.

In this experiment, we choose a size of block of $\lambda = 128$ bits, and a size of key of 256 bits. The recommended number of rounds for 256 bit keys is 14. For each experiment we change the number of rounds to calculate consumed energy and repeat each one many times with different nodes on Senslab platform.

Figure 3.5 show an example of results for the AES algorithm.

3.2/ ENERGY MODEL FOR CRYPTOGRAPHIC ALGORITHMS

In this section we present energy model for DES and AES algorithms depending on previous results.

3.2.1/ ENERGY MODEL FOR THE DES ALGORITHM

Table 3.1 and figure 3.6 show an average of the results obtained from the previous experiments of DES algorithm with various number of rounds. We observe that the relation

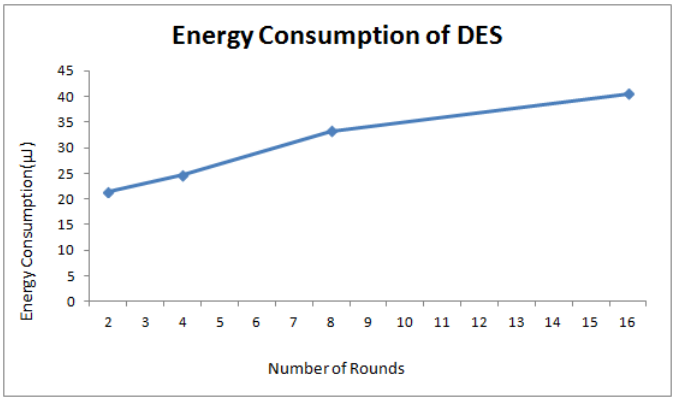

Figure 3.6: Energy consumption of the DES algorithm

Rounds	Energy consumption (μJ)
2	21.487
4	24.749
8	33.266
16	40.551

Table 3.1: Energy consumption of the DES algorithm

between energy consumption and the number of rounds is linear, the form is:

$$E = E_0 + r * E_r \tag{3.1}$$

where E_0 is a constant that represents the energy for constant operations in the algorithm, i.e. operations that do not depend on the number of rounds : initial and final permutation, permuted choice 1 (PC1) in the key schedule; r is the number of rounds and E_r is the energy per round.

Applying a linear regression on the data of table 3.1, we find:

$$\begin{cases} E_0 = 19.149\ \mu\text{J} \\ E_r = 1.348\ \mu\text{J/round} \end{cases}$$

The estimation of this linear regression is also shown on figure 3.6.

As a conclusion of this experiment, we found a linear relationship between energy consumption and the number of rounds for DES. It is of the same kind as the one found in [38] for RC5. These uncorrelated results can lead to a general model for encryption with symmetric cryptographic algorithm.

3.2.2/ Energy model for the AES algorithm

In table 3.2, we find the energy consumption for AES algorithm in different rounds. Figure 3.5 explain the relationship between consumed energy and number of rounds.

Applying a linear regression on the data of table 3.2, we find:

Rounds	Energy (μJ/bit)		
	min	average	max
3	3.29068	3.46390	3.66142
5	5.44577	5.75906	5.98765
10	10.60031	11.27303	11.62170
15	16.29904	17.23740	17.70424
20	21.53302	22.82470	23.53802
25	25.47813	28.26621	30.31364
30	32.29764	34.31705	35.23589

Table 3.2: Energy consumption for the AES algorithm

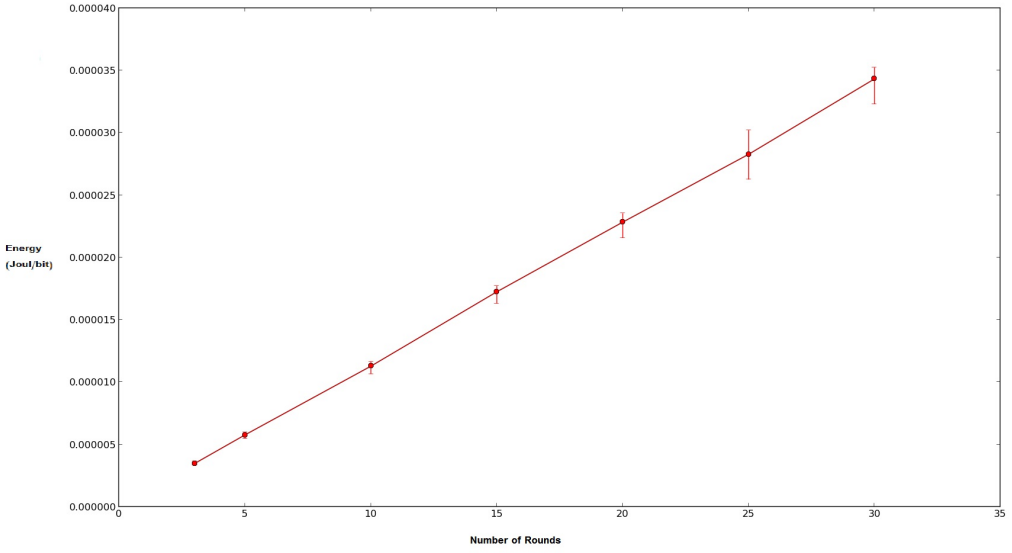

Figure 3.7: Energy consumption for the AES algorithm

Algorithm	$E_0^{enc}(nJ/bit)$	$E_r^{enc}(nJ/round/bit)$
DES	299.2	21.06
AES	2.69	1138.46
RC5 [38]	336.4	173.28

Table 3.3: Values of E_0^{enc} and E_r^{enc} for various encryption algorithms

$$\begin{cases} E_0 = 2.69 \text{ nJ/bit} \\ E_r = 1138.46 \text{ nJ/bit/round} \end{cases}$$

The estimation of this linear regression is also shown on figure 3.5. As a conclusion of this experiment, we found a linear relationship between energy consumption and the number of rounds for AES. It is of the same kind as for DES.

3.2.3/ General model for encryption algorithms

As seen before, we can model the energy consumption of encryption as:

$$E^{enc}(\lambda, r) = \lambda \times (E_0^{enc} + r \times E_r^{enc}) \tag{3.2}$$

In addition to equation 3.1, we introduce λ, the length of the input data, in our model. There should be another constant factor that does not depend on the length of the input data, but we assume this constant factor is negligible. For example, in DES, the only operation that does not depend on the number of rounds and the length of the input data is PC1, which is a very simple operation compared to the rest of the algorithm.

Table 3.3 shows the values of E_0^{enc} and E_r^{enc} for various algorithms. The DES, AES values are computed from our experiments. The RC5 values are computed from figure 3.8 that was deduced from [38]. We took the values of the TelosB node as it is a MSP430-based node, and we summed the ”setup” phase and ”encryption” phase to have the full algorithm.

Figure 3.8 shows the energy consumption of RC5 that was computed from [38] and the estimation that we did for this algorithm. RC5, AES and DES have a similar constant term E_0^{enc} whereas the energy per round E_r^{enc} is much higher for RC5, AES than for DES with a factor greater than 8. As all algorithms are based on the same basic bitwise operations, this difference can be explained by the implementation and the quality of the measures.

The important point is that table 3.3 gives us a good idea of the order of energy consumption of a symmetric cryptography algorithm.

3.3/ Modelling time and memory for cryptographic algorithms

As we know, the sensor node has small memory and also limited energy, for doing its duty it needs to know every things about all cost of all operations that it will do. In normal case the sensor has enough memory and fixed time for doing its job, but when we add

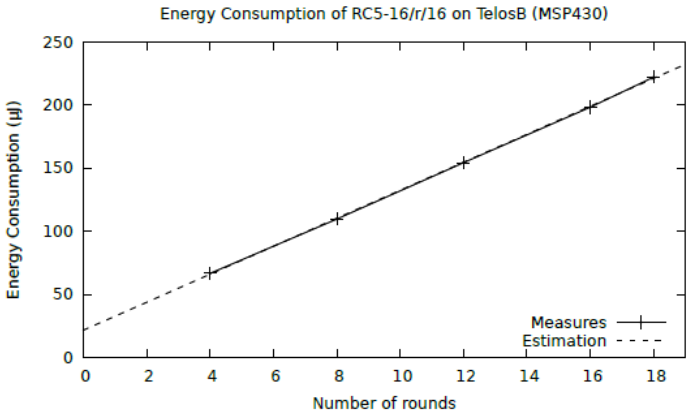

Figure 3.8: Energy consumption of the RC5 algorithm [38]

Rounds	Time (ms/bit)
2	35.9375
4	35.9375
8	53.0468
16	60.8468

Table 3.4: Time for the DES algorithm

encryption process to his work, it needs to know how long it need to perform extra operation, and also need to know the size of memory that it needs to complete this operation. Now we have energy consumption model for two encryption algorithms (DES, AES).

3.3.1/ MODEL FOR TIME

Like energy, we can find a linear model for the time taken by cryptographic algorithms.

$$T^{\text{enc}}(\lambda, r) = \lambda \times (T_0^{\text{enc}} + r \times T_r^{\text{enc}}) \tag{3.3}$$

Table 3.4 gives the time measures for the DES experiments. With a linear regression, we obtain:

$$\begin{cases} T_0 = 32.07535 \text{ ms/bit} \\ T_r = 1.91549 \text{ ms/bit/round} \end{cases}$$

Table 3.5 gives the time measures for the AES experiments. With a linear regression, we obtain:

$$\begin{cases} T_0 = 0.478 \text{ ms/bit} \\ T_r = 73.15 \text{ ms/bit/round} \end{cases}$$

Table 3.6 gives a summary for the time modelling of DES and AES.

Rounds	Time (s/bit)		
	min	average	max
3	0.21437	0.22372	0.23226
5	0.36310	0.37275	0.38185
10	0.72023	0.72906	0.74026
15	1.10677	1.11608	1.12638
20	1.47200	1.47844	1.48777
25	1.78780	1.79153	1.79419
30	2.21482	2.22259	2.23059

Table 3.5: Time for the AES algorithm

Algorithm	$T_0^{\text{enc}}(ms/bit)$	$T_r^{\text{enc}}(ms/round/bit)$
DES	32.07535	1.91549
AES	0.478	73.15

Table 3.6: Values of T_0^{enc} and T_r^{enc} for various encryption algorithms

3.3.2/ MODEL FOR MEMORY

For evaluation the impact on memory of cryptographic algorithms, we propose to look at the whole process. The algorithm itself generally use a constant amount of memory (that is the case for DES, AES and RC5). The only variable part for memory is input and output. So, we examine what happens at input and output, linked with the time model proposed before.

We assume that the original data is generated with a rate of λ bits every δ seconds. The data generation is not continuous but discrete: every δ seconds, a block of λ bits is generated for the algorithm input.

There is a necessary condition for the algorithm to handle such an amount of data: the time for processing λ bits of input data must not exceed δ seconds. Otherwise, there is a need for an input buffer that will fill until it's full. This condition is:

$$\delta \geq T^{\text{enc}}(\lambda, r) = \lambda \times (T_0^{\text{enc}} + r \times T_r^{\text{enc}}) \tag{3.4}$$

We see that a solution for this condition to happen is to reduce r the number of rounds. More precisely, there is an upper limit r_{lim} on the number of rounds that depends of the input data generation rate. More precisely:

$$r \leq \frac{\frac{\delta}{\lambda} - T_0^{\text{enc}}}{T_r^{\text{enc}}} = r_{\text{lim}} \tag{3.5}$$

Table 3.7 shows the values of r_{lim} for DES and AES with different data rate. Here, the data rate is $\frac{\lambda}{\delta}$. We see that DES supports quite high data rate up to 16 bit/s while AES supports only low data rate under 1 bit/s. This difference is due to the nature of the algorithm, AES has more computations to do in a small amount of time while DES computations are easier.

Data rate (bit/s)	r_{lim} for DES	r_{lim} for AES
0.5	1027	27
1	505	13
2	244	6
4	113	3
8	48	1
16	15	–
32	–	–

Table 3.7: Limit for the number of rounds with encryption only

Algorithm	E_0^{comp} (nJ/bit)	α
RLE	1.325	17%
S-LZW	5.6	53%
K-RLE	2.575	56%

Table 3.8: Values of E_0^{comp} and compression ratio α for various compression algorithms [17]

3.4/ ENERGY MODEL FOR COMPRESSION ALGORITHMS

3.4.1/ FIRST RESULTS

In this section, we rely on the results given in [17]. This paper describes an experiment using temperatures from three different locations and compressing them with three algorithms: RLE, K-RLE (with $K = 2$) and S-LZW.

We assume that the energy consumption for compression algorithms is only proportional to the length of the input data. Generally, compression algorithms do not have a setup phase, they only take decisions according to the input data.

$$E^{\text{comp}}(\lambda) = \lambda \times E_0^{\text{comp}} \tag{3.6}$$

Table 3.8 shows the values of E_0^{comp} and the compression ratio α for those three algorithms: S-LZW , RLE and K-RLE. S-LZW and RLE are lossless compression algorithms while K-RLE is a lossly compression algorithm.

