



Université Sidi Mohammed Ben Abdellah
Faculté des Sciences Dhar El Mahraz- Fès
Centre d'Etudes Doctorales
"Sciences et Technologies"

Formation Doctorale : STIC

Discipline : Informatique

Spécialité : Informatique

Laboratoire : LIAN

THESE DE DOCTORAT

Présentée par

Christophe ISHIMWE NGABO

**Secure Integration of the Wireless Sensor Networks with the Cloud
Infrastructure.**
Case Study: Smart Cities

Soutenue le 04/05/2019 devant le jury composé de :

Pr. Mostafa BELLAFKIH	Institut National des Postes et Télécommunications Rabat, Maroc	Président
Pr. Mustapha HEDABOU	Université Cadi Ayyad Marrakech, Maroc	Rapporteur
Pr. Aawatif HAYAR	Université Hassan II Casablanca, Maroc	Rapporteur
Pr. Noureddine CHENFOUR	Université Sidi Mohamed Ben Abdellah Fès, Maroc	Rapporteur
Pr. Rachid EL KOUCH	Institut National des Postes et Télécommunications Rabat, Maroc	Examineur
Pr. Ahmed LBATH	Université Joseph Fourier Grenoble, France	Examineur
Pr. Omar EL BEQQALI	Université Sidi Mohamed Ben Abdellah Fès, Maroc	Directeur de thèse

Acknowledgements

I would like to thank everyone who contributed to the success of my thesis and who helped me in writing this report.

First of all, my thanks to my thesis supervisor, **Prof Omar EL BEQQALI**, who helped me a lot in my research and made me feel at home in our LIAN laboratory. His listening and advice allowed me to target the best directions, and find the information that was fully in line with my expectations.

I would like to thank my colleague, **Dr Sanae EL OUKKAL**, former PhD student of our LIAN laboratory, for her welcome, the time spent together and the sharing of her expertise on a daily basis. Thanks also to his confidence I was able to accomplish totally in my research. She was a precious help in the most delicate moments.

I also thank the Rwandan government for its financial contribution to the completion of this thesis. I was very happy and grateful to know that I was granted as a scholarship recipient for 8 years at the university.

I cannot forget the lecturer **Dr Ghazi Aziz** of Computer Science at the Sup' management school in Fez for his technical assistance with wireless sensor platforms. His help was very helpful to me when I needed it so much.

I also thank the entire team of the CED "Science and Technology" for their hospitality, their team spirit and in particular **Prof Mohamed EL HASSOUNI**, who helped me a lot to follow and complete the doctoral training.

Finally, I would like to thank all people who advised me and read again during the writing of this thesis dissertation.

Dedications

To my dear parents, for all their sacrifices, their love, their tenderness, their financial support and their prayers throughout my studies.

To my dear Sisters **Christine** and **Sylvie** for their constant encouragement, and their moral and spiritual support.

To my dear brothers **Claude** and **Olivier** for his financial support and encouragement.

To all my family for their support throughout my university career, especially my aunts **Clémence** and **Marie**.

May this work be the accomplishment of your vows so much alleged, and flees from your infallible support.

Thank you for always being here for me.

Abstract

This thesis considers the integration of wireless sensor networks with cloud computing. The questions asked are: how to store data from the wireless sensor network despite storage gaps in this type of network? How to successfully integrate two heterogeneous networks (the wireless sensor network and the traditional network such as the internet) so that they can communicate with each other? If cloud computing offers the best storage and computing power solution, how can we ensure data privacy from the wireless sensor network to the end user through the cloud? What to do with a multitude of diverse data despite having a computing power thanks to the cloud? In short, this doctoral dissertation tries to deal with those problems.

To answer the first question, the proposed solution in this thesis is to make useful the advantage of the services offered by cloud computing, such as computing and storage capabilities. In a wireless sensor network, there can be thousands of sensor nodes collecting multiple physical measures that are converted into valuable data for various applications. These data are collected regularly in large quantities so that it is almost impossible to store them locally in the network because each sensor node has only a few megabytes of mass storage and less computing power (a sensor node embeds a processor usually from 8 to 16 bits). These data are often used in real time but what to do when they are just used and another fresh data arrives? It is desirable to store or archive them somewhere for future use as in the prediction, statistics etc. In this thesis, we propose the integration of sensor networks with cloud infrastructures to answer the question asked above.

Another problem that this thesis has been able to deal with is the heterogeneity of the networks, on the one side, we have the wireless sensor network and on the other side it is generally the internet or the intranet. In the wireless sensor network, the network protocols needed to send data from one node to another are not the same protocols that are used on the internet. We proposed an architectural solution dividing the whole system into different physical and logical layers. Logically, the wireless sensor network is reached through the gateway (connected directly to the base station of the wireless sensor network) from the external network, so this gateway mediates between the wireless sensor network and the cloud as well as end users. To enable the wireless sensor network, the cloud, and end users to communicate, we have used interoperable technology (middleware), the choice of which has been the subject of a comparative study between most used interoperability technologies. Java RMI has been identified as the best middleware for the proposed architecture against CORBA and web services.

The sensor nodes in the network are often scattered and are close to each other so that they can communicate directly with each other. Sometimes sensors nodes in the same environment may sense the same or almost similar values, which causes the problem of duplication of data. One of the most common techniques used in the wireless sensor network to eliminate data duplication is to aggregate data from a cluster of sensor nodes into a cluster header in order to send to the base station only an aggregated value (such as the average for example). This thesis also focuses on the confidentiality of data that can be aggregated or not throughout the system (from the wireless sensor network to the end user through the cloud). When the data are not aggregated, the data privacy is performed directly with the normal encryption algorithms (symmetrical or asymmetrical). On the other hand, if the data must be aggregated, when they arrive at the cluster head, they are decrypted and then aggregated and finally the aggregation result is encrypted to be sent to the base station. These encryption and decryption processes

generate additional calculations that can consume a lot of network energy. To deal with this issue, we proposed a solution based on the use of homomorphic encryption algorithms. With such encryption algorithms, it is not necessary to decrypt the data to be aggregated because the aggregation is carried out immediately on the encrypted data without decrypting them.

By hypothesizing, we were able to identify the appropriate homomorphic cryptographic algorithms for the wireless sensor network. These algorithms must not use exponential computations in order to avoid the overflow the microprocessors' capacity of the sensor node while processing data. In addition, they must preferably be additively homomorphic since the aggregation operations often require addition rather than multiplication. Theoretically, the algorithm that best meets the criteria is the homomorphic algorithm by elliptic curves, but unfortunately its implementation on sensor nodes has not been successful due to problems related to mapping data to encrypt with an elliptic curve. The mapping to an elliptic curve requires the inverse operation when one wants to find a data from a point on an elliptic curve, which usually requires calculations including discrete logarithms difficult to resolve and that the elliptic curves used are defined on the finite fields. For these reasons of mapping, we have dropped out the elliptic curves to continue with another homomorphic encryption scheme called Domingo-Ferrer.

Domingo-Ferrer's homomorphic encryption requires that the initial message or data to be encrypted as an integer must be systematically split into several fragments. Therefore, the number of fragments is one of the parameters ensuring the security level of this algorithm. For the implementation carried out in this thesis, we considered four cases. For each case, 10,000 messages or data collected by a sensor node were sent to the cloud, in order to express the impact of the fragmentation of the initial message on the life of sensor node's battery. First, the data was not encrypted, so they were sent in plain text, after 10,000 messages, the battery level had dropped by 1%. Secondly, the messages were encrypted with two fragments initially, which has the effect of lowering the battery level by 6% after 10,000 messages have been sent. For the third and fourth cases, the messages were also encrypted with respectively three and four fragments of the messages before their encryption, the battery levels dropped by 7% and 8% respectively for these last two cases.

It's good to have the data, but what to do with it? We asked ourselves this question too, to answer that, we sought to turn them into something more useful as some information. Unlike data, information is found from the processed data. In order to apply what was done previously, we thought about a smart city application that consists of the monitoring of electric transmission towers. By fixing a sensor node embedding the tri-axial accelerometer on a tower, we were able to determine the tilt angle between the transmission tower and the ground. The x, y and z components given by the tri-axial acceleration allowed us to establish a mathematical relation to calculate this angle of inclination. According to the RADEEF (Régie Autonomie intercommunale de Distribution de l'Eau et d'Electricité de Fès), it has the ways to know if an electric cable is broken and exactly in which place the breakage is located but it does not have tools to know whether a tower or an electric pole is dangerously inclined or even fallen. To validate this application, we compared the angles calculated by this application and the measured angles; we found that the uncertainty is of the order of 1 °.

Keywords: Wireless Sensor Networks, Cloud Computing, Smart Cities, Homomorphic Encryption, Distributed System.

Résumé

Cette thèse considère l'intégration des réseaux de capteurs sans fil avec le Cloud Computing (ou l'informatique en nuage). Les questions posées sont les suivantes : comment stocker les données issues du réseau de capteurs sans fil malgré les lacunes de stockage dans ce type de réseau? Comment réussir à intégrer deux réseaux hétérogènes (le réseau de capteurs sans fil et le réseau traditionnel comme l'internet) pour qu'ils puissent communiquer en entre eux? Si le cloud computing offre la meilleure solution de puissance de stockage et de calcul, comment pouvons-nous assurer la confidentialité des données depuis le réseau de capteur sans fil jusqu'à l'utilisateur final en passant par le cloud? Que faire avec une multitude des données diverse malgré qu'on a une puissance de calcul grâce au cloud? En bref, cette thèse contribue à la résolution de ces problèmes.