3.4.2/ COMPLEMENTARY EXPERIMENTS

We made complementary experiments to confirm these numbers and to have some figures about time. Table 3.9 shows our own results on a Senslab node with the various algorithms. These results are quite different from the results of [17]. First, the ratio are different but that comes from the data set that was used in both experiment. We used a random sequence where each item had a difference between -2 et 2 with the previous item. That's why 2-RLE has such a good performance.

For the energy consumption of the algorithms, RLE and K-RLE are in the same order as the experiments of [17], a few nJ/bit. But we measured a much higher consumption for S-LZW. We think that there was a problem with our experiment and with the code of S-LZW

Algorithm	E_0^{comp} (nJ/bit)	α
RLE	7.50	5.4%
S-LZW	89.67	29%
1-RLE	2.86	68%
2-RLE	1.99	86%

Table 3.9: Values of E_0^{comp} and compression ratio α for various compression algorithms

Algorithm	T_0^{comp} (ns/bit)
RLE	447.48
S-LZW	5311.39
1-RLE	175.91
2-RLE	123.41

Table 3.10: Values of T_0^{comp} for various compression algorithms

that we downloaded from the original website, because we were not able to decompress the compressed data and we could not explain why.

As a conclusion of these complementary experiments, we decided to keep the results of [17] for our energy models because they may be more accurate than ours.

3.5/ MODELLING TIME FOR COMPRESSION ALGORITHMS

In this section, we develop a time model for compression. We used our complementary experiments because the results from [17] did not include time measures.

Like energy, we assume that the time for compression is proportional to the length of the input data. So, we modelize the time for compression with the following equation:

$$T^{comp}(\lambda) = \lambda \times T_0^{comp} \tag{3.7}$$

Table 3.10 shows the value of T_0^{comp} for various compression algorithms. Once again, the value for S-LZW appears to be very high compared to the others. But we assume that the value for RLE has the good order of magnitude. We will use this value later in our experiments.

3.6/ SUMMARY

We found a model for the energy consumption of encryption algorithms DES and AES. This model is defined by equation 3.2. It shows different initial values of energy and values of energy over number of rounds. We used the Senslab platform to calculate the energy consumed for each encryption algorithm, it help us to get results on real sensor node. Memory size of sensor node compute by special model for encryption algorithm, also we calculate time delay for encryption algorithm with different number of rounds in equation 3.3. At last we find energy model of compression algorithm in equation 3.6, and calculate a time model for compression algorithm.

4

ENERGY-FREE SECURITY

Contents

In this chapter, we compute the available maximum level of security in wireless sensor network. In section 4.1, we present how find the maximum number of rounds in different scenarios, also how to get energy free security. In section 4.2, we present energy model of linear network, and analysis its results. In section 4.3, we check maximum number of rounds in random network with different scenarios by mix encryption and compression algorithms. In section 4.4, we Summarizes this chapter in view lines.

4.1/ ANALYSIS OF DIFFERENT SCENARIOS WITH COMPRESSION AND ENCRYPTION

In this section, we give a full image for what we do on data from obtaining up to sending. We have four scenarios to examine the best case of them by using some of operations on data (compression, encryption and sending). We work on each scenario and calculate the energy consumption of node by using models of encryption as in equation 3.3, compression as in equation 3.6 and communication as in equations 2.1 and 2.2.

- *Scenario T*: in this scenario, we only consider the transmission of λ bits of input data.
- *Scenario CT*: in this scenario, we consider the compression of λ bits of input data with a compression ratio of α that is then transmitted.

- *Scenario ET*: in this scenario, we consider the encryption of λ bits of input data that is then transmitted.
- *Scenario CET*: in this scenario, we consider the compression of λ bits of input data with a compression ratio of α that is then encrypted and transmitted.

There is no need for a fifth scenario with encryption followed by compression and then by transmission as it would consume more energy than scenario CET.

Our goal is to show that scenario CET can consume as much energy as scenario T which would provide energy-free security.

4.1.1/ ENERGY FOR THE DIFFERENT SCENARIOS

Depending on the models of calculating the energy for each operation, we can get the models for two or three operations together.

The energy consumption for scenario T is given by:

$$\begin{aligned} E^{\mathsf{T}}(\lambda) &= E^{\mathsf{trans}}(\lambda) \\ &= \lambda \times (E_0^{\mathsf{trans}} + \epsilon \times d^2) \end{aligned} \tag{4.1}$$

It means the message will be sent without doing any change on the original information. We note the energy will be affected by the distance.

The energy consumption for scenario CT is given by:

$$\begin{aligned} E^{\mathsf{CT}}(\lambda) &= E^{\mathsf{comp}}(\lambda) + E^{\mathsf{trans}}((1-\alpha) \times \lambda) \\ &= \lambda \times (E_0^{\mathsf{comp}} + (1-\alpha) \times (E_0^{\mathsf{trans}} + \epsilon \times d^2)) \end{aligned} \tag{4.2}$$

The energy consumption for scenario ET is given by:

$$\begin{aligned} E^{\mathsf{ET}}(\lambda, r) &= E^{\mathsf{enc}}(\lambda, r) + E^{\mathsf{trans}}(\lambda) \\ &= \lambda \times (E_0^{\mathsf{enc}} + r \times E_r^{\mathsf{enc}} + E_0^{\mathsf{trans}} + \epsilon \times d^2) \end{aligned} \tag{4.3}$$

The energy consumption for scenario CET is given by:

$$\begin{aligned} E^{\mathsf{CET}}(\lambda, r) &= E^{\mathsf{comp}}(\lambda) + E^{\mathsf{enc}}((1-\alpha).\lambda, r) + E^{\mathsf{trans}}((1-\alpha).\lambda) \\ &= \lambda \times (E_0^{\mathsf{comp}} + (1-\alpha) \times (E_0^{\mathsf{enc}} + r \times E_r^{\mathsf{enc}} + E_0^{\mathsf{trans}} + \epsilon \times d^2)) \end{aligned} \tag{4.4}$$

Table 4.1 shows the energy consumptions with $\lambda = 64$ bits, $r = 8$ rounds and $d = 25$m in the different scenarios. $\lambda = 64$ bits is a typical size for a physical scalar data like temperature. $r = 8$ rounds is rather weak for DES and RC5. $d = 25$m is a typical distance in wireless sensor networks. We observe that, as expected, with any choice of algorithms, scenario CT consumes less energy than scenario T that consumes less energy than scenario CET that consumes less energy than scenario ET.

Table 4.2 shows the energy consumptions with $\lambda = 64$ bits, $r = 16$ rounds and $d = 75$m in the different scenarios. In this case, the number of rounds is $r = 16$, which is the maximum for DES and which is nearly the recommended number of rounds for RC5. The distance has been extended to 75m. We note that the energy consumption of scenario CET is less

Algorithms	E^{T}	E^{CT}	E^{ET}	E^{CET}
DES + RLE	7.200	6.061	37.132	30.904
DES + S-LZW	7.200	3.742	37.132	17.810
DES + K-RLE	7.200	3.333	37.132	16.503
RC5 + RLE	7.200	6.061	117.449	97.567
RC5 + S-LZW	7.200	3.742	117.449	55.559
RC5 + K-RLE	7.200	3.333	117.449	51.842

Table 4.1: Energy consumptions (in μJ) with $\lambda = 64$ bits, $r = 8$ rounds and $d = 25$m

Algorithms	E^{T}	E^{CT}	E^{ET}	E^{CET}
DES + RLE	39.200	32.621	79.914	66.414
DES + S-LZW	**39.200**	18.782	79.914	**37.918**
DES + K-RLE	**39.200**	17.413	79.914	**35.327**
RC5 + RLE	39.200	32.621	238.168	197.765
RC5 + S-LZW	39.200	18.782	238.168	112.298
RC5 + K-RLE	39.200	17.413	238.168	104.959

Table 4.2: Energy consumptions (in μJ) with $\lambda = 64$ bits, $r = 16$ rounds and $d = 75$m

than the energy consumption of scenario T, with the DES algorithm combined with S-LZW or K-RLE. In the other case, the compression algorithm does not have a good enough compression ratio (RLE), or the encryption algorithm consumes so much that it cannot be counterbalanced by compression (RC5). These figures, with realistic parameters, show that it is possible to have energy-free security but we need a more precise condition.

4.1.2/ ENERGY-FREE SECURITY

In this section, we try to precise the previous result, i.e. we try to compute the maximum number of rounds $r_{\max}$ for various values of d and the given compression algorithms. The following theorem gives the computation of $r_{\max}$:

Theorem [1]. *Security is free if and only if:*

$$r \leq \underbrace{\frac{\alpha \times (E_0^{trans} + \epsilon \times d^2) - (1-\alpha) \times E_0^{enc} - E_0^{comp}}{(1-\alpha) \times E_r^{enc}}}_{=r_{\max}(\alpha,d)}$$

The proof is straightforward, it directly comes from equations 4.1 and 4.4 with the condition that $E^{\mathsf{CET}}(\lambda, r) \leq E^{\mathsf{T}}(\lambda)$. We note that the condition does not depend anymore on the length of the data which is normal because, in each scenario, the energy is proportional to the length of the data.

Table 4.3 shows $r_{\max}(\alpha, d)$ for the three compression algorithms and the DES algorithm, with d varying from 25m to 100m. We observe that for distance of 25m, security can not be free whatever the compression algorithm is, i.e. $r_{\max}(\alpha, d) < 0$. The RLE algorithm do not compress enough and can never provide free security.

The results for S-LZW and K-RLE are quite close. For a distance of 50m, the maximum number of rounds is 1 and 3 respectively, which provides no security at all as there exists

Algorithms	25m	50m	75m	100m
RLE	–	–	–	–
S-LZW	–	1.291	18.024	41.450
K-RLE	–	3.645	22.531	48.970

Table 4.3: $r_{\max}(\alpha, d)$ with the DES algorithm

Algorithms	25m	50m	75m	100m
RLE	–	–	–	–
S-LZW	–	–	1.976	4.823
K-RLE	–	0.228	2.524	5.737

Table 4.4: $r_{\max}(\alpha, d)$ with the RC5 algorithm

some easy known attacks on DES. For a distance of 75m, as already seen, the full DES with 16 rounds can be used for free with both compression algorithms. For a distance of 100m, Triple-DES, that has 48 rounds and provides strong security, can be used for free in the case of K-RLE.

Table 4.4 shows $r_{\max}(\alpha, d)$ for the three compression algorithms and the RC5 algorithm, with d varying from 25m to 100m. RC5 consumes more energy than DES and the results for RC5 are not very good. Even for a distance of 100m, the maximum number of rounds is 4 and 5 for S-LZW and K-RLE respectively, which does not provide any security. The solution in this case is to optimize the implementation of RC5 or to find a better compression algorithm.

4.2/ Analysis with a linear network

In this section, we try to improve the previous results considering a linear network, as shown in figure 4.1. Our idea is that the energy consumed on the sending node can be counterbalanced globally over the network by the savings of the other nodes, due to the size of the compressed data. This could improve the security of the data while still competing with scenario T.

4.2.1/ Model

We use a linear network of $n+1$ nodes, the first node that generates data and n relays in the multi-hop communication to the base station. Each node only communicates with its closest neighbors that are at distance d. The last node communicates with the base station that is at distance d too. Figure 4.1 shows an example of linear network.

On this linear network, we examine the global energy in scenario T and scenario CET. In each scenario, the data is sent by the first node, then received n times and transmitted n times until the base station.

The energy consumption for scenario T with a linear network is given by:

$$\begin{aligned} E^{\mathsf{T}}_{\mathsf{net}}(\lambda, n) &= E^{\mathsf{trans}}(\lambda) + n \times (E^{\mathsf{recv}}(\lambda) + E^{\mathsf{trans}}(\lambda)) \\ &= \lambda \times (n \times E^{\mathsf{recv}}_0 + (n+1) \times (E^{\mathsf{trans}}_0 + \epsilon \times d^2)) \end{aligned} \tag{4.5}$$

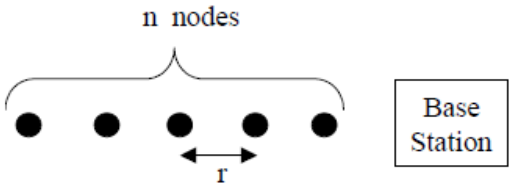

Figure 4.1: A linear network with $n+1$ nodes and a base station (BS)

Algorithms	$n=1$	$n=2$	$n=5$	$n=10$	$n=15$
RLE	–	–	–	2.615	10.517
S-LZW	–	8.653	34.756	78.262	121.767
K-RLE	2.134	11.955	41.416	90.518	139.620

Table 4.5: $r_{\max}^{\text{net}}(\alpha, 25, n)$ with the DES algorithm and a linear network

The energy consumption for scenario CET with a linear network is given by:

$$\begin{aligned} E_{\text{net}}^{\text{CET}}(\lambda, r, n) &= E^{\text{CET}}(\lambda, r) + n \times (E^{\text{recv}}((1-\alpha).\lambda) + E^{\text{trans}}((1-\alpha).\lambda)) \\ &= \lambda \times (E_0^{\text{comp}} + (1-\alpha) \times (E_0^{\text{enc}} + r \times E_r^{\text{enc}} + n \times E_0^{\text{recv}} + \\ &\quad (n+1) \times (E_0^{\text{trans}} + \epsilon \times d^2))) \end{aligned} \tag{4.6}$$

Now we can state an extension of theorem 1 for a linear network and compute $r_{\max}^{\text{net}}(\alpha, d, n)$:

Theorem [2]. *Security is free in a linear network of $n+1$ nodes if and only if:*

$$r \leq \underbrace{\frac{n.\alpha.E_0^{recv} + (n+1).\alpha.(E_0^{trans} + \epsilon.d^2) - (1-\alpha).E_0^{enc} - E_0^{comp}}{(1-\alpha) \times E_r^{enc}}}_{=r_{\max}^{net}(\alpha,d,n)}$$

4.2.2/ RESULTS

Table 4.5 shows $r_{\max}^{\text{net}}(\alpha, d, n)$ for the three compression algorithms and the DES algorithm, with $d = 25$m and n varying from 1 to 15. We observe that with only one-hop before the base station, security is free with the K-RLE compression algorithm, even if the number of rounds provides very weak security. For $n \geq 5$, the maximum number of rounds for S-LZW and K-RLE exceeds the number of round for DES, and for $n \geq 10$, it exceeds the number of rounds for Triple-DES. This shows that very strong security can be achieved for free on a wide network.