Pour répondre à la première question, la solution proposée dans cette thèse consiste à profiter les services offerts par le cloud computing à savoir la puissance de calcul et de stockage. Dans un réseau de capteur sans fil, il peut y avoir des milliers de nœuds capteurs collectant des multiples grandeurs physiques et qui sont convertis en données précieuses pour les diverses applications. Ces données sont collectées régulièrement en grande quantité de façon qu'il est quasi-impossible de les stocker localement dans le réseau car chaque nœud capteur ne dispose que quelques kilooctets de RAM et quelques mégaoctets de stockages de masse et moins de puissance de calcul (un nœud capteur embarque un processeur généralement de 8 à 16 bits). Ces données sont souvent exploitées en temps réel mais quoi faire lorsqu'elles viennent d'être utilisées et que les données fraîches arrivent ? Il est souhaitable de les stocker ou les archiver quelque part pour l'utilisation futur comme dans la prédiction, statistiques etc. Dans cette thèse, on propose l'intégration des réseaux de capteur avec les infrastructures du cloud afin de répondre à la question posée ci-haut.

Un autre problème que cette thèse a pu traiter est en rapport avec l'hétérogénéité des réseaux, d'un côté nous avons le réseau de capteur sans fil et de l'autre côté c'est en général l'internet ou l'intranet. Dans le réseau de capteurs sans fil les protocoles réseaux nécessaires pour envoyer des données d'un nœud à un autre ne sont pas les mêmes protocoles qu'on utilise sur l'internet. Nous avons proposé une solution architecturale divisant l'ensemble du système en différents couches physiques et logiques. Logiquement, le réseau de capteur est accédé à travers une passerelle (connectée directement à la station de base du réseau de capteur sans fil) depuis le réseau externe, donc cette passerelle est un point intermédiaire entre le réseau de capteur sans fil et le cloud ainsi que les utilisateurs finaux. Pour que, le réseau de capteur sans fil, le cloud ainsi que les utilisateurs finaux puissent communiquer, nous avons recouru à l'utilisation d'une technologie interopérable (middleware) dont son choix a été l'objet d'une étude comparative entre les technologies d'interopérabilité les plus utilisées. Java RMI a été identifié comme meilleur middleware pour l'architecture proposé contre CORBA et les web services.

Les nœuds capteurs dans le réseau sont souvent éparpillés et sont proches les eux des autres pour qu'ils puissent communiquer directement entre eux. Il arrive que des capteurs se trouvant dans le même milieu captent les valeurs identiques ou quasi-similaires, ce qu'engendre le problème de duplication des données. L'une des techniques plus utilisés dans le réseau de capteurs sans fil pour éliminer la duplication des données consiste à agréger les données d'un groupe ou grappe (cluster) des nœuds capteurs en un point de concentration local (tête de grappe ou cluster header) pour ne transmettre qu'une valeur agrégée (comme la moyenne par exemple). Cette thèse s'intéresse aussi sur la confidentialité des données qui peuvent être agrégées ou non

tout le long du système (depuis le réseau de capteurs sans fil jusqu'à l'utilisateur final en passant par le cloud). Lorsque les données ne sont pas agrégées, la confidentialité des données se fait directement avec les algorithmes de chiffrement normales (symétriques ou asymétriques). Par contre, si les données doivent être agrégées, lorsqu'elles arrivent à la tête de grappe, elles sont déchiffrées et puis agrégées et enfin le résultat d'agrégation est chiffré pour être envoyé vers la station de base. Ces processus de chiffrement et de déchiffrement engendrent des calculs supplémentaires qui peuvent être coûteux en terme d'énergie du réseau. Pour remédier à cela, nous avons proposé une solution basée sur l'utilisation des algorithmes de chiffrement homomorphe. Avec des tels algorithmes de chiffrement, il n'est pas nécessaire de déchiffrer les données pour être agrégées car l'agrégation se fait immédiatement sur les données chiffrées sans les déchiffrer.

En posant des hypothèses, nous avons pu identifier les algorithmes de chiffrement homomorphes adéquats pour le réseau de capteurs sans fil. Ces algorithmes ne doivent pas utiliser les calculs exponentiels pour ne pas saturer voire même déborder la capacité de traitement des microprocesseurs des nœuds capteurs et ils doivent être de préférence additivement homomorphe car les opérations d'agrégation nécessitent souvent de l'addition que de la multiplication. Théoriquement, l'algorithme répondant au mieux aux critères posés est l'algorithme de chiffrement homomorphe par les courbes elliptiques, mais malheureusement son implémentation sur des nœuds capteurs n'a pas été une réussite suite aux problèmes liées mappage des données à chiffrer sur une courbe elliptique. Le mappage vers une courbe elliptique nécessite l'opération inverse lorsqu'on veut retrouver une donnée à partir d'un point d'une courbe elliptique, ce que nécessite les calculs incluant les logarithmes discrets difficile à retrouver puis que les courbes elliptiques utilisées sont définies sur les corps finis. Pour ces raisons de mappage, nous avons abandonné les courbes elliptiques en faveur d'un autre schéma de chiffrement homomorphe dit Domingo-Ferrer.

Le chiffrement homomorphe de Domingo-Ferrer exige que le message initial ou une donnée à chiffrer sous forme d'un nombre entier doit être scindé systématiquement en plusieurs fragments de nombre. Donc le nombre de fragment est l'un des paramètres déterminant le niveau de sécurité de cet algorithme. Pour l'implémentation effectuée dans cette thèse, nous avons considéré 4 cas. Pour chaque cas 10 000 messages ou données collectées par un nœud capteur ont été envoyés vers le cloud, dans le but d'exprimer l'effet de la fragmentation du message initial sur l'état de la batterie du nœud capteur. Premièrement, les données n'étaient pas chiffrées, donc elles ont été envoyées en texte clair, après 10 000 messages, le niveau de la batterie avait chuté de 1%. Deuxièmement, les messages étaient chiffrés avec deux fragments au départ, ce qui a pour l'effet de faire baisser le niveau de la batterie de 6% après que 10 000 messages aient été envoyés. Pour le troisième et le quatrième cas, les messages étaient aussi chiffrés avec respectivement trois et quatre fragments des messages avant leur chiffrement, les niveaux de la batterie ont chuté de 7% et 8% respectivement pour ces deux derniers cas.

C'est bien d'avoir les données, mais qu'est-ce qu'il faut faire avec ? Nous nous sommes posé cette question aussi, pour répondre à cela, nous avons cherché à les transformer en quelque chose plus utile comme une information quelconque. A la différence des données, l'information c'est le résultat de traitement des données. Afin d'appliquer ce qui a été fait précédemment, nous avons pensé à une application de smart city qui consiste à la surveillance des tours de transmission électrique. En fixant un nœud capteur embarquant l'accéléromètre triaxial sur un tour, nous avons pu déterminer l'angle d'inclinaison entre le tour et le sol. Les valeurs x , y et z données par l'accélération triaxial nous ont permis d'établir une relation mathématique pour calculer cet angle d'inclinaison. D'après la RADEEF (Régie Autonomie intercommunale de Distribution de l'Eau et d'Electricité de Fès), elle a les moyens pour savoir si un câble électrique est cassé et à quel endroit exact se trouvant la coupure mais elle n'a pas des moyens pour savoir

un tour ou un poteau électrique est dangereusement incliné voire même tombé. Pour valider cette application, nous avons comparé les angles calculés par cette application et les angles mesurés, nous avons constaté que l'incertitude est de l'ordre de 1°.

Mots clés : Réseaux de capteurs sans fil, Cloud Computing, Villes Intelligentes, Chiffrement Homomorphe, Systèmes distribués.

Table of Contents

Acknowledgements	i
Dedications	iii
Abstract	v
Résumé	vii
Table of Contents	xi
List of Tables	xvii
List of Figures	xix
List of Acronyms and Abbreviations	xxiii
General Introduction	1
I. Motivation of Integrating the WSN with the Cloud Computing	2
II. Research Problem and Methodology	3
III. Contribution of this Thesis	4
IV. Document Organization	5
Chapter -1- Overview of the Wireless Sensor Networks	7
1.1. Introduction	7
1.2. Structure of the Wireless Sensor Node	8
1.3. Base Station and Sink Node of the WSN	8
1.4. Designs and Architectures of the WSN	8
1.4.1. Flat Architecture	9
1.4.2. Hierarchical Architecture	10
1.5. Characteristics of the Wireless Sensor Networks and Applications	10
1.6. Data Aggregation in Wireless Sensor Networks	11
1.6.1. Data Aggregation Techniques	11
1.6.1.1 Data Aggregation in Flat Networks	12
1.6.1.2 Data Aggregation in Hierarchical Networks	12
1.6.2. Challenges in Data Aggregation	14
1.7. WSN Attacks and Security requirements	15
1.7.1. Safety Hurdles in Wireless Sensor Networks	15
1.7.2. Security Requirements for the WSN	16
1.7.3. Frequent Attacks in the WSN	22
1.8. Applications of Wireless Sensor Networks	24
1.8.1. Precision Agriculture	24

1.8.2. Environmental Monitoring.....	25
1.8.3. Vehicle Tracking.....	27
1.8.4. Health Care Monitoring	27
1.8.5. Smart Buildings	28
1.8.6. Military Applications	28
1.8.7. Animal Tracking	29
1.9. WSN towards Internet of Things.....	29
Chapter -2- Cloud Computing Concept	33
2.1. Introduction	33
2.2. The origins of Cloud Computing.....	34
2.3. Reasons for migration to the Cloud Computing.....	34
2.3.1. Optimize the IT budget	35
2.3.2. Improve the work of users	35
2.3.3. Access data anywhere and anytime	35
2.3.4. Secure company's data	35
2.4. Desired features of Cloud Computing.....	36
2.5. Service models of the Cloud Computing.....	37
2.5.1. Software as a Service (SaaS)	38
2.5.2. Platform as a Service (PaaS).....	38
2.5.3. Infrastructure as a Service (IaaS).....	38
2.6. Deployment Models of the Cloud Computing	39
2.6.1. Public Cloud.....	39
2.6.2. Private Cloud	40
2.6.3. Community Cloud.....	40
2.6.4. Hybrid Cloud	41
2.7. Virtualization and the basis of the Cloud Computing	41
2.7.1. Virtualization Mechanism.....	42
2.7.2. Understanding Hypervisors	42
2.7.2.1 Type 1 Hypervisor Type	42
2.7.2.2 Type 2 Hypervisor Type	43
2.7.3. Types of Virtualization	44
2.7.3.1 Full Virtualization	44
2.7.3.2 Para-Virtualization	44
2.7.3.3 Isolation.....	45
2.7.4. Advantages of Virtualization	46