Table 4.6 shows $r_{\max}^{\text{net}}(\alpha, d, n)$ for the three compression algorithms and the RC5 algorithm, with $d = 25$m and n varying from 1 to 15. In this case, the situation is better than in the experiment with a single node, but the level of security is not as strong as the level for DES. For $n \geq 15$, the maximum number of rounds is 14 and 16 for S-LZW and K-RLE respectively, which a little less than the recommended 18-20 rounds for good security. Once again, the implementation of the algorithm must be improved in order to achieve better results.

Algorithms	$n = 1$	$n = 2$	$n = 5$	$n = 10$	$n = 15$
RLE	–	–	–	0.103	1.064
S-LZW	–	0.837	4.010	9.297	14.585
K-RLE	0.045	1.238	4.819	10.787	16.754

Table 4.6: $r_{\max}^{\text{net}}(\alpha, 25, n)$ with the RC5 algorithm and a linear network

4.3/ ANALYSIS WITH RANDOM NETWORKS

In this section, we compute $r_{\max}$ on random networks. We focus on DES and K-RLE as it's the best combination of a compression algorithm and an encryption algorithm that we have.

4.3.1/ SIMULATIONS

The difficulty in this experiment is to choose an application to make the measures. We decided to test a simple routing application on a square area of width w. The network is composed of n nodes uniformly distributed on the area. Two nodes can communicate if their distance is less than $0.4 \times w$ so that there is, on average, half of the nodes in the neighborhood of each node, whatever the width of the area.

Among the n nodes, 20 nodes are chosen to be sources of messages of size $\lambda = 64$ bits. Those messages are routed to a sink which is placed at the coordinates $(0.9 \times w, 0.9 \times w)$ thanks to a shortest path algorithm which takes into account the square of the distance between each node (as the energy for transmitting a message is proportional to the square of the distance). Then, each source sends a message to the sink along the chosen path.

To compute $r_{\max}$ for a given network, we first compute the energy E_T that is consumed for scenario T. Then, we compute the energy that is consumed for scenario CET with $r = 0$ rounds, which is necessarily less than the E_T (scenario CET with $r = 0$ rounds is similar to scenario CT). Then, the number of rounds is increased until the energy is more that E_T which means we have reached $r_{\max}$.

This experiment is repeated on 1000 different random network for each value of (n, w).

4.3.2/ RESULTS WITH DES AND K-RLE

Figure 4.2 shows the distribution of $r_{\max}$ for $n = 50$ nodes and $w = 50$ m. We observe that this distribution is not symmetric and is quite wide so that the computation of an average is not very relevant. That's why we decided to compute the first decile of the measures, i.e. the value of $r_{\max}$ that divide the data set in 10% of low values and 90% of high values. In this case, the first decile is 45 which means that for a random network with our simple routing application, taking DES with 45 rounds (nearly Triple DES) and K-RLE is energy-free with a probability of 0.9.

This result is not a surprise as it can be compared to the results of table 4.5. The range of the network is a little shorter in the case of random networks (20 m), but the average number of hops from the sources to the sink must be high enough so that the energy for

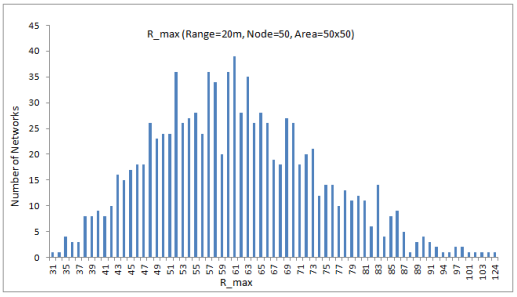

Figure 4.2: Distribution of $r_{\max}$ for $n = 50$ nodes and $w = 50$ m for DES/K-RLE

n \ w	50m	100m	200m	300m	400m
50	45	52	72	102	141
100	76	84	97	119	148
150	98	108	121	138	163
200	101	125	137	152	173

Table 4.7: First decile of $r_{\max}$ for DES/K-RLE

encryption is counterbalanced by the savings along the paths.

Table 4.7 shows the computation of the first decile for many values of (n, w). This table shows that in any case, it is possible to have strong energy-free security on a random network. We observe that, for a fixed w, $r_{\max}$ increases sub-linearly w.r.t. n. Adding more nodes on the area make paths shorter, but not short enough so that the gain in energy can bring many more rounds. We also observe that for a fixed n, $r_{\max}$ increases over-linearly w.r.t. w. In this case, the distances between nodes is increased and the gain in energy can be used to do many more rounds.

4.3.3/ RESULTS WITH OTHER CONFIGURATIONS

In this part, we present all possible scenarios that mix two algorithms one compression with another encryption.

4.3.3.1/ RESULTS WITH RC5 AND K-RLE

Figure 4.3 shows the distribution of $r_{\max}$ for RC5 and K-RLE with $n = 50$ nodes and $w = 50$ m. We note that values of $r_{\max}$ are between 6 and 21, it means the range of possibility of $r_{\max}$ is short, so that the computation of an average is not fixed. By same way of calculating(the first decile of measures), the first decile is 6. This value is approximately same value in table(4.6).

Table 4.8 shows the result of the first decile for $r_{\max}$ for RC5 and K-RLE. Also we note $r_{\max}$ is increasing when the distances between nodes is increasing, similarly when number of nodes in increasing, but this augmentation not high.

In this experiment, we can note the maximum number of rounds that we can obtain in case of RC5 as cryptography algorithms with K-RLE as compression algorithms is low

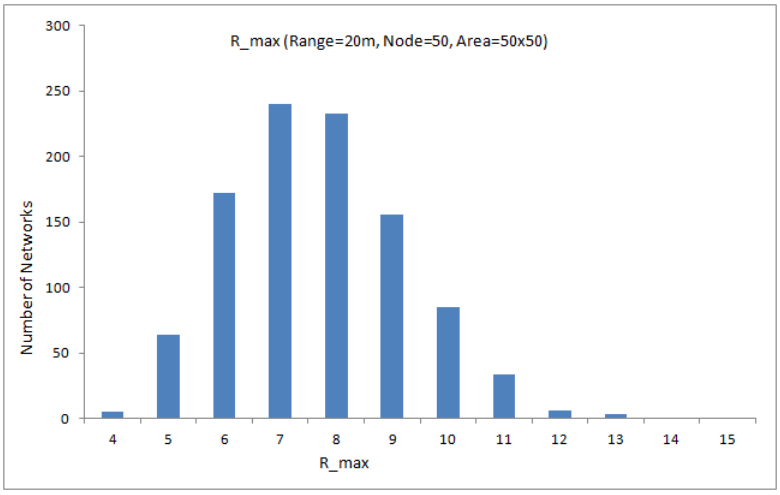

Figure 4.3: Distribution of $r_{\max}$ for $n = 50$ nodes and $w = 50$ m for RC5/K-RLE

n \ w	50m	100M	200m	300m	400m
50	6	7	9	13	17
100	9	10	12	15	18
150	12	13	15	17	20
200	14	15	17	19	21

Table 4.8: First decile of $r_{\max}$ for RC5/K-RLE

more than case of change the cryptography algorithm to DES. also the change of area has not big difference at fix the number of nodes, we can say the distance has low effect more than type of cryptography algorithm. Similarly increasing the number of nodes has little change for increasing level of security.

All results in random network, it seem as in linear network with little difference. So it proofs RC5 with compression algorithm provide low level of security in two different topologies.

4.3.3.2/ RESULTS WITH DES AND S-LZW

Figure 4.4 shows the distribution of $r_{\max}$ for DES and S-LZW with $n = 50$ nodes and $w = 50$ m. By change our selected input $r_{\max}$ is changed between 38 upto 151 which means Probability of find $r_{\max}$ is high. for computation an average of $r_{\max}$ is done by choose first decile that is approximately 38, compare to table (4.5) the values are nearly same

Table 4.9 shows the result of the first decile for $r_{\max}$ for DES and S-LZW. the values of $r_{\max}$ is changed when the distances and number of hops are changed(positive relation)

In this experiment, by fix cryptography algorithm DES and change compression algorithm S-LZW. The values of number of rounds are near as in case of DES and K-RLE. Also we note the distance effect the level of security with DES, by increasing the number of nodes in same network can improve level of security.

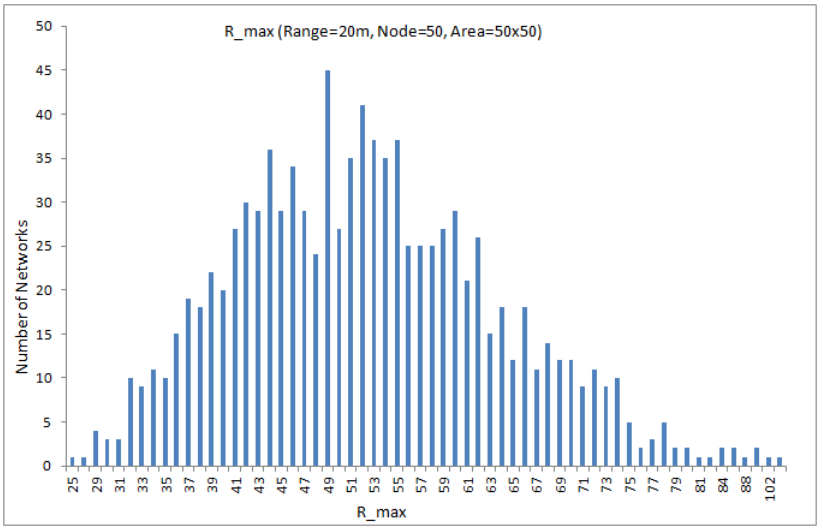

Figure 4.4: Distribution of $r_{\max}$ for $n = 50$ nodes and $w = 50$ m for DES/S-LZW

n \ w	50m	100M	200m	300m	400m
50	38	44	62	88	123
100	66	72	84	103	129
150	85	93	106	120	142
200	100	109	120	132	151

Table 4.9: First decile of $r_{\max}$ for DES/S-LZW

4.3.3.3/ RESULTS WITH RC5 AND S-LZW

Figure 4.5 shows the distribution of $r_{\max}$ for RC5 and S-LZW with $n = 50$ nodes and $w = 50$ m. In this case the value of $r_{\max}$ is particular between two values 5,19, so the $r_{\max}$ is very short and small around 5. Table(4.6) has nearly value in linear network.
Table 4.10 shows the result of the first decile for $r_{\max}$ for RC5 and S-LZW. the effectiveness of change the distances and number of nodes is small.

In this experiment, we same values of nodes and distances with two algorithms of cryptography and compression, we note the $r_{\max}$ is low. it means by this combination of compression and algorithms can not give us good level of security. also when change the number of nodes or to increasing the distance, it is same the changed range is limited. So in this case if we are looking to get good security, we can say it is not good for all applications that used important data.

n \ w	50m	100M	200m	300m	400m
50	5	6	8	11	15
100	8	9	10	13	16
150	11	12	13	15	18
200	12	13	15	16	19

Table 4.10: First decile of $r_{\max}$ for RC5/S-LZW

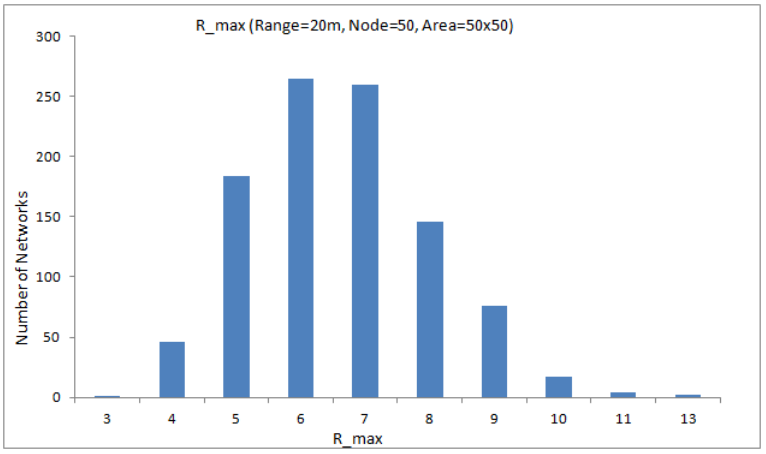

Figure 4.5: Distribution of $r_{\max}$ for $n = 50$ nodes and $w = 50$ m for RC5/S-LZW

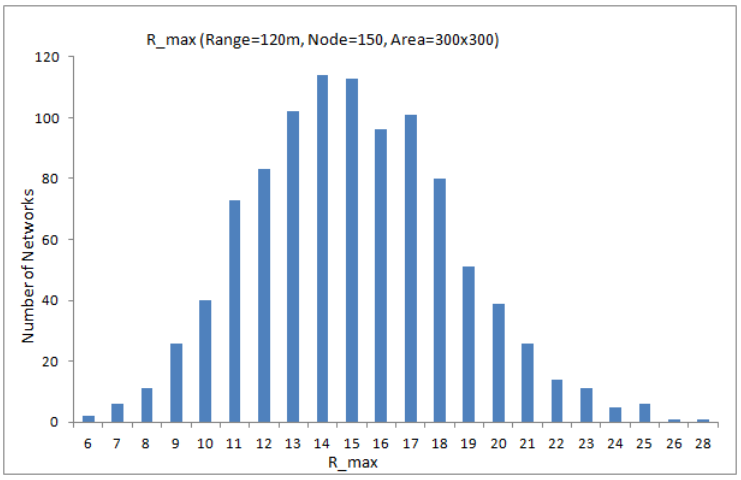

Figure 4.6: Distribution of $r_{\max}$ for $n = 150$ nodes and $w = 300$ m for DES/RLE

4.3.3.4/ RESULTS WITH DES AND RLE

Figure 4.6 shows the distribution of $r_{\max}$ for DES and RLE with $n = 150$ nodes and $w = 300$ m. By choose RLE algorithm with DES, we found $r_{\max}$ is zero in short distance with low number of rounds as in table (4.5). So to get energy-free security we increasing the distances between the hops. The Probability of high security is small.
Table 4.11 shows the result of the first decile for $r_{\max}$ for DES and RLE. The $r_{\max}$ has small values and the range of its change also is limited.

In this experiment, increasing the values of nodes and changing the distances has little effect on $r_{\max}$. also in this case the possibilities of get suitable level of security is so low that can give us general vision this combination of algorithms of cryptography and compression is not useful.

n \ w	50m	100M	200m	300m	400m
50	0	0	0	5	11
100	1	2	4	8	12
150	4	6	8	11	15
200	7	9	11	13	16

Table 4.11: First decile of $r_{\max}$ for DES/RLE

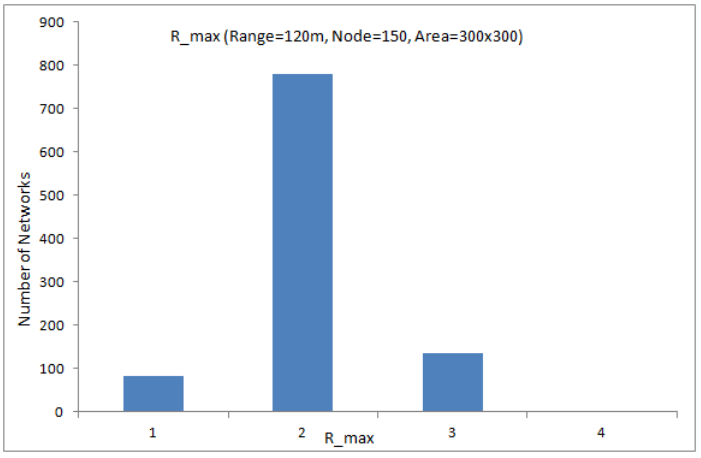

Figure 4.7: Distribution of $r_{\max}$ for $n = 150$ nodes and $w = 300$ m for RC5/RLE

4.3.3.5/ RESULTS WITH RC5 AND RLE

Figure 4.7 shows the distribution of $r_{\max}$ for RC5 and RLE with $n = 150$ nodes and $w = 300$ m. In case of combination RC5 and RLE algorithms $r_{\max}$ is so small or there is no effect. Table(4.6) has nearly values in our experiment, it means by choose this situation we can not get high security with free energy.
Table 4.12 shows the result of the first decile for $r_{\max}$ for RC5 and RLE.

In last experiment, all values of $r_{\max}$ with different distances and number of rounds are approximately zero. This case is worst of our proposal, we can say nothing will get from this combination because algorithm of cryptography is not high of security, similarly compression algorithm is not best one in save energy.

n \ w	50m	100M	200m	300m	400m
50	0	0	0	1	2
100	0	0	1	1	2
150	1	1	1	2	2
200	1	1	2	2	2

Table 4.12: First decile of $r_{\max}$ for RC5/RLE

4.4/ SUMMARY

We have three different scenarios that combine two algorithms between encryption algorithm with compression algorithm. In each scenario give us values of energy consumed. We combine two algorithms of encryption and compression for data transmission without extra energy. Second point calculate maximum number of rounds in Linear networks, At last we also compute maximum number of rounds in Random network with different scenarios of combine two encryption algorithm and three compression algorithm.

5

System Modelling with Time and Memory

Contents

In this chapter, we try to examine the conditions for time and memory so that we can have energy-free security. The idea is to check if the time taken for the whole process of compression, encryption and transmission is fair considering the application requirements, and if the memory is large enough for the different intermediate buffers. In section 5.1, we describe carefully the whole process with the different possible strategies. In section 5.2, we give a more precise evaluation of the energy for all the strategies. In section 5.3, we analyse time and memory, first with a theoretical analysis and then with a simulation.

5.1/ Description of the process

In this section, we give a description of the whole process of compression, encryption and transmission, that we call the *CET process*, with all the parameters. Figure 5.1 shows the whole process graphically, with the different buffers that are used.

5.1.1/ Hypothesis

First, the sensor generates λ bits of data every δ seconds. This first stage is continuous and regular. It can be achieved easily on the described nodes thanks to their clocks. This sensor data is put in a first buffer that is the input buffer for compression.

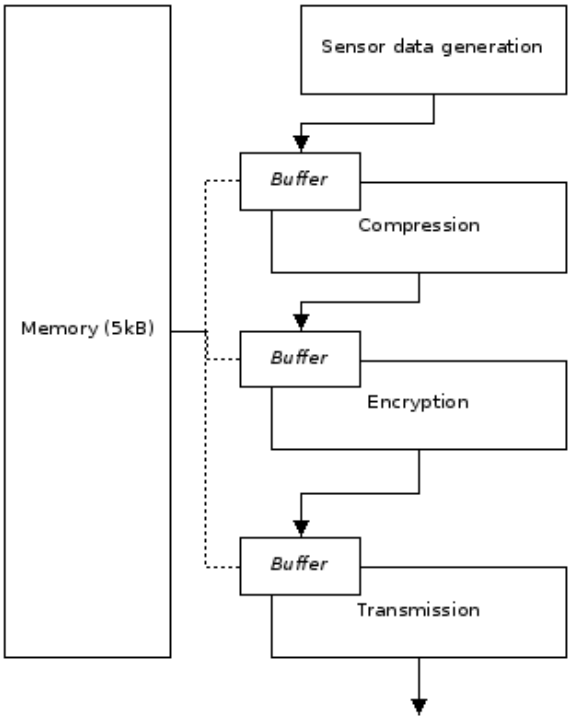

Figure 5.1: The CET process

Second, the compression process waits for some data to be put in its input buffer. As the compression algorithm that we examined can work in a streaming mode, the compression process handles the incoming data as soon as it can. It put its result in a second buffer that is the input buffer for encryption. The compression process is defined by its compression ratio α and by its time model $T^{\text{comp}}(\lambda)$. We note $\lambda' = (1-\alpha)\lambda$ the size of data at the output.

Third, the encryption process waits for a block of data. Then it handles the block and put the result in memory. The encryption algorithms that we examined are all block algorithm. We note B the size of the block of the algorithm: 64 bits for DES, 128 bits for AES. The encryption process is defined by its time model $T^{\text{enc}}(\lambda, r)$.

Fourth, the transmission process takes data from the memory and send them. The memory has a size of M bits. From now on, we fix $M = 5$ KiB, which represents half of the memory of a TelosB or a WSN430. The transmission can follow different strategies that we describe in the next section. The transmission sent data with a data rate of ρ (in bits/s). From now on, we fix $\rho = 250$ kbps, which is the transmission data rate of the CC2420 transceiver. This gives us a time model for transmission:

$$T^{\text{trans}}(\lambda) = \frac{\lambda}{\rho} \tag{5.1}$$

The transmission is done with packets that have a header of size H and a payload of size D. This means that the maximum amount of data that can be sent in a single packet is D. We fix $H = 11$ bytes and $D = 28$ bytes, like in the TinyOS documentation.

We also consider the *T process* where there is no compression and no encryption. It will be a reference when computing the delay that is the consequence of compression and

encryption. Like the CET process, there are several possible strategies, the same as the stragegies for the CET process.

Finally, we fix the amount of generated data to $10M$, which means that for a compression ratio α that is less than 90%, the amount of compressed data can fill the memory of size M at least once. For our simulations, the amount of data is 50 KiB.

5.1.2/ STRATEGIES

Transmission can be done with different strategies. The chosen strategy depends on the application. Every strategy has some advantages and some drawbacks that need to be examined.

Real-Time Strategy In this strategy, data is sent as soon as possible. When data arrives in memory, i.e. when it has just been encrypted, then it is sent immediately, whatever the size of ready data. In the case of the CET process, the size of ready data is a block size B which is strictly less than D. In the case of the T process, the size of the ready block is the size of the generated data λ which is also strictly less than D.

The main advantage of this strategy is that the data does not suffer any more delay than the delay of the process itself. The main drawback is that the packet that is sent is not filled and the number of packets to send everything is not optimal.

Energy-Aware Strategy In this strategy, data is sent in filled packets. Data is put in the packet and as soon as the packet is filled, then it is sent. Blocks can be divided into two consecutive packets in the CET process, as well as the generated data in the T process.

The main advantage of this strategy is that packet are filled so that the number of packet is optimal. The main drawback is that some data is delayed.

Maximum-Data Strategy In this strategy, data is sent when the memory is full. Data is kept in memory after encryption and when the memory is nearly full, i.e. there is no more space for another block of encrypted data, then transmission begins until the memory is empty again.

The main advantage of this strategy is that no data is emitted for quite a long time, which can be an application requirement. And like the previous strategy, the packets are filled so the transmission is optimal. The main drawback is that the delay is the worst possible delay.

5.1.3/ ALGORITHM

In this section, we present the different algorithm for the strategies. Even if the whole process could be done with parallel tasks, generally node processors can do only a single task or cooperative task. So we provide sequential algorithms for the whole process.

The only task that is special is the reading of the sensor value. Algorithm 5 is a function that must be called regularly by the system. Generally, this is done with a hardware timer

that can be triggered regularly and that can call a specified function. So this function should be called every δ seconds. This function is quite simple, it reads the data of the sensor and put it in the input buffer of the compression task.

Algorithm 5 Function that is executed every δ seconds

```
function SENSOR
    PUT(sensor_value, compress_input)
end function
```

Algorithm 6 presents the main algorithm for the real-time strategy. After the initialization, the algorithm enters in the main infinite loop. It first waits for some input from the sensors and sleeps while it is waiting. Then, it starts compression immediately as explained before. Then, if the size of compressed data, is large enough, then encryption begins. And finally, encrypted data is transmitted as soon as possible.

Algorithm 6 Full process for the real-time strategy

```
function MAIN
    INIT
    while true do
        while compress_input.size == 0 do
            SLEEP
        end while
        COMPRESS(compress_input, encrypt_input)
        while encrypt_input.size ≥ B do
            ENCRYPT(encrypt_input, transmit_input)
        end while
        if transmit_input.size > 0 then
            SEND(transmit_input, min(transmit_input.size, D))
        end if
    end while
end function
```

Algorithm 7 presents the main algorithm for the energy-aware strategy. The difference with the previous algorithm is the condition for transmitting the encrypted data. Here, the algorithm checks if it has enough data to fill a packet. Then it sends the filled packets, leaving the rest of the data in memory until the next loops.

Algorithm 8 presents the main algorithm for the maximum-data strategy. The difference with the previous algorithms is that the transmission of the data in memory only starts when the memory is nearly full. Then, data is transmitted with filled packets continuously. At the end, memory is empty and the process can start again.