2.8. Cloud Challenges and Risks	47
Chapter -3- Background and Literature Review	49
3.1. WSN and Internet Integration Approaches	49
3.1.1. Front-End Approach	49
3.1.2. Gateway Approach.....	50
3.1.3. TCP/IP Approach.....	50
3.2. Infrastructural Issues of the Integration.....	50
3.3. Security Challenges	51
3.3.1. WSN-specific Security Challenges.....	51
3.3.2. Integration security challenges	52
3.4. Security Achievements and Open Issues.....	53
3.4.1. Securing Wireless Sensor Networks.....	53
3.4.2. Securing the Integration Strategies	54
3.4.2.1 Front-End Solution.....	54
3.4.2.2 TCP/IP Solution	55
3.4.2.3 Gateway Solution	55
3.5. Related Works to the WSN - Cloud Integration.....	56
3.5.1. Integrating Wireless Sensor Networks with Cloud Computing.....	56
3.5.2. WSN Integrating with Cloud Computing for Patient Monitoring	57
3.5.3. Cloud Computing System Based on Wireless Sensor Network	58
3.5.4. Integrating WSN into Cloud services for real-time data collection.....	58
3.5.5. Combining Cloud Computing and Wireless Sensor Networks.....	59
3.5.6. Synthesis of the Proposed WSN-Cloud Architectures.....	60
Chapter -4- Integration of the WSN with the Cloud Computing	63
4.1. Distributed System based on Cloud and WSN.....	63
4.1.1. Overview of Distributed System Architecture.....	64
4.1.2. Main Characteristics of a Distributed System	65
4.1.3. Middleware as the basis of the Distributed System	67
4.1.4. Comparative Survey between Technologies of Interoperability	67
4.1.5. Presentation of Java Remote Method Invocation	70
4.1.5.1 RMI Architecture	70
4.1.5.2 Implementation of RMI.....	70
4.1.5.3 Running an RMI program	71
4.1.5.4 RMI Registry.....	71
4.2. The Proposed WSN-Cloud Architecture	71

4.2.1. Physical Architecture	71
4.2.2. Logical Architecture	74
4.2.3. System Design and Algorithms	75
4.2.3.1 Definition of Remote Interfaces	75
4.2.3.2 Registration in the RMI name registry	78
4.2.3.3 Implementing Intended Features and Services	79
4.3. Conclusion	84
Chapter -5- Data Privacy of the Integrated WSN-Cloud	87
5.1. Homomorphic Encryption and Data Privacy	88
5.1.1. Brief History of Homomorphic Encryption	88
5.1.2. Homomorphic Encryption and Wireless Sensor Networks	90
5.1.3. Homomorphic Encryption applied to the Cloud Computing	91
5.1.4. Homomorphic Encryption on Aggregated data	92
5.2. Modelling Context and General Hypotheses	95
5.3. EC-ElGamal Cryptosystem	97
5.4. Domingo-Ferrer's Cryptosystem	98
5.4.1. Implementation of Domingo-Ferrer Encryption Scheme	101
5.4.2. Performance Analysis of the Domingo-Ferrer's Implementation	102
5.5. Discussion and Conclusion	105
Chapter -6- Implementation of the WSN-Cloud Architecture (Case Study: Smart Cities) ..	107
6.1. Smart Cities and the Wireless Sensor Networks	107
6.1.1. Understanding Data and Information in the Smart Cities	108
6.1.2. Sensors at the basis of the Smart Cities	109
6.1.2.1 Noise Pollution Sensing	110
6.1.2.2 Air Quality Sensing	110
6.1.2.3 Structural Health Monitoring	112
6.1.2.4 Smart Garbage Collecting	112
6.1.2.5 Smart Street Lightning	113
6.2. Accelerometer-based Wireless Sensor Network for Tilt Sensing	114
6.2.1. Designing of Accelerometer-based Wireless Sensor Network	115
6.2.1.1 Network Kit and Coverage	115
6.2.1.2 Power Optimization and Latency	116
6.2.2. Hardware and Calculation of the Tilt Angle	118
6.2.3. Software and Tilt Angle Algorithms	120
6.2.3.1 Data Sensing	120

6.2.3.2 Data Acquisition and Parsing.....	121
6.2.3.3 Data Processing.....	122
6.3. Experimental Results.....	122
6.4. Discussion and Conclusion.....	124
Conclusion and Future Work	125
I. Summary of Contributions	125
II. Future Work.....	126
Appendix A Results of the Tilt Sensing Experiment in Java 3D	127
Appendix B Scientific Activities.....	135
I. Regional and International Publications	135
II. International Conferences	135
REFERENCES.....	137

List of Tables

Table 1 Comparison between presented architectures related to this thesis	61
Table 2 The experiment configuration of Middleware Comparative Survey	68
Table 3 Comparison of Homomorphic Cryptosystems.....	96
Table 4 Sample of encrypted data received after Domingo-Ferrer’s Implementation.....	101
Table 5 Experiment configuration and range of varying parameter values	102
Table 6 Summary of the experiment results.....	105
Table 7 Calculations of the accuracy of the reading and the acceleration resolution depending on the range	119
Table 8 Summary of the experiment about rotating Waspmote core board around the Y-axis	119
Table 9 Tilt sensing experiment results showing sensed and measured angles	122

List of Figures

Figure 1 Wireless Sensor Network.....	7
Figure 2 Sensor node structure.....	8
Figure 3 Diagram illustrating WSN architectures.....	9
Figure 4 Taxonomy of Data Aggregation based on Network Topology.....	11
Figure 5 A Clustered Wireless Sensor Network.....	12
Figure 6 The Chain based organization in a Wireless Sensor Network.....	13
Figure 7 The Tree based Data Aggregation in the WSN.....	13
Figure 8 Grid based Data Aggregation in Wireless Sensor Networks[7].....	14
Figure 9 Symmetric Cryptosystem Diagram.....	17
Figure 10 Asymmetric Cryptosystem Diagram.....	17
Figure 11 Message Authentication Code diagram.....	19
Figure 12 Digital Signature diagram.....	20
Figure 13 Applications of the Wireless Sensor Networks.....	24
Figure 14 Internet of Things (IoT) Concept.....	30
Figure 15 Convergence of various advances leading to the advent of cloud computing.....	34
Figure 16 The cloud computing stack.....	37
Figure 17 Types of clouds based on services.....	37
Figure 18 Public cloud deployment model.....	39
Figure 19 Private cloud deployment model.....	40
Figure 20 Community Cloud Deployment Model.....	40
Figure 21 Hybrid cloud deployment model.....	41
Figure 22 Type 1 hypervisors run directly on bare metal.....	43
Figure 23 Type 2 hypervisors run within an operating system environment.....	43
Figure 24 Full virtualization.....	44
Figure 25 Para-virtualization.....	45
Figure 26 Isolator.....	45
Figure 27 WSN and Internet integration strategies.....	49
Figure 28 Sensor-Cloud Integration Framework [100].....	56
Figure 29 Layer-based WSN-Cloud integration for real-time monitoring [103].....	58
Figure 30 Cloud computing based sensor data analysis environment [104].....	59
Figure 31 A Distributed System.....	65
Figure 32 Time required to invoke remote method.....	68

Figure 33 Memory used while invoking remote method	69
Figure 34 Consumed bandwidth while invoking remote method	69
Figure 35 Java RMI Architecture	70
Figure 36 Illustration of RMI Registry process	71
Figure 37 Physical WSN-Cloud Architecture	72
Figure 38 Hypervisor Type-1 based Cloud Infrastructure	73
Figure 39 Logical WSN-Cloud Architecture	74
Figure 40 Diagram of the Gateway Interface Definition and Implementation	76
Figure 41 Diagram of the Cloud Interface Definition and Implementation.....	77
Figure 42 Diagram of the EndUser Interface Definition and Implementation	77
Figure 43 Registration sequence of remote objects to the RMI registry.....	78
Figure 44 The flowchart of the Gateway's Algorithm.....	80
Figure 45 The flowchart of the Cloud Infrastructure's Algorithm	81
Figure 46 The flowchart of the End User's Algorithm.....	82
Figure 47 Logical architecture showing data on-demand service.....	83
Figure 48 The concept of level sensing of the sewage water.....	84
Figure 49 Homomorphic encryption on aggregated data in the integrated Cloud-WSN.....	94
Figure 50 Example of a graph representing the elliptic curve $y^2 = x^3 - 4x + 7$	97
Figure 51 Encryption and decryption using Domingo-Ferrer's cryptosystem.....	99
Figure 52 Tree-based data aggregation using Domingo-Ferrer's cryptosystem	100
Figure 53 Configuration of the Domingo-Ferrer's Implementation	101
Figure 54 Battery status without Domingo-Ferrer homomorphic encryption.....	103
Figure 55 Battery state with Domingo-Ferrer homomorphic encryption using 2 message fragments.....	103
Figure 56 Battery state with Domingo-Ferrer homomorphic encryption using 3 message fragments.....	104
Figure 57 Battery state with Domingo-Ferrer homomorphic encryption using 4 message fragments.....	104
Figure 58 Wireless Sensor Network for Noise Sensing.....	110
Figure 59 Wireless Sensor Network for Particle Pollution Sensing	111
Figure 60 Wireless Sensor Network for Structural Health Monitoring	112
Figure 61 Wireless Sensor Network for Garbage Level Sensing.....	113
Figure 62 Wireless Sensor Network for Street Lighting.....	114
Figure 63 Wireless Sensor Network for Tilt Sensing	116
Figure 64 Introducing clustering into accelerometer-based wireless sensor network	117
Figure 65 Accelerometer Sensor embedded in Waspote (PRO 1.2) Core Board	118

Figure 66 Accelerometer Axes' Orientation in Waspote (PRO 1.2) Core Board, communication module plugged in).....	118
Figure 67 Flowchart of the accelerometer's data sensing	121
Figure 68 Graph comparing the sensed angle and the measured angle	123
Figure 69 Tilt sensing experiment's configuration	123
Figure 70 Screenshot of the end user application reproducing the 3D scene of the 75° tilted tower (See all screenshots in Appendix A)	124