5.1.4/ CONDITION

In this section, we try to give a condition for this process to happen. The condition looks like the one in section 3.3.2, i.e. the time for compressing, encrypting and transmitting λ bits of generated data must not be greater than δ the period of generation. The condition is given by the following equation:

Algorithm 7 Full process for the energy-aware strategy

```
function MAIN
    INIT
    while true do
        while compress_input.size == 0 do
            SLEEP
        end while
        COMPRESS(compress_input, encrypt_input)
        while encrypt_input.size ≥ B do
            ENCRYPT(encrypt_input, transmit_input)
        end while
        while transmit_input ≥ D do
            SEND(transmit_input, D)
        end while
    end while
end function
```

Algorithm 8 Full process for the maximum-data strategy

```
function MAIN
    INIT
    while true do
        while compress_input.size == 0 do
            SLEEP
        end while
        COMPRESS(compress_input, encrypt_input)
        while encrypt_input.size ≥ B do
            ENCRYPT(encrypt_input, transmit_input)
        end while
        if transmit_input.size ≥ (M − B) then
            while transmit_input ≥ D do
                SEND(transmit_input, D)
            end while
        end if
    end while
end function
```

Data rate (bit/s)	r_{lim} for DES	r_{lim} for AES
0.5	1723	45
1	853	22
2	418	11
4	200	5
8	92	2
16	37	1
32	10	–
64	–	–

Table 5.1: Limit for the number of rounds with the whole process

$$T^{\text{comp}}(\lambda) + T^{\text{enc}}(\lambda', r) + T^{\text{trans}}(\lambda') \leq \delta \tag{5.2}$$

This condition implies a limit on the maximum number of rounds that is given by the following equation:

$$r \leq \frac{\frac{\delta}{\lambda} - T_0^{\text{comp}} - (1-\alpha)(T_0^{\text{enc}} + \frac{1}{\rho})}{(1-\alpha)T_r^{\text{enc}}} = r_{\text{lim}} \tag{5.3}$$

From now on, to decrease the number of cases, we use an imaginary compression algorithm C with the following properties: it has a compression ratio α of 40%, it has a speed T_0^{comp} of 400ns/bit. This is a median value considering our experiments on compression in table 3.10.

Table 5.1 give some values of r_{lim} for DES and AES combined with the imaginary compression algorithm C. It shows that the maximum data rate for a DES with 16 rounds is between 16 and 32 bits/s. Further computations show that it is 26 bits/s. We use this limit later in our computations. As for AES, the maximum data rate for 10 rounds is 2 bits/s. For a data of 64 bits/sec, it is impossible to apply the CET process. This means that compression, encryption and transmission are not fast enough compared to data generation. The consequence is a memory overflow as we will see later.

5.2/ ENERGY FOR TRANSMISSION

The energy of the different strategies in the CET process are the same for compression and the same for encryption. Indeed, the same amount of data is compressed and encrypted in both case and the energy is linear with the size of data. So, the energy only differs when transmitting, and depends directly on the strategy that is used.

More precisely, the energy-aware and maximum-data strategy consume the same energy as the process always send filled packets with these strategies. In the real-time strategy, the process send non-filled packets. There is a constant cost when sending a packet that is due to the header of the packet. So the real-time strategy consumes more energy than the two others.

Energy for optimal transmission In the case of the T process, the generated data of size $10M$ is split in packets of size D, so the number N_1^{OPT} of packets is:

$$N_1^{\mathsf{OPT}} = \left\lceil \frac{10M}{D} \right\rceil$$

And the energy E_1^{OPT} to send these packets is:

$$E_1^{\mathsf{OPT}} = E^{\mathtt{trans}}(N_1^{\mathsf{OPT}}(H + D))$$

In the case of the CET process, cnce compressed, the original generated data of size $10M$ has a size of $10M(1-\alpha)$. The number N_2^{OPT} of packets that need to be sent is:

$$N_2^{\mathsf{OPT}} = \left\lceil \frac{10M(1-\alpha)}{D} \right\rceil$$

And the energy E_2^{OPT} to send these packets is:

$$E_2^{\mathsf{OPT}} = E^{\mathtt{trans}}(N_2^{\mathsf{OPT}}(H + D))$$

Energy for real-time transmission In the case of the T process, with the real-time strategy, data is sent as soon as it is available, i.e. as soon as it has been generated. Generation produces blocks of size λ so data is sent by group of λ bits. So the number N_1^{RT} of packets to be sent is:

$$N_1^{\mathsf{RT}} = \left\lceil \frac{10M}{\lambda} \right\rceil$$

And the energy E_2^{RT} to send these packets is:

$$E_1^{\mathsf{RT}} = E^{\mathtt{trans}}(N_1^{\mathsf{RT}}(H + B))$$

In the case of the CET process, as soon as possible means as soon as encryption is done. As encryption produces blocks of data of size B, data is sent by groups of B bits. So the number N_2^{RT} of packets that need to be sent is:

$$N_2^{\mathsf{RT}} = \left\lceil \frac{10M(1-\alpha)}{B} \right\rceil$$

And the energy E_2^{RT} to send these packets is:

$$E_2^{\mathsf{RT}} = E^{\mathtt{trans}}(N_2^{\mathsf{RT}}(H + B))$$

Data rate (bits/s)	Δ_1^{RT} (s)	Δ_2^{RT} for DES (s)	Δ_2^{RT} for AES (s)
26	0	8.01	–
2	0	56.01	197.69
1	0	108.01	301.69
0.25	0	420.01	925.69

Table 5.2: Time-delay for Real-Time Strategy

5.3/ Analysis of Time and Memory

5.3.1/ Theoretical Analysis

We analyse the delay, and more specifically the delay of the first packet, i.e. the difference between the time of the end of the generation of the first bit of data and the time of the beginning of transmission of this first bit of data. In the real-time strategy of the T process, this delay is $\Delta_1^{\mathsf{RT}} = 0$ by definition. In every other cases, the delay is strictly positive.

Real-Time Strategy With the CET process, the delay consists in waiting that a block of size B is ready to be encrypted. For this to happen, the process has to wait for $\frac{B}{(1-\alpha)}$ bits of data to be generated, at least. So, the delay Δ_2^{RT} is:

$$\Delta_2^{\mathsf{RT}} = \left\lfloor \frac{B}{\lambda'} \right\rfloor \delta + T^{\mathsf{comp}}(\lambda) + T^{\mathsf{enc}}(B)$$

The first term represents the time of generation. As the condition holds, during this time, the first chunks of generated data are compressed. Then, the second term represents the time for compressing the last chunk of generated data. Finally the last term represents the time for encrypting the first block of compressed data. The result of encryption is sent immediately, which indicates the end of the delay.

Table 5.2 shows the time delay for the real-time strategy with the T process and the CET process when using DES and AES.

Energy-Aware Strategy With the T process, we need to wait for at least D bits to be generated. This happens after a delay Δ_1^{EA} of:

$$\Delta_1^{\mathsf{EA}} = \left\lfloor \frac{D}{\lambda} \right\rfloor \delta$$

With the CET process, we need at least D bits of encrypted data, which represents $\left\lceil \frac{D}{B} \right\rceil$ blocks of encrypted data. So, we need $x \left\lceil \frac{D}{B} \right\rceil B$ bits of compressed data. So, the delay Δ_2^{EA} is:

$$\Delta_2^{\mathsf{EA}} = \left\lfloor \frac{x}{\lambda'} \right\rfloor \delta + T^{\mathsf{comp}}(\lambda) + T^{\mathsf{enc}}(B) = \left\lfloor \frac{\left\lceil \frac{D}{B} \right\rceil B}{\lambda'} \right\rfloor \delta + T^{\mathsf{comp}}(\lambda) + T^{\mathsf{enc}}(B)$$

This equation looks like the one for real-time strategy except that we need more compressed data: only B bits for the real-time strategy and x for the energy-aware strategy.

Data rate (bits/s)	Δ_1^{EA} (s)	Δ_2^{EA} for DES (s)	Δ_2^{EA} for AES (s)
26	2.46	20.32	–
2	32	216.01	305.69
1	64	428.01	517.69
0.25	256	1700.01	1789.69

Table 5.3: Time-delay for Energy-Aware Strategy

Data rate (bits/s)	Δ_1^{MD} (s)	Δ_2^{MD} for DES (s)	Δ_2^{MD} for AES (s)
26	1575,38	2630	–
2	20480	34136	34226
1	40960	68268	68358
0.25	163840	273060	273150

Table 5.4: Time-delay for Maximum-Data Strategy

Table 5.3 shows the time delay for the real-time strategy with the T process and the CET process when using DES and AES.

Maximum-Data Strategy With the T process, we need to wait for the memory of size M to be filled. This happens after a delay Δ_1^{MD} of:

$$\Delta_1^{MD} = \left\lfloor \frac{M}{\lambda} \right\rfloor \delta$$

With the CET process, the memory is filled with compressed data of size M and then transmission begins. So the delay Δ_2^{MD} is:

$$\Delta_2^{MD} = \left\lfloor \frac{M}{\lambda'} \right\rfloor \delta + T^{comp}(\lambda) + T^{enc}(B)$$

Again, this equation looks like the one for real-time strategy except that we need the maximum compressed data, which is the total size M of the memory.

Table 5.4 shows the time delay for the real-time strategy with the T process and the CET process when using DES and AES.

5.3.2/ EXPERIMENTAL ANALYSIS

5.3.2.1/ SIMULATOR

In order to view the evolution of memory during the CET process, we made a simulator that implements the CET process with the various strategies. It simulates every tasks (compression, encryption and transmission) and monitors the size of the buffers: `compress_input`, `encrypt_input` and `transmit_input` from the previous algorithms.

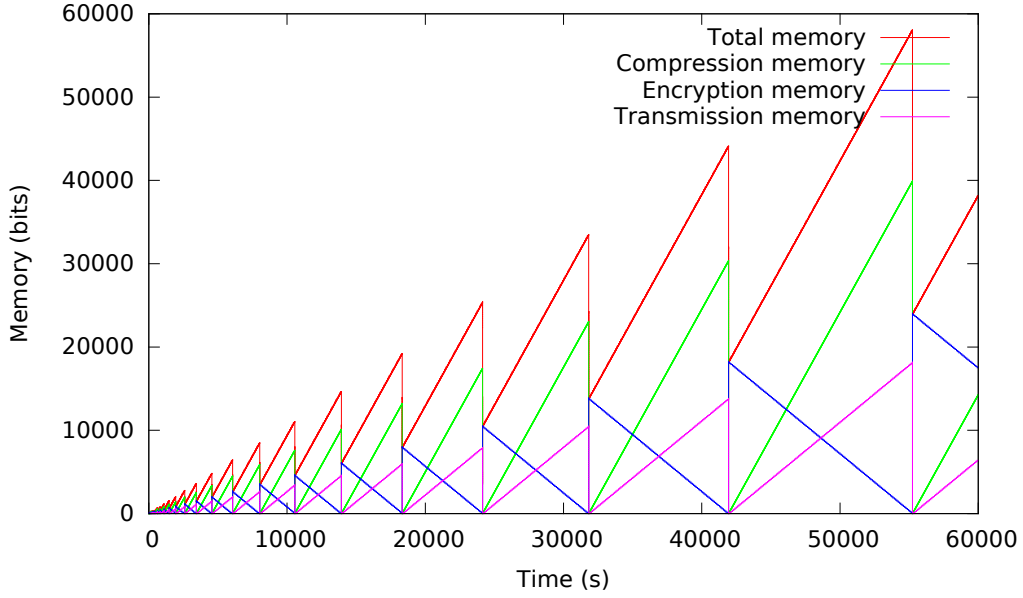

Figure 5.2: Evolution of memory with AES and a data rate of 3 bits/s

5.3.2.2/ RESULTS

Incompatible Data Rate First, we choose a data rate that is not compatible with the condition of equation 5.2. Figure 5.2 shows the CET process with 10 round AES and a data rate of 3 bits/s. We can clearly observe the memory overflow. Once some data is transmitted (when the pink curve falls), compression memory is full and compression starts and is completed quite quickly. It fills the encryption memory and encryption starts, block by block, which fills the transmission memory. And once finished, data is sent. But the time to do all this, the compression memory is bigger than it was at the start of the previous loop. So the total memory increases globally until an overflow.

Real-Time Strategy Figure 5.3 shows the evolution of memory with DES and a data rate of 26 bits/s in the real-time strategy. This data rate is the maximum data rate for DES. Contrary to the previous experiment, we see a nearly-cyclic process that computes data until transmission without memory overflow. There is a startup phase where data is compressed immediately after generation. Once the encryption buffer has enough data (a block), encryption occurs during roughly 4 seconds. And then, data is sent immediately. This happens after about 8 seconds which corresponds to what has been computed in table 5.2. Then, compression occurs again and it produces enough data for encryption so that encryption starts immediately until transmission.

We see that there is not much sleep time. The simulators tells us that on a total time of 15756 s (juste 3.2 seconds after the end of generation), sleep time was 396.984 s, which represents 2.5% of sleep time. This can be explained by the choice of data rate which is quite close to the limit for which there is an overflow.

The simulator also tells us that the maximum total memory is less than 120 bits which is very reasonable. The compression buffer is the one that is the most filled with a maximum of 104 bits.