List of Acronyms and Abbreviations

AAA	Authentication, Authorization and Accounting
AES	Advanced Encryption Standard
API	Application Programming Interface
ARQ	Automatic Repeat reQuest
ATM	Asynchronous Transfer Mode
AWS	Amazon Web Services
AWT	Abstract Window Toolkit
CED	Centre d'Etudes Doctorales
CMT	Castelluccia Mykletun Tsudik
COM	Communication Port
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
CSV	Comma-Separated Values
DGHV	Dijk Gentry Halevi Vaikuntanathan
DPU	Data Processing Unit
DRP	Disaster Recovery Plan
DTN	Delay-Tolerant Network
EBS	Elastic Block Store
EC2	Amazon Elastic Compute Cloud
ECC	Elliptic Curve Cryptography
ECDH	Elliptic-Curve Diffie–Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme

EEPROM Electrically-Erasable Programmable Read-Only Memory

ERP Enterprise Resource Planning

FEC Forward Error Correction

FFT Fast Fourier Transform

GAE Google App Engine

GIOP General Inter-ORB Protocol

GPRS General Packet Radio Service

GPS Global Positioning System

GSM Global System for Mobile Communications

GUI Graphical User Interface

HTTP Hypertext Transfer Protocol

IAMU Identity and Access Management Unit

ICT Information and Communication Technology

IDS Intrusion Detection Systems

IEEE Institute of Electrical and Electronics Engineers

IIOP Internet Inter-ORB Protocol

IP Internet Protocol

IT Information Technology

JAX-RPC Java API for XML-based RPC

JDK Java Development Kit

JOGL Java OpenGL

JSON JavaScript Object Notation

JVM Java Virtual Machine

KB Kilobyte

KMS Key Management System

LAN Local Network Area

LCD	Liquid Cristal Display
LED	Light Emitting Diode
LIAN	Laboratoire d'Informatique, d'Imagerie et d'Analyse Numérique
LINQ	Language Integrated Query
LSB	Least Significant Bit
MAC	Media Access Control
MIC	Message Integrity Code
MODA	Multi-functiOnal secure Data Aggregation
NFC	Near Field Communication
NIST	National Institute of Standards and Technology
OSI	Open Systems Interconnection
PKC	Public Key Cryptography
PKI	Public Key Infrastructure
PSK	Pre-Shared Keys
RADEEF	Régie Autonome intercommunale de Distribution d'Eau et d'Electricité de Fès
RAM	Random Access Memory
RDP	Remote Desktop Protocol
REST	Representational State Transfer
RFID	Radio Frequency Identification
RMI	Remote Method Invocation
ROM	Read Only Memory
RRL	Remote Reference Layer
RSA	Rivest, Shamir and Adleman
SHA	Secure Hash Algorithm
SHM	Structural Health Monitoring
SKE	Symmetric Key Encryption

SMS	Short Message Service
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Sockets Layer
STIC	Sciences et Technologies de l'Information et de la Communication
SWT	Standard Widget Toolkit
TCP	Transport Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
URL	Uniform Resource Locator
USB	Universal Serial Bus
WAP	Wireless Application Protocol
WBSN	Wireless Body Sensor Network
WHO	World Health Organization
WIFI	Wireless Fidelity
WSDL	Web Services Description Language
WSN	Wireless Sensor Network
XML	eXtensible Markup Language

General Introduction

This research work is doctoral dissertation in the field of Wireless Sensor Networks integrated with Cloud Computing. It is a work of study and application for doctoral training in Information and Communication Sciences and Technologies (Sciences et Technologies de l'Information et de la Communication, STIC). It was carried out in Laboratory of Informatics, Imaging and Numerical Analysis (Laboratoire d'Informatique, d'Imagerie et d'Analyse Numérique, LIAN) at the Faculty of Sciences Dhar El Mahraz of the Sidi Mohamed Ben Abdellah University located in Fez/Morocco. The main purpose of this thesis is to explore various interoperable technologies concerning the interconnection of Wireless Sensor Networks and Cloud infrastructure, in order to implement a secure software architecture based on a distributed system and applicable in the field Smart Cities.

For some decades, the need to observe and control physical phenomena such as temperature, pressure or brightness is essential for many industrial and scientific applications. In the field of ecology, monitoring pollutants could significantly increase the quality of life in cities. Not so long ago, the only solution for routing sensor data to the central controller was cabling that had the major drawbacks of being expensive and cumbersome. Wireless sensor networks are one of the technologies to solve the problems of this new era of embedded and ubiquitous computing. This new technology promises to revolutionize the way we live, work and interact with the physical environment around us. Wireless communicating sensors with computing capabilities facilitate a series of unrealizable or expensive applications a few years ago. Today, tiny, inexpensive sensors can be literally scattered over roads, structures, walls, or machinery, creating a kind of "second digital skin" that can detect a variety of physical and biological phenomena.

Nowadays, with the advanced research in the new information and communication technologies, it is very easy to develop small embedded systems of communication, not expensive, comprising at least a power unit, a data collection unit (sensing unit), a processing unit and a transmission unit for mainly surveillance, discovery and detection purposes [1]. These microsystems (sensor nodes or motes) can integrate a wide range of sensors collecting information from the physical environment for various applications such as military, medical, industrial, natural disasters, etc. [2].

Thanks to these transmission units, these nodes communicate with each other to build a network, often wireless, which can generate thousands of nodes transforming the state of an observed physical quantity (temperature, pressures, humidity, vibration, etc.) into signals that can be converted into codes in order to be processed by computers. A wireless sensor network is a set of nodes integrating the sensors controlled by a main node called the *base station* or *sink node*. These sensor nodes cooperate together to perform observation and sensing tasks, then the transfer of the sensed data to the sink node which constitutes an output gateway to the rest of the digital world. Given the diversity of applications, the deployment of this set of sensor nodes is often random. It can also concern dangerous areas that are not controlled or require mobility. This results in a dynamic topology, unsecured and indeterminate in space and time. All these constitute challenges to overcome in order to achieve the communication between sensor nodes, essentially the routing and broadcast tasks.

The random deployment of wireless sensor networks can induce high redundancy between the nodes. This redundancy represents for the sensor networks a characteristic that is both profitable

and attractive. It can be used to improve the reliability of the detection or the accuracy of the data collected. On the other hand, it generates a larger data transfer and involves an additional traffic load. Aggregation of data is a very interesting approach to reduce the load on communications. It consists of processing the data collected by each sensor at a node called aggregator. Only the result produced will be transmitted to the base station. In this way the amount of data communicated in the network can be decreased, which consequently reduces the bandwidth consumption and the energy depletion of the sensors. However, ensuring security together with aggregation techniques is very difficult because a captured node poses a double problem. It compromises the confidentiality of the data (possibility of listening) and their availability (possibility of attack of the denial of service type). Also a compromised aggregation node endangers all collected metrics that are part of the aggregate whose node is responsible. The use of homomorphic encryption is a powerful tool since the aggregator will aggregate encrypted data which it does not have access to, which has the advantage of ensuring confidentiality even if the aggregator has been compromised. The use of a clustered architecture of the network contributes to the reduction of the overhead (number of transmitted and received messages), which has the effect of saving the energy of the sensors and thus prolonging the service life of the sensors network.

On the other hand, data aggregation minimizes the effect of redundancy in terms of the amount of data flowing through the network, but the wireless sensor network still lacks enough computing and storage power. These precious data from the wireless sensor networks are available in multiple natures and huge size so that it will not be an easy task to exploit them in real time, they need to be stored, processed and then transformed into useful information for end users. This lack of computing power, processing and storage can be found elsewhere outside the wireless sensor network as in data centers of a cloud computing. The integration of the wireless sensor network with the cloud is a solution to overcome the aforementioned challenges in order to provide users with information in the form of services across the cloud.

Cloud computing offers elastic provisioning of large-scale infrastructure to multiple and simultaneous users [3]. Through its three service models namely infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS), cloud computing effectively dissociates applications and services from the deployed infrastructure [4] and allows several actors to use it as and when needed. Service providers use platforms (offered as PaaS) to provide applications and services offered in SaaS, pay-per-view, end-user or other applications. Platforms facilitate the procurement process by adding abstraction levels to the IaaS infrastructure. Infrastructure is the real dynamic pool of resources used by applications. Cloud computing has several inherent advantages, such as resource utilization, scalability, elasticity, and rapid development and introduction of new applications. Cloud computing uses the established concept of virtualization to provide simultaneous access to resources via abstractions. Virtualization allows users (applications and services) to use the resources dedicated to them when in reality, several users access it simultaneously [5]. This is usually done by dedicated software such as a hypervisor or middleware.

I. Motivation of Integrating the WSN with the Cloud Computing

There are two new technologies, in particular, that are beginning to revolutionize not only the use of applications and services, but also the way they are provisioned. These technologies are Internet-of-Things (IoT) and Cloud Computing. IoT is called as the next technological paradigm that aims to achieve communication between many types of objects, machines and devices on a large scale [3]. IoT should have a profound impact on our daily lives, even bigger than the Internet, which would raise many challenges [4]. WSN can be considered as one of the basic components of IoT because it can help users (humans or machines) to interact with their

environment and react to real events. IoT not only allows the plurality of heterogeneous devices to connect and communicate with each other, but also allows application and service providers to offer new applications and services.

To manage, track and provide services around these connected objects, this involves creating applications. These new solutions must have the capacity to handle billions of devices, store and analyze zettabytes of data in an evolutionary, resilient and very resilient way. The IoT users need to take advantage of the omnipresent aggregation points in the cloud, all in a cost-effective way. They require development logic, using infrastructure automation and distributed architectures, such as those that power large-scale Web applications. These applications are delivered to users as a cloud service, called "native cloud".