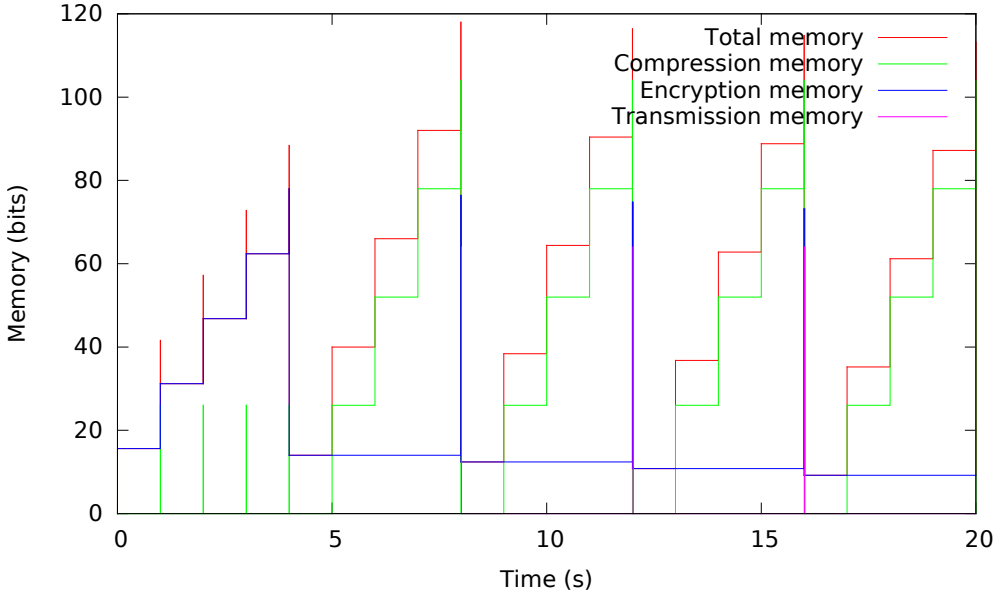

Figure 5.3: Evolution of memory with DES and a data rate of 26 bits/s in the real-time strategy

Figure 5.4 shows the evolution of memory with AES and a data rate of 2 bits/s in the real-time strategy. This data rate is the maximum for AES. We can see a similar startup but in this case, when encryption finishes, there is not enough data for another encryption so there is four loops of compression until encryption can start again. The first transmission is just before 200 s, which again corresponds to what has been computed in table 5.2.

Here, because we see four loops of compression, it means that there is more sleep time than in the previous case: the compression task is waiting for newly generated data. The simulators tells us that the sleep time in this experiment is 28252.8 s for a total time of 204988 s, which represents 13.8% of the total time. Again, this is explained by the choice of the data rate which is close to the limit for which there is an overflow but relatively further than the previous case.

As for the total memory, the simulators has computed a maximum of 256 bits. The maximum for the compression buffer is 184 bits and the maximum for the encryption buffer is 132 bits.

Energy-Aware Strategy Figure 5.5 shows the evolution of memory with DES and a data rate of 26 bits/s in the energy-aware strategy. Compared to the previous experiment with DES, we observe that the transmission memory increases until it reaches the size of a packet, 28 bytes. This happens with several rounds of compression and five rounds of encryption. Again, we see a nearly-cyclic process that alterns between every task. The first transmission occurs near 20 s, which corresponds to what has been computed in table 5.3.

The sleep time and the total time is exactly the same. This is due to the condition in equation 5.2. Indeed, the only difference between the strategies is in the scheduling of the tasks, not on their length as there is exactly the same amount of data generated then compressed then encrypted then transmitted. So the difference between both sides of

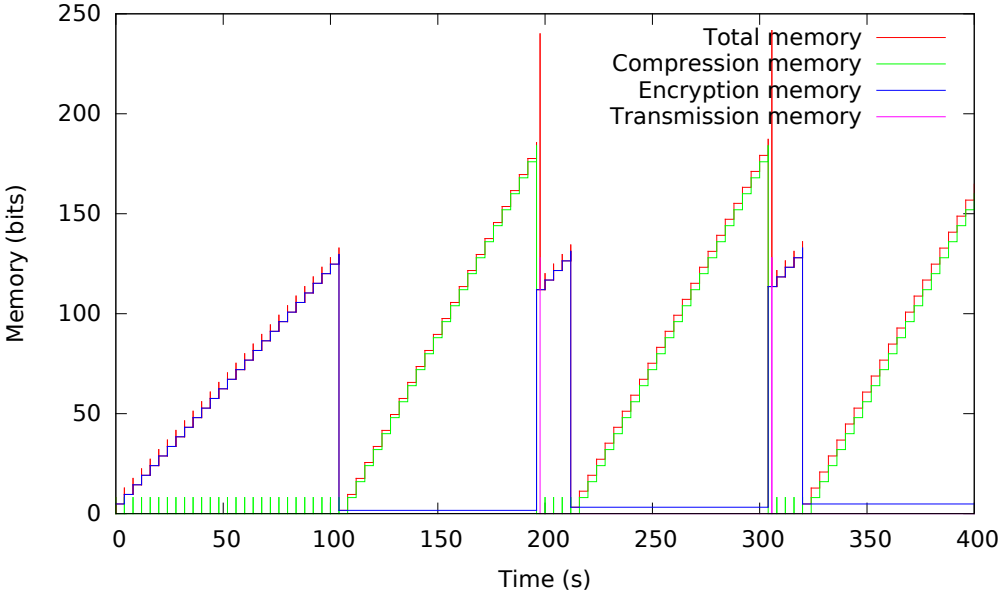

Figure 5.4: Evolution of memory with AES and a data rate of 2 bits/s in the real-time strategy

the equation is constant, and does not depend on the strategy.

The maximum memory that is used is 311 bits, which is not much compared to the previous case. This validates this strategy because it improves the energy consumption while not using so much memory. This strategy appears to be a good compromise between the two other extreme strategies.

Figure 5.6 shows the evolution of memory with AES and a data rate of 2 bits/s in the energy-aware strategy. Again, the figure looks like the previous one for AES except that the transmission memory increases until it reaches the size of a packet. The first transmission occurs after a little more than 300 s, which corresponds again to what has been computed in table 5.3.

The maximum memory for this case is 378 bits. It's just 47% more than the real-time strategy, which is very reasonable. Again, the conclusion is that the energy-aware strategy is a good compromise.

Maximum-Data Strategy Figure 5.7 shows the evolution of memory with DES and a data rate of 26 bits/s in the maximum-data strategy. In this case, the main memory impact is on the transmission memory. The other variations of memory are negligible. Once the buffer of 5 kB is full, then it is transmitted, which happens after 2100 s, which corresponds again to what has been computed in table 5.4.

The maximum total memory is of course the memory size.

Figure 5.8 shows the evolution of memory with AES and a data rate of 2 bits/s in the maximum-data strategy. Like DES, the main memory impact is one the transmission memory that is used until it's full. The first transmission occurs after more than 34000 s, which corresponds again to what has been computed in table 5.4. And the maximum total memory is again the memory size.

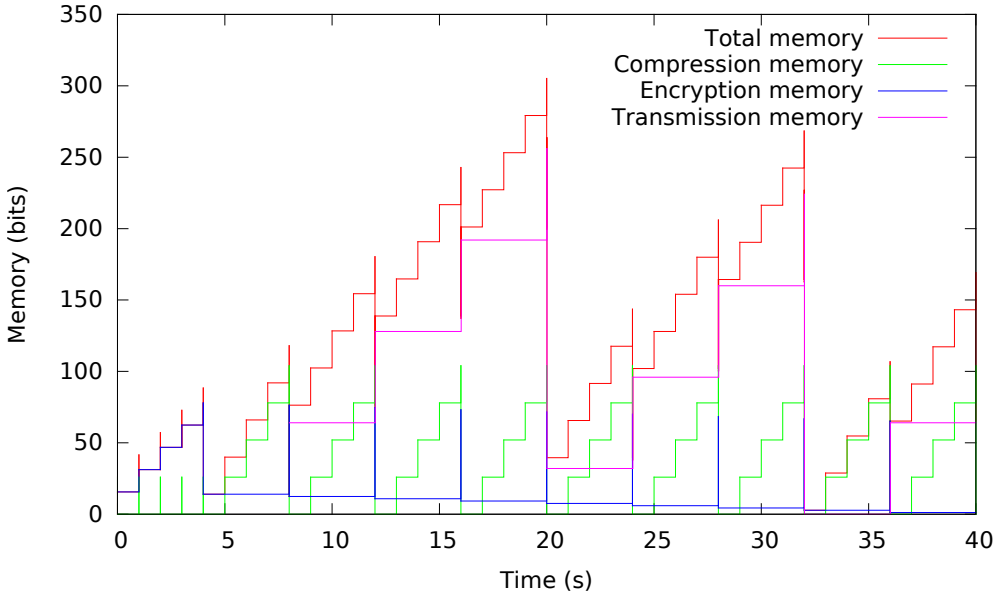

Figure 5.5: Evolution of memory with DES and a data rate of 26 bits/s in the energy-aware strategy

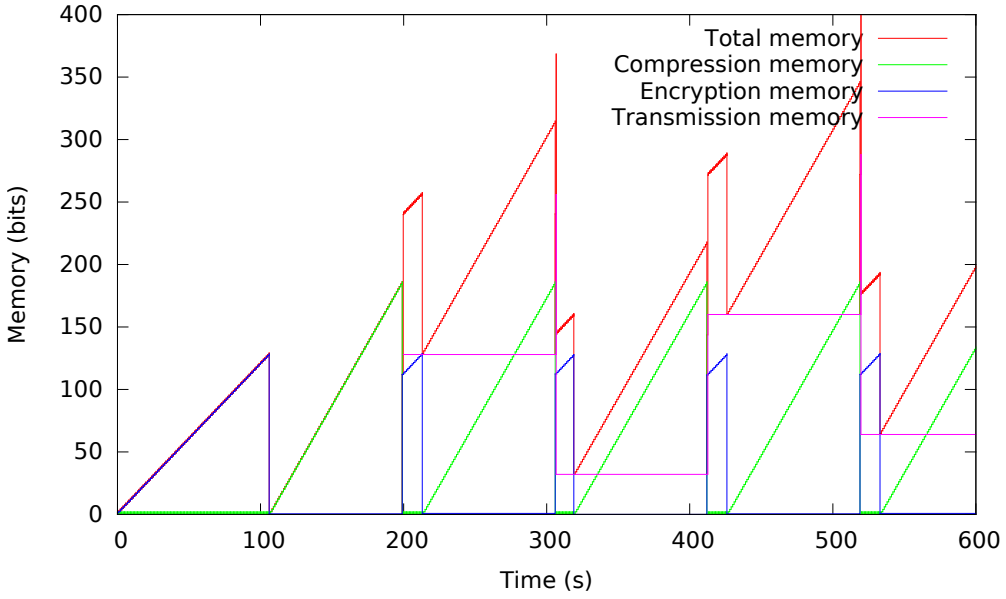

Figure 5.6: Evolution of memory with AES and a data rate of 2 bits/s in the energy-aware strategy

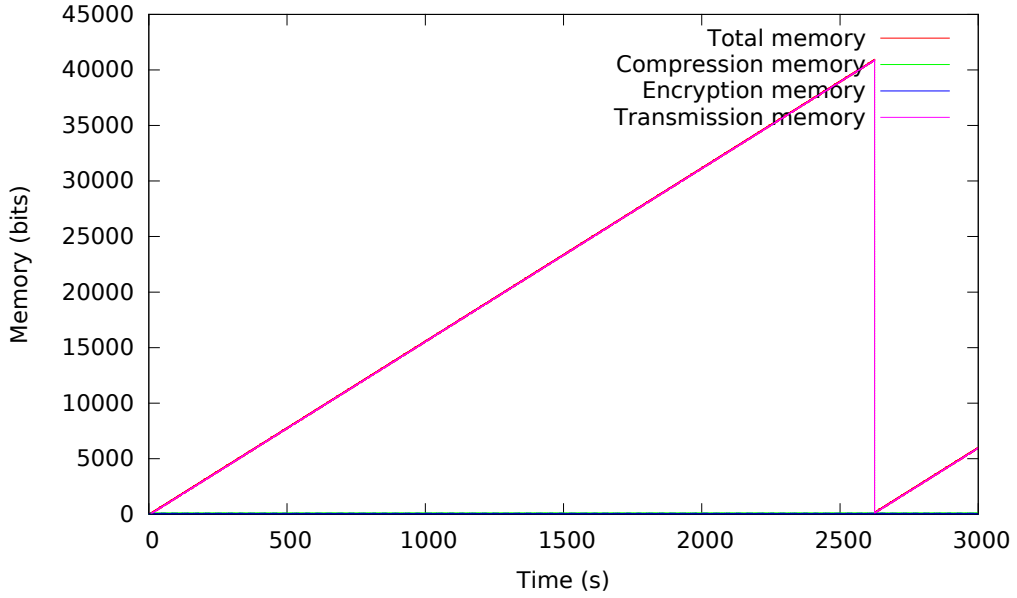

Figure 5.7: Evolution of memory with DES and a data rate of 26 bits/s in the maximum-data strategy