Connected objects are often equipped with at least one network interface to communicate on traditional networks, making them easier to integrate with cloud computing. Manufacturers and suppliers of these connected objects provide to their customers the software, services and storage memories that are easily configurable and ready to use, so IoT is better developed and better advanced than the wireless sensor networks. *Currently, most researches in the Wireless Sensor Networks have focused on the internal network but the interaction with external networks and modern communication technologies (Cloud Computing, Cloud Messaging, etc.) remains less explored.* In fact, Cloud infrastructure offers several advantages such as mass storage, the demand for self-service access to the network wide resource consolidation, the measurement service, massive scalability, consistency, virtualization, low cost software, distribution, orientation of service and advanced security. All these qualities are essential to fill the gaps found in the wireless sensor networks.

II. Research Problem and Methodology

The topic of this thesis is to provide a secure integration of wireless sensor networks with cloud computing, but talking about that, we already hear two different types of networks, one the sensor network and the other the internet or intranet. Communicating these two networks is a challenge for this thesis as the constraints are not similar for these two types of network. We must find an effective way to integrate them and ensure their mutual security. When we talk about cloud computing, we tend to make life easier as a result of the services offered by this technology, the integration of the wireless sensor network with the cloud will not change the philosophy of the cloud by giving only the data to users but also information from the data.

To communicate two types of network with different communication protocols, it is first necessary to find an interoperable technology to connect them by considering the constraints of each network. In the previous section, it is written that the data collected by all the sensor nodes of the network are collected at a point called sink node, it is through this node sink that the sensor network can communicate with the external networks. So, to communicate the wireless sensor network with the cloud, it is a must to design a physical architecture with a gateway between these two networks. This gateway must be chosen in such a way that it can communicate indifferently on both networks. Then, we must to think logically how the entities on the two different networks will communicate together.

As the wireless sensor networks generate a huge amount of data, the sensed data may have some similarities since the sensor nodes are often very close and scattered in a geographical area. So, if the sensor nodes are very close to each other, they logically capture very close or even identical values. If no preprocessing is done on the collected data, it will be sent to the cloud with unnecessary duplicates. A viable solution would be to do some type of processing that does not have to mitigate the accuracy and quality of the data. Such a solution can be a statistical function like the average, by collecting temperature data for example with three or four sensors,

we could send only the average to the cloud since the sensors are very close. In the wireless sensor network, such a solution is called *aggregation of data*. By choosing a sensor node that has enough energy and computing power than other sensor nodes, it could act as an aggregator node that aggregates data from other sensor nodes and sends the result to the sink node.

As we mentioned earlier, we have two types of network that must communicate together but with different constraints, ensuring data privacy is not an easy task in this case. There is a need to find ways to keep data confidential all the way from the WSN to the cloud and the end user. This concept of security must be designed so that it must include the aggregation of data. A perfect solution to this problem is to use a *homomorphic encryption algorithm*. Homomorphic encryption scheme allows performing arithmetic operations on the encrypted data without decrypting them and the result remains encrypted. Such an algorithm has the advantage of aggregating the encrypted data without decrypting them. So, each sensor node encrypts its data before sending it and the aggregator node gathers the received data, aggregates them and sends the result to the sink node. In this thesis, we consider that encryption and decryption keys should not be stored in the cloud for security reasons, the encryption keys are available in each sensor node and the decryption keys are kept secretly at the final user. So, if an end user has to perform complex calculations on the data stored in the cloud, he/she does not send the keys of decryption, the calculations are performed on the encrypted data and the results are calculated thanks to the homomorphic encryption. The end user receives only the encrypted results; he/she uses his/her decryption keys to find the result in plaintext. To implement this idea, it is first necessary to carry out a comparative study on homomorphic encryption algorithms by considering the level of security and constraints of WSNs.

It's good to collect the data but what to do with this data? The data from WSN are multiple and huge for an end-user, so it would be better to get value-added information to the end user. For example, a farmer does not need to know how much temperature he/she will be getting today or if the pressure level has dropped, what he needs is the weather forecasting. From the data of pressure, temperature, humidity, etc. it is possible to apply a model to forecast the rain, this is the information that the user needs to know if it will rain or not. So in this thesis, we applied the architecture integrating the wireless sensor network with the cloud in the smart city to monitor electrical transmission towers. By fixing acceleration sensors at each electrical pole, it is possible to apply a mathematical formula to the collected accelerometer data to find information relating to the inclination of each electrical pole. With this tilt angle, we can know if the post is dangerously tilted to cause the damage.

III. Contribution of this Thesis

Several research projects have been conducted to solve the problems of integrating wireless sensor networks with web-based cloud computing. We summarize below the contributions presented in this thesis.

- **Integrating WSNs with the Cloud:** We were interested in our work to study first if the web services used in other related research works are the most suitable for this integration. For this we have conducted a study for the different interconnection technologies considering that the WSN-Cloud architecture and the end users constitute a single distributed system. There are several ways to interconnect the components of the same distributed system, but we preferred to compare the interoperable middleware (CORBA, Java RMI and JAX-WS type web services) that are mostly used to find out which of them matches better the criteria and constraints of the such architecture. We found that in general Java RMI provides the best performance compared to two other

technologies in terms of time needed to invoke methods on remote objects and in terms of consumption of network resources. Thanks to the Java virtual machine, Java RMI can be run on several platforms, which competes CORBA in terms of integration of several programming languages. Web services are slow, consume a lot of memory and bandwidth, so we have classified the web services are not to use in the proposed architecture.

- **Data privacy for the Cloud-WSN architecture:** A common approach to overcome the limitations of sensor networks is to aggregate data at the intermediate node level. Thus the individual readings of each sensor are replaced by a global collaborative view on a given area. The aggregation of data in sensor networks is relatively trivial, but becomes problematic when you want to add security. Indeed, guaranteeing security together with aggregation techniques is difficult because a captured node poses a double problem. It compromises the confidentiality of the data (possibility of listening) and their availability (possibility of attack of the denial of service type). Also, a compromised aggregation node endangers the integrity of all the measures that are part of the aggregate for which the node is responsible. In our work, we do not consider the confidentiality of data as a major problem. The proposed architecture requires that data remains secure all the way from the WSN to the end user. For that we thought about homomorphic cryptography, by posing the hypotheses in relation to the constraints of the WSN, we could design the most adapted encryption algorithms to this architecture. We have found that public based-key additive homomorphic algorithms are the best classified for implementations on sensor nodes with less energy and computing resources. We have identified elliptic curve encryption (EC-ElGamal) as the ideal solution for our architecture but its implementation generates additional problems such as the resolution of discrete logarithms. For our implementation we adopted for the Domingo-Ferrer cryptosystem on sensor nodes embedding 16-bit microprocessors.
- **Validation of the WSN-Cloud Architecture:** To validate our architecture, we proceeded to the implementation. The implementation was about to use our architecture to provide the information to the end users. We applied our approach in the smart city in order to super-spirited the electric transmission tower. Using the data provided by the acceleration sensors (accelerometers fixed on the transmission towers) on three axes, we were able to calculate the angle of inclination between the towers and the ground. We have been able to find a formula to find the angle of inclination that is directly proportional to the Z component of the accelerometer. For the end user, the information he will get will be this tilt angle instead of three acceleration components. We were able to reproduce the 3D scene to facilitate real-time monitoring using the proposed WSN-Cloud architecture.

IV. Document Organization

This manuscript is organized in six chapters followed by a general conclusion. The positioning of our work is presented in the first three chapters and our contributions are detailed in the last three chapters.

In the first, the second and the third chapter, we present the general introduction to the concepts of integrating wireless sensor networks with cloud computing. We start by defining the various concepts that revolve around this theme, then we explain the motivation that drives us to carry

out research in this area. And finally we present the summary of the contributions made by this thesis.

In the second chapter, we first discuss the architectures, operations, security, data aggregation, and applications of wireless sensor networks. We continue with the overview of cloud computing, its service models and deployment models, and finally the concept of virtualization. The last section of this chapter is devoted to the bibliographic study on our research axis, namely, the integration of WSNs with the Cloud. We review the existing work that has been identified in the literature, performing a deep analysis, and an evaluation and comparison between the different solutions.

In the fourth chapter we start with the overview of distributed systems, then a comparative study of the most used middleware to determine the appropriate middleware for integration of WSN with the cloud. The next section of this chapter provides an architecture for integrating WSN with cloud based distributed systems.

The fifth chapter focuses on data privacy, primarily confidentiality using homomorphic cryptograms. This chapter first summarizes the history of homomorphic cryptosystems, then a short review of homomorphic cryptography applied to cloud computing and its application in wireless sensor networks. The next section of the fourth chapter focuses on the confidentiality of aggregated data in the wireless sensor network, we start by modelling a homomorphic encryption algorithm adapted to WSNs, then the implementation of homomorphic encryption on a sensor node and finally the performance analysis of the homomorphic encryption's implementation.

To implement our research contribution, we contributed to a case study in the field of Smart Cities. Indeed, the sixth chapter is dedicated to the tilt monitoring of electric transmission towers applicable in the city of Fez. In this chapter, we begin with the overview of Smart Cities, then the study and the design of a tilt detection system of the electric transmission towers and after this step we continue by the mathematical modelling of the tilt angle and end by the presentation of the results.

Chapter -1-

Overview of the Wireless Sensor Networks

1.1. Introduction

Since their inception, wireless communication networks have been increasingly successful in the scientific and industrial communities. Thanks to its various advantages, this technology has been able to establish itself as a key player in current network architectures. Wireless media offers unique properties, which can be summarized in three points: ease of deployment, ubiquity of information and reduced installation cost. During its evolution, the wireless paradigm has seen the emergence of various derived architectures, such as: cellular networks, wireless local area networks and others.

Nowadays wireless technologies are part of our everyday life. Whether it is from simple infrared remote controls to wireless internet, to bluetooth devices such as mobile phones, keyboard, etc. no modern house escapes the rule. On the market, we find more and more electronic devices able to communicate through the network, most often without wires, and we do not see what could slow down this development.

However, most of these wireless technologies, starting with Wi-Fi, are based on fixed infrastructures, limiting the mobility of users. To facilitate this mobility, there is another type of network, more and more common, which allows nodes of the network to communicate directly with each other without the need for infrastructure: these are ad hoc networks.