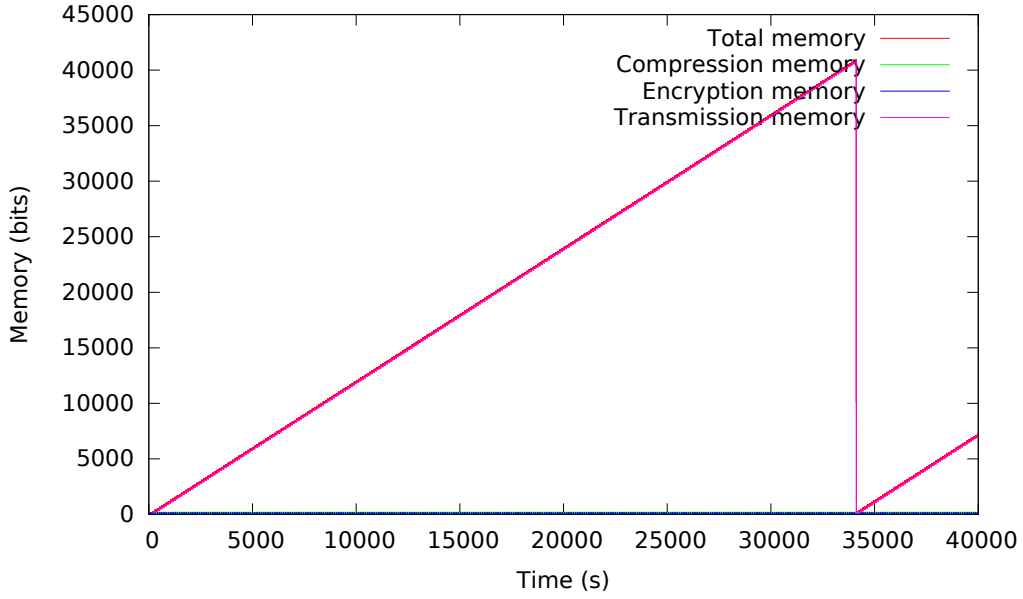

Figure 5.8: Evolution of memory with AES and a data rate of 2 bits/s in the maximum-data strategy

5.4/ SUMMARY

Memory-size of sensor node and time-delay of transmission is done effected depending on type of strategy that we will use for our application. We suppose a system to do compression, encryption and transmission as one process, then we assume fixed values for all algorithms that we are using. For making control in sensor node to do function of selected application under fixed condition, we suppose three different strategies, each one will give us some advantage relate to our selected application. These strategies will calculate the energy transmission to give us which strategy will be proper for some applications that worry about energy consumption in case of security. They give us also hand to avoid loss of data in case of high reading data. For guaranteeing arriving data transmission to its target in short time as some applications needed, the strategy will give us some solutions for it.

6

CONCLUSION AND PERSPECTIVES

Contents

6.1/ CONCLUSION

In this thesis we are working on wireless sensor network to improve our monitoring system for some natural phenomena. We have addressed problem related to consumptions of energy with increasing the level of security. We can find the relationship between our conditions for getting some information about some natural phenomenon with proper level of security.

We have presented principles of our work that we building on them, they divided into some way, first How we can send secure data through wireless sensor network?, which best encryption algorithm we can select, second How we can implement our model without consume extra energy more than unsecured situation, third Which technologies we can choose to avoid extra energy consumed when we apply cryptography techniques.

We have used Senslab platform to implemented our code on real sensor WSN430 to get precise results that can give us good decision for building the network and lifetime of its sensor.

We have chosen some encryption algorithms for wireless sensor network to improve and protect our data when we send it trough the wireless networks. Each algorithm has level of security and level of energy consumption, we find positive relationship between selected level of security and amount of energy consumed, these algorithm are used in some applications of sensor node depending on its conditions of security and its lifetime.

We combine algorithms of encryption and algorithms of compression to improve of performance of sensor node to send secure data without extra energy, this improvement is done by choosing special two algorithms one encryption with other compression. By obtained models of memory size of sensor node We calculate the proper number of bytes for applying our models without loss sensed data.

We have chosen two types of network topologies to test our models of energy in different scenarios to know more about energy consumption of sensor. we have tested three encryption algorithms to decide which one will be best for selected application that we

will choose depend on level of security or/and lifetime of sensor.

In random network, we combine two encryption algorithms and three compression algorithms in separate scenarios to test level of security, we found best case that mix DES encryption algorithm with K-RLE compression algorithm.

For obtain optimal system of sending secure data without extra energy of sensor node, we suppose a system has three sequential operations with fixed conditions that control flow of data from sensing until sending. This system will give us precise view on How we can choose all requirements of selected application of sensor node?, by different strategies we can take some decisions for improving sensor node to be work properly.

6.2/ PERSPECTIVES

After doing this work, we can extended it to cover some other problems of sensor node that can worry about energy consumption and how we can get long life for the sensor with working in properly level of security.

For getting more precise results in real time of calculate memory-size and time-delay of each process like compression and encryption we can using Senslab as a platform that has real sensor.

We can improve the security side by apply some modified cryptography algorithms for wireless sensor networks that can give us low level of energy consumed, similarly we can applying some modified compression algorithms that can give us high compression ratio and low level of energy consumed.

Topologies of wireless sensor networks has different types, each one has advantages and disadvantages we can improve some problems of each one by make one network has two types of sensor node (fixed, mobile) to avoid energy consumption of find its neighbours.

Data transmission consumes maximum value of energy, we can improve the size of packet for some modified routing algorithms to be proper for save energy and avoiding data losing.

Using different simulator to prove our improvement on all algorithms of encryption, compression and transmission to get proper model of each process and for mixed all operations to calculate energy consumptions, memory-size and time-delay.

BIBLIOGRAPHY

[1] Introduction to tinyOS and nesC. In *wireless sensor networks , Getting Started Guide*, volume 9, pages 53–59, 2004.

[2] Kemal Akkaya and Mohamed Younis. A survey on routing protocols for wireless sensor networks. *Ad hoc networks*, 3(3):325–349, 2005.

[3] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A survey on sensor networks. *Communications magazine, IEEE*, 40(8):102–114, 2002.

[4] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless Sensor Networks: A Survey. *Computer networks*, 38(4):393–422, 2002.

[5] Wiam Al Hayek . An effective method for data compression based on adaptive character wordlength. *INTERNATIONAL Arab Journal of e-Technology*, 2(4):197–202, June 2012.

[6] Mohammed Al-Laham and Ibrahiem MM El Emary. Comparative study between various algorithms of data compression techniques. *IJCSNS*, 7(4):281, 2007.

[7] Dr. Yossra H. Ali. Proposed 256 bits RC5 encryption algorithm using type-3 feistel network. *Eng.and Tech. Journal*, 28(12):2337–2352, 2010.

[8] F. Amin, A. H. Jahangir, and H. Rasifard. Analysis of public-key cryptography for wireless sensor networks security. *Proceedings of World Academy of Science: Engineering & Technology*, 2(5):403–408, 2008.

[9] Giuseppe Anastasi, Marco Conti, Mario Di Francesco, and Andrea Passarella. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Netw.*, 7(3):537–568, May 2009.

[10] Ayushi. Article: A symmetric key cryptographic algorithm. *International Journal of Computer Applications*, 1(14):1–4, February 2010. Published By Foundation of Computer Science.

[11] Lu Bai, Lian Zhao, and Zaiyi Liao. Energy balance in cooperative wireless sensor network. In *Wireless Conference, 2008. EW 2008. 14th European*, pages 1–5, June 2008.

[12] Ms. Neha A. Bhatia, Prof. Himanshu Arora, and Ms. Anuradha Konidena. An efficient RLE algorithm for compressing image based upon tolerance value. In *International Journal of Engineering and Management Research*, volume 3, pages 30–32, 2013.

[13] David Boyle and Thomas Newe. Securing wireless sensor networks: Security architectures. *Journal of Networks*, 3(1):65–77, 2008.

[14] E. Callaway, P. Gorday, L. Hester, J. A. Gutierrez, M. Naeve, B. Heile, and V. Bahl. Home networking with IEEE 802.15.4: a developing standard for low-rate wireless personal area networks. *Communications Magazine, IEEE*, 40(8):70–77, August 2002.

[15] Ed Callaway, Paul Gorday, Lance Hester, Jose A Gutierrez, Marco Naeve, Bob Heile, and Venkat Bahl. Home networking with IEEE 802.15. 4: a developing standard for low-rate wireless personal area networks. *IEEE Communications Magazine*, 40(8):70–77, 2002.

[16] H. Cam, S. Ozdemir, D. Muthuavinashiappan, and P. Nair. Energy efficient security protocol for wireless sensor networks. In *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, volume 5, pages 2981–2984, October 2003.

[17] E. P. Capo-Chichi, H. Guyennet, and J-M Friedt. K-RLE: A new data compression algorithm for wireless sensor network. In *Sensor Technologies and Applications, 2009. SENSORCOMM '09. Third International Conference on*, pages 502–507, June 2009.

[18] Jae-Hwan Chang and Leandros Tassiulas. Maximum lifetime routing in wireless sensor networks. *IEEE/ACM Trans. Netw.*, 12(4):609–619, August 2004.

[19] Culler David, Estrin Deborab, and Srivastava Mani. Guest editors' introduction: Overview of sensor networks. *IEEE Computer*, 37(8):41–50, August 2004.

[20] G. de Meulenaer, F. Gosset, O.-X. Standaert, and O. Pereira. On the energy cost of communication and cryptography in wireless sensor networks. In *Networking and Communications, 2008. WIMOB '08. IEEE International Conference on Wireless and Mobile Computing,*, pages 580–585, October 2008.

[21] Clément Burin des Roziers, Guillaume Chelius, Tony Ducrocq, Eric Fleury, Antoine Fraboulet, Antoine Gallais, Nathalie Mitton, Thomas Noël, and Julien Vandaele. Using sensLAB as a first class scientific tool for large scale wireless sensor network experiments. In *Proceedings of the 10th International IFIP TC 6 Conference on Networking - Volume Part I*, NETWORKING'11, pages 147–159. Springer-Verlag, 2011.

[22] Tony Ducrocq, Julien Vandaele, Nathalie Mitton, and David Simplot Ryl. Large scale geolocalization and routing experimentation with the sensLAB testbed. pages 751–753, 2010.

[23] Omer Elkeelany . Design and analysis of various models of RC5-192 embedded information security algorithm. *INTERNATIONAL JOURNAL OF APPLIED MATHEMATICS AND INFORMATICS*, 2(1):18–27, 2008.

[24] Ana-Belen Garcia-Hernando, Jose-Fernan Martinez-Ortega, Juan-Manuel Lopez-Navarro, Aggeliki Prayati, and Luis Redondo-Lopez. *Problem Solving for Wireless Sensor Networks*. Springer-Verlag London, 2008.

[25] Soheil Ghiasi, Ankur Srivastava, Xiaojian Yang, and Majid Sarrafzadeh. Optimal energy aware clustering in sensor networks. *Sensors*, 2(7):258–269, 2002.

[26] Andrea Goldsmith . *Wireless Communications*, volume 43. Cambridge University Press, 2005.

[27] A. J. Goldsmith and S. B. Wicker. Design challenges for energy-constrained ad hoc wireless networks. *Wireless Communications, IEEE*, 9(4):8–27, August 2002.

[28] Shivangi Goyal. A survey on the applications of cryptography. *International Journal of Science and Technology Volume, IJST*, 1(3):137–140, March 2012.

[29] Dae-Man Han and Jae-Hyun Lim. Smart home energy management system using IEEE 802.15.4 and zigbee. *Consumer Electronics, IEEE Transactions on*, 56(3):1403–1410, August 2010.

[30] Wendi R. Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*. IEEE, January 2000.

[31] Mark Hempstead, Michael J. Lyons, David Brooks, and Gu-Yeon Wei. Survey of hardware systems for wireless sensor networks. *Journal of Low Power Electronics*, 4:580–585, October 2008.

[32] Y. T. Hou, Yi Shi, H. D. Sherali, and S. F. Midkiff. On energy provisioning and relay node placement for wireless sensor networks. *Wireless Communications, IEEE Transactions on*, 4(5):2579–2590, September 2005.

[33] En hui Yang and John C. Kieffer. Efficient universal lossless data compression algorithms based on a greedy sequential grammar transform—part one: Without context models. *TRANSACTIONS ON INFORMATION THEORY,, IEEE*, 46(3):755–777, 2000.

[34] Yang Jing, Li Zetao, and Lin Yi. An improved routing algorithm based on LEACH for wireless sensor networks. In *Control and Decision Conference (CCDC), 2013 25th Chinese*, pages 3716–3720, May 2013.

[35] Raja Jurdak, Antonio G. Ruzzelli, and Gregory M. P. OHare. Adaptive radio modes in sensor networks: How deep to sleep? In *Sensor, Mesh and Ad Hoc Communications and Networks, 2008. SECON '08. 5th Annual IEEE Communications Society Conference on*, pages 386–394, June 2008.