During the last decade, a new architecture has emerged: Wireless Sensor Networks. This type of network is the result of a fusion of two fields of modern computing: embedded systems and wireless communications. A Wireless Sensor Network (WSN), consists of a set of on-board processing units, called "motes" (also sensors nodes), communicating via wireless links. The general purpose of a WSN is the collection of a set of environmental parameters surrounding the motes, such as the temperature, vibration, humidity, earthquake or pressure of the atmosphere, and transform them into electrical signal in order to be proceeded by the computer.

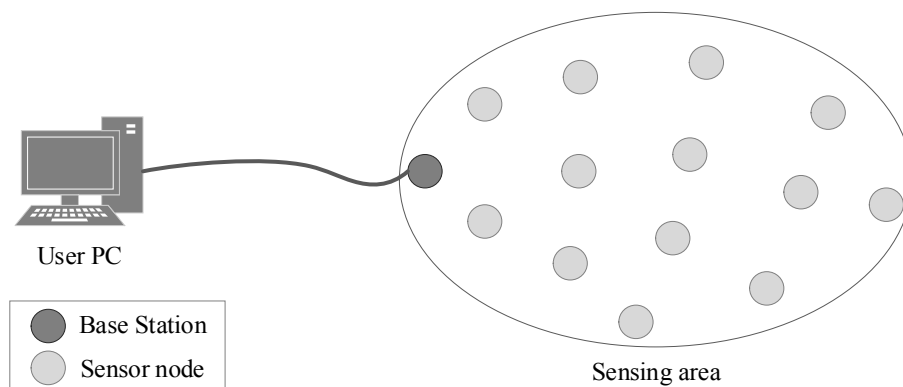


Figure 1 Wireless Sensor Network

As shown on the Figure 1 above, a wireless sensor network consists of three main components: sensor nodes, sink nodes, and software. The sensed data are wirelessly transmitted from one

sensor node to another until one or more collection powerful node called “sink node” (also sometimes called “base station”) that is located close to or inside the sensing region and connected directly to the wired world (usually a powerful computer such as a workstation) where the measured raw data can be processed, analysed, and presented by using software. The late workstation can serve as a gateway to the remote users situated over external networks (e.g. the internet) for storage, analysis and processing.

1.2. Structure of the Wireless Sensor Node

Sensors nodes are devices of extremely small size with very limited resources, autonomous, able to process information and transmit it, via radio waves, to another node over a limited distance a few meters away. Sensor networks use a large number of these sensor nodes to form a network without an established infrastructure. A sensor node analyses its environment, and spreads the collected data to sensor nodes belonging to its coverage area.

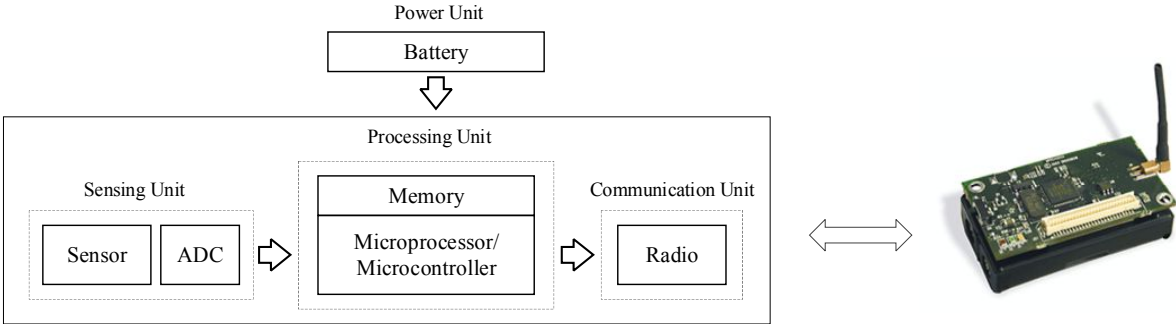


Figure 2 Sensor node structure

A sensor node contains four basic units: the sensing unit, the processing unit, the transmission unit, and the energy control unit. It can also contain, according to its field of application, additional modules such as a tracking system (GPS) or an energy generating system (solar cell). One can even find micro-sensors, a little more bulky, equipped with a mobilizing system charged to move the micro-sensor if necessary[1].

1.3. Base Station and Sink Node of the WSN

Unlike sensor nodes, the base station has many more computing power, more memory and is often connected to a better source of power than batteries (e.g., computer USB). The base station can be viewed as an entry point to the WSN where the main purpose of the base station is to gather data captured from the sensor nodes in WSN.

A wireless sensor network typically includes a large number of densely deployed sensor nodes in a region of interest, and one or more sinks or base stations located near or within the sensing zone serving as a gateway to external networks, for example the Internet. It collects sensor node data, performs simple processing on them, and then sends the relevant information (or processed data) over the external network (LAN, mobile networks) to the users who requested it or use the information[6].

It is possible, however, to have several fixed or mobile base stations in the network that can relay information, but for reasons of cost, the network typically has many more sensor nodes than base stations.

1.4. Designs and Architectures of the WSN

To send data to the base station, each sensor node may use a single-hop long distance link, which leads to the single hop network architecture, as shown in *Figure 3.a*). However,

transmission over long distances is expensive in terms of energy consumption. In wireless sensor networks, the energy consumed for communication is much higher than the energy required for sensing and calculation. In addition, the energy consumed for transmission dominates the total energy consumed for communication and the required transmission power increases exponentially with increasing transmission distance.

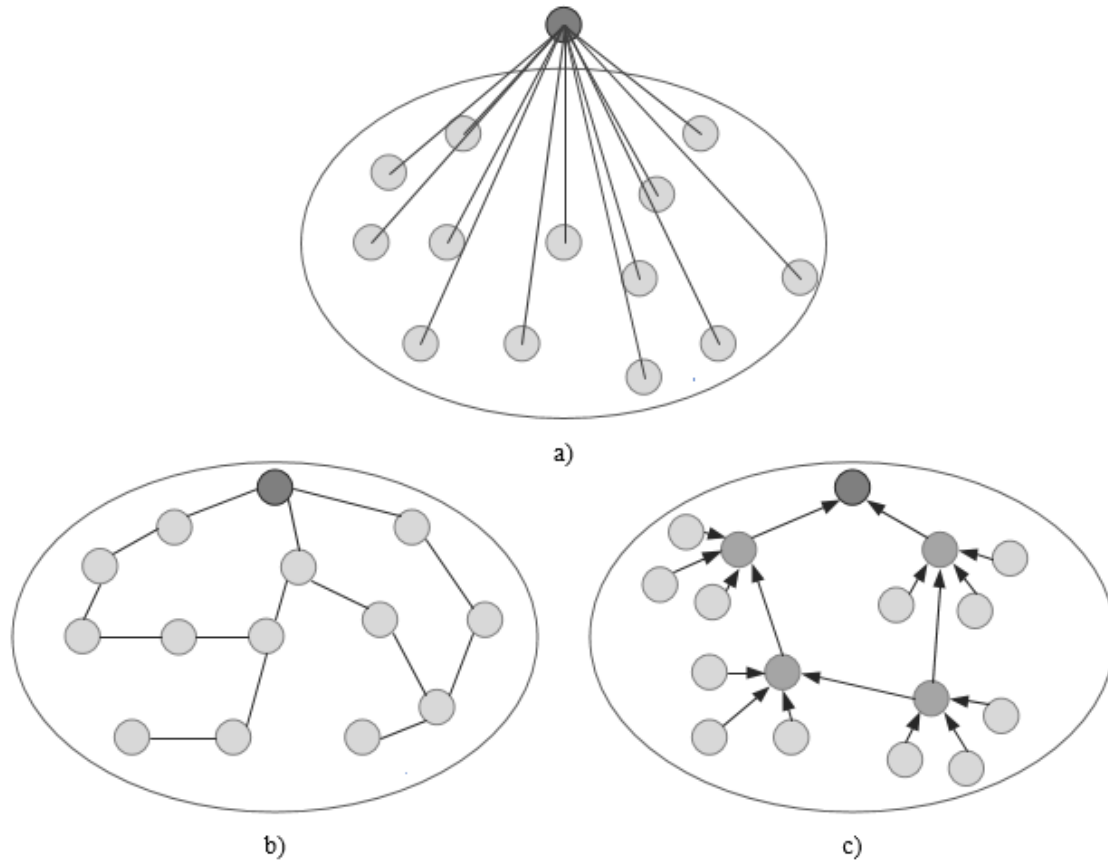


Figure 3 Diagram illustrating WSN architectures

Indeed, a short distance multi-hop communication is highly preferred in most wireless sensor networks where the sensor nodes are densely deployed and neighbouring nodes are close together, making it possible to use short range communications. In multi-hop communication, a sensor node transmits its collected data to the base station via one or more intermediate sensor nodes, which can reduce power consumption for communication. The architecture of a multi-hop network can be split in two types: flat and hierarchical.

1.4.1. Flat Architecture

In this type of network, each node execute the role of a sensing and all the sensor nodes are peers. Due to the large number of sensor nodes, it is not possible to assign a global identifier to each node in a sensor network. As a result, data collection is generally performed using data-centric routing, wherein the sink transmits a flood request to all the nodes of the sensing area and only the sensor nodes having the corresponding data respond. As shown in *Figure 3.b*), each sensor node communicates with the base station via a multi-access path and uses its peer nodes as relays.

1.4.2. Hierarchical Architecture

In a hierarchical network shown in the *Figure 3.c*), the sensor nodes are clustered where each sensor node sends its collected data to the cluster head, which in turn serves as a relay to the base station. Often, the nodes responsible for collecting the data have lower energy compared to the chosen nodes like the cluster heads which have in addition the data processing resources quite substantial. This process not only reduces power consumption, but also balances traffic load and improves scalability as network size increases.

When all the nodes of the network have the same capability of transmission, periodic clustering can be performed where each node becomes cluster head at a certain moment in order to ensure the balance load of the traffic. In addition, data can be aggregated at the head of the cluster to reduce the amount of data to be transmitted to the base station while effectively improving network energy[7].

1.5. Characteristics of the Wireless Sensor Networks and Applications

Sensor networks have significant advantages over conventional wired networks because they do not only reduce the cost and duration of deployment, but they are also applied to any environment especially in places where it seems impossible to establish wired connections for example on the battlefields.

- **Discovering of natural disasters:** You can create an autonomous network by dispersing nodes in the wild. Sensors can report events such as forest fires, storms or floods. This allows a much quicker and more efficient response of the rescues.
- **Intrusion detection:** By placing sensors at various strategic points, burglaries or wildlife passages can be prevented on a railway track (for example) without the need for costly video surveillance devices.
- **Business applications:** One could imagine having to store food requiring a certain humidity level and a certain temperature (min or max). In these applications, the network must be able to collect this information and alert in real time if the critical thresholds are exceeded.
- **Pollution control:** Sensors could be dispersed over an industrial site to detect and control gas or chemical leaks. These applications would provide the warning in record time and be able to follow the evolution of the disaster.
- **Agriculture:** Nodes can be incorporated in the soil. The network of sensors can then be questioned on the state of the field (for example, to determine the drier sectors in order to water them first). It is also possible to equip herds of cattle with sensors to know their position at any time, which would prevent breeders from using sheepdogs.
- **Medical monitoring:** By implanting under the skin of mini video sensors, one can receive real-time images of a part of the body without any surgery for about 24 hours. We can thus monitor the progression of a disease or the reconstruction of a muscle.
- **Control of buildings:** Sensors can be included on the walls of the dams to calculate the pressure exerted in real time. It is therefore possible to regulate the water level if the limits are reached. One can also imagine to include sensors between sacks of sand forming a makeshift dike. The rapid detection of water infiltration can be used to reinforce the dam accordingly. This technique can also be used for other constructions such as bridges, railways, mountain roads, buildings and other structures.

1.6. Data Aggregation in Wireless Sensor Networks

In networks, the term aggregation can be used in many situations. For example, [8] provides a solution based on packet aggregation to improve the performance of voice over IP applications. Packet aggregation consists of bringing together in a single package several smaller ones with the same destination. This technique makes it possible to reduce the total size of the packet headers and thus the proportion of time spent in transmission mode. This method also makes it possible to reduce the number of transmitted packets and the number of collisions (thus the need for retransmission), which offers opportunities for energy savings. Data aggregation is also a method conventionally used in networks. Mainly developed in sensor networks [9], it processes data as it is relayed by sensors to the collection point. The goal is to combine the data with each other to reduce the number of transmissions and thus the energy consumption of the network. Finally, there is also the aggregation of flows, used in particular in [10], to collect telemetry data in a network in an optimized way in terms of performance, monitoring or security.

The sensor nodes are often deployed in large numbers in a specific area where the different neighbouring sensor nodes may sense similar or even redundant data on a specific phenomenon which cause excessive transmissions and as the high consumption of energy. Because the sensor nodes are powered by the batteries, it is essential to perform each operation efficiently on the energy plan. In this case, it is preferable that a sensor node removes the redundant data received from its neighbour nodes before transmitting them to the base station. Data aggregation is an effective technique for reducing redundant data while effectively saving energy in wireless sensor networks. The basic trick during the aggregation process is to assemble data received from different sources to minimize redundancy of data and to mitigate transmission power consumption[1].

1.6.1. Data Aggregation Techniques

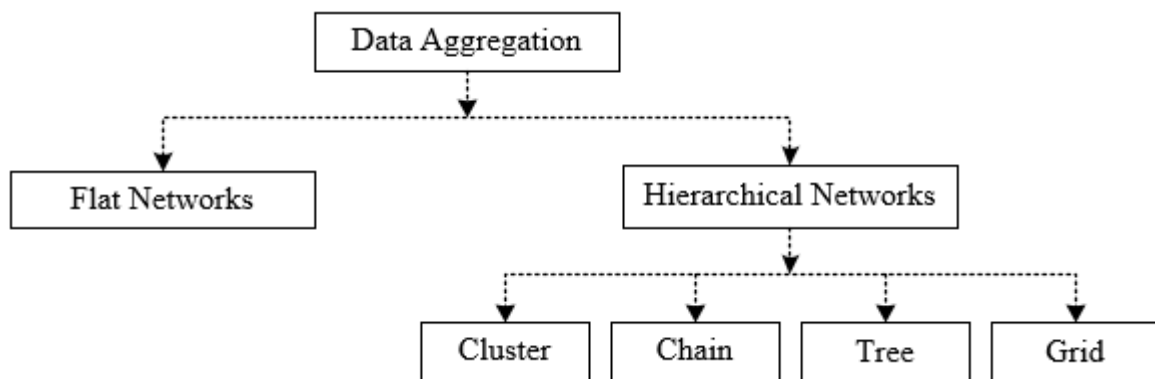


Figure 4 Taxonomy of Data Aggregation based on Network Topology

An efficient data aggregation technique must focus on the energy state of the entire network, it must balance the amount of energy consumed by each sensor node during the aggregation process since the energy conservation is one of the biggest challenges in aggregating data. Normally a WSN is deployed with a specific application, the required quality of service of this application must be guaranteed. There are variety of techniques for data aggregation in wireless sensor networks but the aggregation techniques that we discussed this thesis are generally based on the network topology.

As the wireless sensor networks architectures can be flat or hierarchical, the techniques of data aggregation is also categorized into two categories which are flat and hierarchical network. On

the *Figure 4*, further hierarchical network is sub-classified into four parts which are cluster based, chain based, tree based and grid based[11], [12].

1.6.1.1 Data Aggregation in Flat Networks

In this type of network, all the sensor nodes on the same role and are powered by batteries of the same capacity. Therefore, the data aggregation is done by practicing data-centric routing, the sink floods the network with a request and the sensor nodes with the data corresponding to the request respond to the sink[13]. The power of the sink's battery may run out quickly due to excessive calculations and communications, so the failure of the sink causes the network to malfunction. Hence, to ensure scalability and the energy efficiency of wireless sensor networks the various hierarchical data aggregation techniques have been introduced[11].

1.6.1.2 Data Aggregation in Hierarchical Networks

1.6.1.2.1 Cluster based Data Aggregation

In energy-sensitive and large sensor arrays, sensor nodes face difficulties in transmitting their data directly to the base station. In this situation, the sensor nodes can use the intermediate aggregator nodes to transmit the information to the sink. Thus, in this network, the sensor nodes are organized in the form of clusters i.e. the sensors transmit the data at the head of the cluster which aggregate all the data received from the neighbouring sensor nodes, and then transmit the concise data to the base station (see the following Figure 5). The cluster head can communicate with the base station either via long-range transmissions or via multicasting by other cluster heads[7]. Thus, this process saves energy and is primarily useful for low energy sensor nodes.

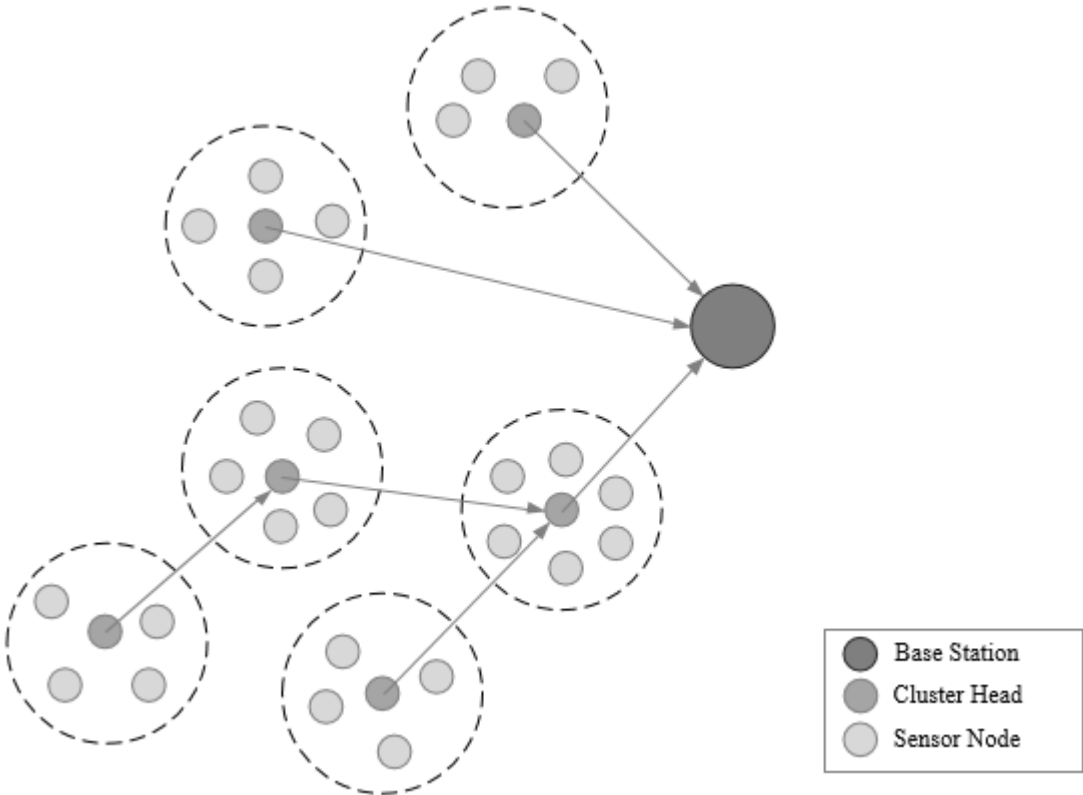


Figure 5 A Clustered Wireless Sensor Network

Sensor nodes in an environment collect data and send it to the base station or in collaboration with other sensor nodes. Some applications areas of wireless sensor networks use sensor

clustering to provide scalability, robustness, and reduced network traffic. Such a clustering is illustrated in the *Figure 5* where the sensor nodes form groups with group heads and these cluster heads transmit aggregated data to the base station[14]. The main benefit of clustering is the scalability of performance on growing sensor networks.