[36] Ansgar Kellner, Omar Alfandi, and Dieter Hogrefe. A survey on measures for secure routing in wireless sensor networks. *International Journal of Sensor Networks and Data Communications*, 1:1–17, 2012.

[37] Yee Wei Law, Jeroen Doumen, and Pieter Hartel. Survey and benchmark of block ciphers for wireless sensor networks. *ACM Trans. Sen. Netw.*, 2(1):65–93, February 2006.

[38] Jongdeog Lee, Krasimira Kapitanova, and Sang H. Son. The price of security in wireless sensor networks. *Comput. Netw.*, 54(17):2967–2978, December 2010.

[39] F. L. LEWIS. *Wireless Sensor Networks*, volume 43. John Wiley Sons, 2004.

[40] Hsing-Shao Liu , Chi-Cheng Chuang, Chih-Chung Lin, Ray-I Chang, Chia-Hui Wang, and Ching-Chia Hsieh . Data compression for energy efficient communication on ubiquitous sensor networks. *Tamkang Journal of Science and Engineering*, 14(3):245–254, 2011.

[41] Omar Adil Mahdi, Mazin AbedMohammed, and Ahmed Jasim Mohamed. Implementing a novel approach an convert audio compression to text coding via hybrid technique. volume 9, pages 53–59, 2012.

[42] A. S. Malik, Jingming Kuang, iakang Liu, and Huihui Xiang. Affect of confinement of event field within the sensor field on network lifetime and energy consumption for cluster-based wireless sensor networks. In *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on*, pages 1–6, September 2009.

[43] Shanta Mandal and Rituparna Chaki. A secure encryption logic for communication in wireless sensor networks. *International Journal on Cryptography and Information Security (IJCIS)*, 2(3):107–108, September 2012.

[44] Francesco Marcelloni and Massimo Vecchio. A simple algorithm for data compression in wireless sensor networks. *IEEE Communications Letters*, 12(6):411–413, June 2008.

[45] Francesco Marcelloni and Massimo Vecchio. An efficient lossless compression algorithm for tiny nodes of monitoring wireless sensor networks. *The Computer Journal*, pages 1–19, April 2009.

[46] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[47] Richard A. Mollin. *An INTRODUCTION to CRYPTOGRAPHY*. Taylor and Francis Group, LLC, second edition edition, 2007.

[48] Mark Nelson and Jean loup Gailly. *The Data Compression Book 2nd edition*. League For Programming Freedom, 1995.

[49] Suat Ozdemir and Yang Xiao. Secure data aggregation in wireless sensor networks: A comprehensive overview. *Computer networks*, 12(53):2022–2037, March 2009.

[50] Madhumita Panda. Security in wireless sensor networks using cryptographic techniques. *American Journal of Engineering Research (AJER)*, 3:50–56, 2014.

[51] Abhishek Pandey and R. C. Tripathi . A survey on wireless sensor networks security. *International Journal of Computer Applications*, 3:43–49, June 2010.

[52] Adrian Perrig, Robert Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler. SPINS: Security protocols for sensor networks. *Wirel. Netw.*, 8(5):521–534, September 2002.

[53] Krzysztof Piotrowski, Peter Langendoerfer, and Steffen Peter. How public key cryptography influences wireless sensor node lifetime. In *Proceedings of the Fourth ACM Workshop on Security of Ad Hoc and Sensor Networks*, SASN '06, pages 169–176, New York, NY, USA, 2006. ACM.

[54] Shrusti Porwal, Yashi Chaudhary, Jitendra Joshi, and Manish Jain. Data compression methodologies for lossless data and comparison between algorithms. *International Journal of Engineering Science and Innovative Technology (IJESIT)*, 2(2):142–147, 2013.

[55] Gregory J Pottie and William J Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, 2000.

[56] P. Prasithsangaree and P. Krishnamurthy. Analysis of energy consumption of RC4 and AES algorithms in wireless LANs. In *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, volume 3, pages 1445–1449, December 2003.

[57] Sebastian Puthenpurayil, Ruirui Gu , and Shuvra S Bhattacharyya. Energy-Aware Data Compression for Wireless Sensor Networks. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 2, pages 45–48, April 2007.

[58] Kodituwakku S. R. and Amarasinghe U. S. Comparison of lossless data compression algorithms for text data. *Indian Journal of Computer Science and Engineering*, 1(4):416–425, June 2012.

[59] V. Raghunathan, C. Schurgers, Sung Park, and M. B. Srivastava. Energy-aware wireless microsensor networks. *Signal Processing Magazine, IEEE*, 19(2):40–50, March 2002.

[60] Srivaths Ravi, Anand Raghunathan, and Nachiketh Potlapally. Securing wireless data: System architecture challenges. In *Proceedings of the 15th International Symposium on System Synthesis*, ISSS '02, pages 195–200, New York, NY, USA, 2002. ACM.

[61] Tobias Reusing. Comparison of operating systems tinyOS and contiki. Technical report, Technische University München, 2012.

[62] Ronald L Rivest. The RC5 encryption algorithm. In *Fast Software Encryption*, pages 86–96. Springer, 1995.

[63] V. Rodoplu and T. H. Meng. Minimum energy mobile wireless networks. In *Communications, 1998. ICC 98. Conference Record. 1998 IEEE International Conference on*, volume 3, pages 1633–1639, June 1998.

[64] Clément Burin Des Rosiers, Guillaume Chelius, Eric Fleury, Antoine Fraboulet, Antoine Gallais, Nathalie Mitton, and Thomas Noël. SensLAB Very Large Scale Open Wireless Sensor Network Testbed. In *Proc. 7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCOM)*, Shanghai, Chine, April 2011.

[65] N Ruangchaijatupon and P Krishnamurthy. Encryption and power consumption in wireless LANs. In *The Third IEEE workshop on wireless LANS*, pages 148–152, 2001.

[66] Christopher M. Sadler and Margaret Martonosi. Data compression algorithms for energy-constrained devices in delay tolerant networks. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, SenSys '06, pages 265–278. ACM, 2006.

[67] Mahmud Salauddin. An improved data compression method for general data. *International Journal of Scientific and Engineering Research*, 3:1–4, March 2012.

[68] H. Saxena, Chunyu Ai, M. Valero, Yingshu Li, and R. Beyah. DSF - A distributed security framework for heterogeneous wireless sensor networks. In *MILITARY COMMUNICATIONS CONFERENCE, 2010 - MILCOM 2010*, pages 1836–1843, October 2010.

[69] Stefan Schmid and Roger Wattenhofer. Algorithmic Models for Sensor Networks. In *14th International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS), Island of Rhodes, Greece*, April 2006.

[70] R. C. Shah and J. M. Rabaey. Energy aware routing for low energy ad hoc sensor networks. In *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, volume 1, pages 350–355, March 2002.

[71] Vikas Singla, Rakesh Singla, and Sandeep Gupta. Data compression modelling: Huffman and arithmetic. *International Journal of The Computer, the Internet and Management*, 16(3):64–68, 2008.

[72] S. Slijepcevic, M. Potkonjak, V. Tsiatsis, S. Zimbeck, and M. B. Srivastava. On communication security in wireless ad-hoc sensor networks. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2002. WET ICE 2002. Proceedings. Eleventh IEEE International Workshops on*, pages 139–144, 2002.

[73] A. Somov, I. Minakov, A. Simalatsar, G. Fontana, and R. Passerone. A methodology for power consumption evaluation of wireless sensor networks. In *Emerging Technologies Factory Automation, 2009. ETFA 2009. IEEE Conference on*, pages 1–8, September 2009.

[74] John A. Stankovic. Wireless sensor networks. pages 1–3, 2006.

[75] I Made Agus Dwi Suarjaya. A new algorithm for data compression optimization. *International Journal of Advanced Computer Science and Applications,IJACSA*, 3(8):14–17, 2012.

[76] Paul Tomsy and Kumar G. Santhosh. Safe contiki OS: Type and memory safety for contiki OS. In *Advances in Recent Technologies in Communication and Computing, 2009. ARTCom '09. International Conference on*, DOI 10.1109/ARTCom, pages 169–171. IEEE Computer Society, October 2009.

[77] V Chaitanya Tummalapalli and MD. Khwaja Muinnuddin Chisti. Implementation of low power RC5 algorithm in XILINX FPGA. *International Journal of Engineering Research and Applications (IJERA)*, 2(3):924–928, May 2012.

[78] Chandni Vaghasia and Kirti Bathwar. Public key encryption algorithms for wireless sensor networks in tinyOS. *International Journal of Innovative Technology and Exploring Engineering , IJITEE*, 2:28–34, March 2013.

[79] Ranganathan Vidhyapriya and Ponnusamy Vanathi. Energy efficient data compression in wireless sensor networks. *The International Arab Journal of Information Technology*, 6(3):297–303, July 2009.

[80] Gupta Vishwa, Singh Gajendra, and Gupta Ravindra. Advance cryptography algorithm for improving data security. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2, January 2012.

[81] A. S. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz. Energy analysis of public-key cryptography for wireless sensor networks. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, pages 324–328, March 2005.

[82] Qin Wang and Woodward Yang. Energy consumption model for power management in wireless sensor networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON '07. 4th Annual IEEE Communications Society Conference on*, pages 142–151, June 2007.

[83] You-Chiun Wang. Data compression techniques in wireless sensor networks. *Pervasive Computing, New York: Nova Science Publishers, Inc*, 2012.

[84] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Comput. Netw.*, 52(12):2292–2330, August 2008.

[85] Mohamed Younis, Moustafa Youssef, and Khaled Arisha. Energy-aware management for cluster-based sensor networks. *Computer Networks*, 43(5):649–668, 2003.

[86] Xinmiao Zhang and K. K. Parhi. On the optimum constructions of composite field for the AES algorithm. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 53(10):1153–1157, October 2006.

[87] Jun Zheng and Abbas Jamalipour. *Wireless sensor networks: a networking perspective*. John Wiley Sons, 2009.

List of Figures

List of Tables

Abstract:

Wireless sensor networks give us opportunities to improve many applications in many fields (medicine, military, etc.). The data collected by sensor node flies as plain text on sensor network and can be intercepted by a spy. Depending on the importance of data, the wanted level of security could be high which may impact the energy consumption of sensor nodes. These two constraints, security and energy are difficult to combine. There is a trade-off between energy savings that will determine the lifetime of the network and the level of security desired by the application.
The objective of this thesis is to study the trade-off between these two constraints, both from a theoretical perspective and from a practical point of view (with an implementation of algorithms and real tests on sensor networks). The DES symmetric cryptographic algorithm was chosen as a case study. The results obtained on the academic platform Senslab have shown a clear relationship between energy consumption and the number of rounds of DES and therefore the level of security. These experiments were repeated with the AES algorithm, newer and safer, but also more energy consumer.
Then, from the results, a generic model of consumption for cryptographic algorithms has been built for a complete network. The complementary use of data compression has reduced this impact of energy consumption in an interesting way. Finally, the memory usage and the time of encryption and compression were evaluated in order to stay within realistic ranges of use.

Keywords: Wireless sensor networks, Security, Energy

Résumé :

L'utilisation des réseaux de capteurs offre de nouvelles perspectives dans de nombreux domaines (médecine, militaire, etc). les données récoltées par ces capteurs circulent en clair sur les réseaux de capteurs et peuvent être interceptées par un espion. Selon le domaine d'utilisation, le niveau de sécurité souhaité peut être élevé, ce qui peut provoquer une hausse de la consommation d'énergie sur les nœuds. Ces deux contraintes, sécurité et énergie, sont difficilement conciliables. Il y a donc un compromis à trouver entre l'économie d'énergie qui va conditionner la durée de vie du réseau, et le niveau de sécurité souhaité par l'application.
L'objectif de cette thèse est d'étudier les compromis à trouver entre ces deux contraintes, à la fois d'un point de vue théorique et d'un point de vue pratique (par une implémentation des algorithmes et des tests réels sur des réseaux de capteurs par choisissant les algorithmes DES et AES). L'algorithme de cryptographie symétrique DES a été choisi comme objet d'étude. Les résultats obtenus sur la plateforme académique Senslab ont permis de déterminer une relation précise entre la consommation d'énergie et le nombre de rondes de DES et donc le niveau de sécurité. Ces expériences ont été renouvelées avec l'algorithme AES, plus récent et plus sûr, mais aussi plus consommateur en énergie.
Puis, à partir des résultats obtenus, un modèle générique de consommation pour les algorithmes cryptographiques a été construit pour un réseau complet. L'utilisation complémentaire de la compression des données a permis de réduire cet impact de manière intéressante. Enfin, l'utilisation de la mémoire et l'évaluation du temps de chiffrement et de compression ont été évalué de manière à rester dans des fourchettes réalistes d'utilisation réaliste.

Mots-clés : Réseaux de capteurs sans fil, Sécurité, Énergie

■ École doctorale SPIM 16 route de Gray F - 25030 Besançon cedex
■ tél. +33 (0)3 81 66 66 02 ■ ed-spim@univ-fcomte.fr ■ www.ed-spim.univ-fcomte.fr

UFC
UNIVERSITÉ DE FRANCHE-COMTÉ