In addition, the clustering approach offers many secondary benefits. It ensures reliability and avoids failures at one point due to its localized solutions. A clustering solution may suggest a sleeping/waking program for a WSN to effectively reduce power consumption. In many sensor applications, not all sensor nodes should be in a standby state and consume power. On the basis of temporal and spatial dependencies, some sensor nodes can be put into sleep mode in which no energy is consumed. Effective planning can be designed and communicated to these sensor nodes via the sink or administrator.

1.6.1.2.2 Chain based Data Aggregation

In the previous section, it was discussed that the cluster head aggregates the information it has received from the same cluster members so that it passes the result to the sink but in the case where distance separating the cluster head and the sink is quite large and it took a lot of energy to be able to communicate with the sink so the chain data aggregation is applied, the aggregated data is sent only to its closest neighbour to the sink.

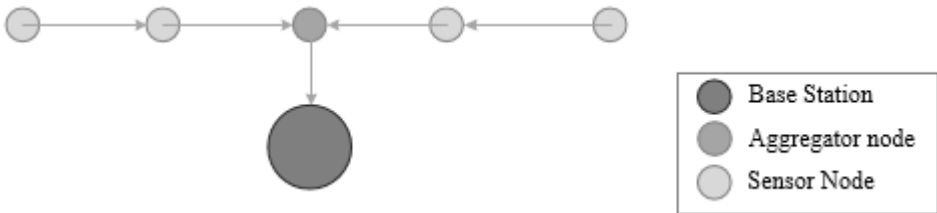


Figure 6 The Chain based organization in a Wireless Sensor Network

In this process, the nodes are constructed as chains for data transmission to the aggregator node. Each node in the network sends the sensed data only to its neighbour node rather than to the aggregator node and each node aggregates the received data to decrease the amount of data transferred[11].

1.6.1.2.3 Tree based Data Aggregation

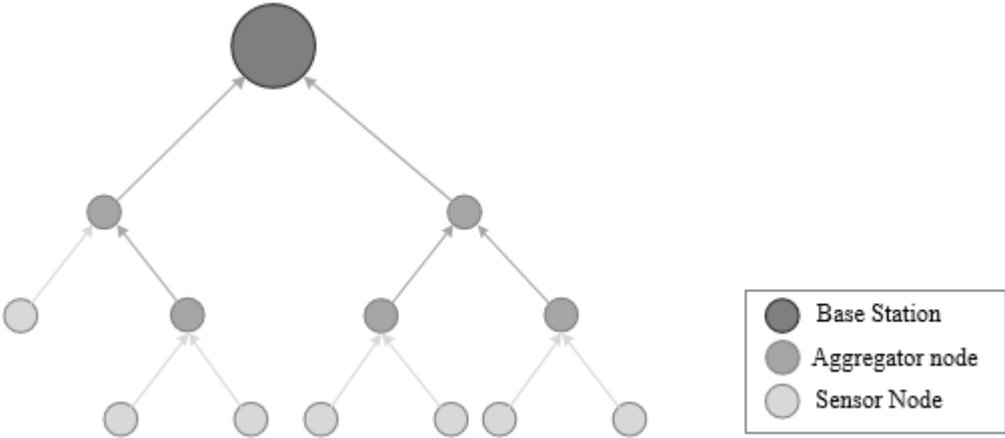


Figure 7 The Tree based Data Aggregation in the WSN

In a tree-based network, the sensor nodes form a topological tree structure where the sink is chosen as the root of the tree. Aggregation of data is performed by mounting an aggregation

tree where the source sensor nodes of the data are the leaves (nodes without any descendants) and rooted at the sink. In this case, the flow of data starts from the leaves sensor nodes to end at the sink, so all the intermediate sensor nodes perform the aggregation process and transfer the aggregation results to the root i.e. the sink or the base station. The main objective for this mode of aggregation is the construction of an energy saving tree.

One of the disadvantages of this approach, as known by the network of wireless sensors are not free of crash (failure). If data packets fail at any tree level, the data will be missing not only for one level, but also for all connected subtrees[15].

1.6.1.2.4 Grid based Data Aggregation

Within a grid, all the sensor nodes are apt to become an aggregator node but in rotation and unless the sensor node considered do not have enough energy to perfect the role of aggregator. So once aggregator nodes are known in each of the grids, they regroup the data from all the other sensors within the same grid and no sensor node does not communicate with another sensor node within the same grid. It has been proposed that this kind of technique is especially useful for applications such as meteorological forecasting or military surveillance[11].

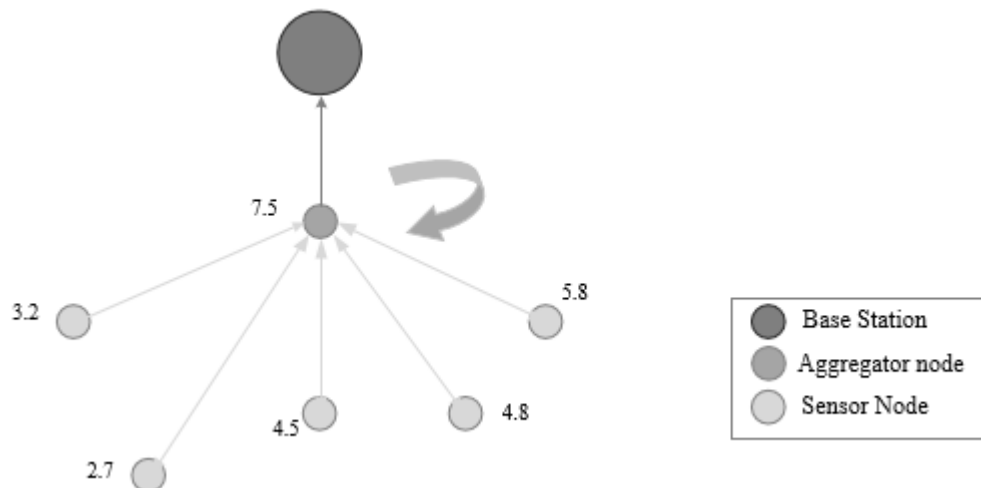


Figure 8 Grid based Data Aggregation in Wireless Sensor Networks[7]

1.6.2. Challenges in Data Aggregation

The challenges of data aggregation in wireless sensor networks consist of a few major aspects and their impacts on a data aggregation scheme are described as follows:

- **Header merge:** when aggregating data in WSN, some header information that is relatively specific to the application concerned must be included in the payload. Examples of header-specific information are the source physical address of the package or the creation date and time of its creation. In order to merge the packets into unused header fields or actual payload, each sensor node considered as relay must know the structure and order of this information even if the same application context applies to the entire network[16].
- **Energy Efficiency:** since sensor nodes are supplied by lower power batteries, effective sleep planning protocols will need to be implemented to save energy from inactive sensor nodes. In the WSN, it seems obvious that most of the sensor nodes are close to one another, which has the consequence that the attributes of the data collected by the neighbouring sensor nodes are approximately similar, in particular in an environmental

monitoring system. Therefore, it is preferable to aggregate the data by collaboration between neighbouring sensors to completely eliminate redundancy in the data so that only useful data is transferred to the base station. This can save the energy used by the sensor nodes for multiple transmissions of redundant data to the base station[1].

- **Security:** for security reasons according to the importance of the WSN application, the content or the payload of the packet can be encrypted and the aggregation of the data must ensure the same level of confidentiality for the aggregated data as the original packets. On the other hand, aggregation of data should not include unbearable computational loads on intermediate aggregator nodes for decryption and re-encryption [16], [17].
- **Downlink traffic:** To aggregate data from the WSN on an uplink, it is sufficient to concatenate the packet payload and the header information to a single aggregated packet and no additional relay logic is required but for the downlink traffic, where the contents of the packet can be destined for several sensor nodes, the problem is more considerable. Large packets should be split into smaller sub-packets based on their destinations and routes. For arbitrary flow or sensor actuator networks, a combination of aggregation and disaggregation would be required, which would increase the solution space and make it difficult to identify a sufficient aggregation scheme[16].

1.7. WSN Attacks and Security requirements

Security in Wireless Sensor Networks differs from other solutions proposed for conventional wired or wireless networks. This is because the sensor nodes are miniature components with limited resources (processing, memory and energy). Wireless sensor networks typically do not have trusted units permanently present in the network. Wireless sensor networks may be similar to wireless Ad-Hoc networks, but they differ in a number of ways, which makes it impossible to directly apply existing security solutions. In order to be able to develop security solutions adapted to these networks, it's better to understand their constraints.

The security requirements in sensor networks depend on the nature of the application. Military applications are very demanding in terms of security, or even non-tolerant at this level; because the network is a double-slice sword, and can become an enemy weapon. But in environmental monitoring applications, for example, safety is not very demanding. The application of a simple mechanism remains sufficient to protect the network from primary attacks. The application of security solutions must be accompanied by a key management mechanism.

1.7.1. Safety Hurdles in Wireless Sensor Networks

- **Low Power Resources:** The use of security algorithms requires memory resources for code and data storage, energy and computing resources[18]. For example for the sensor node Mica2 has a processor clocked at 7.3728 MHz frequency and a memory for the program of 128 KB. On the same sensor node must be installed the operating system and the applications. So the security code and the relative data must be very small. Energy limitation is the most important constraint, since generally the sensor nodes are deployed in the difficult access areas, it is not easy to change their batteries or recharge them. Then the power capacity embedded with the sensor nodes must be conserved to extend their life and subsequently that of the entire network. The additional security functions has an effect on memory usage, processing time and subsequently on energy consumption[19].
- **Unreliable Communication:** This feature is inherited from wireless networks. The data is transmitted in the air, so each sensor node in the coverage radius can listen to the

messages being exchanged. Applying noise to the channel may make the sensors unable to transmit messages because the media may appear as permanently occupied.

- **Unexpected Risks:** Depending on the application of the wireless sensor network, the sensor nodes are unattended or monitored after a long period of time. Sensor nodes are unattended when, for example, they are deployed behind enemy lines. The unattended sensor node warnings are:
 - **Physical attacks risks:** the sensor node is usually deployed in an environment open to enemies, and can cope with difficult weather conditions.
 - **Remote management:** Remote management of the network makes the detection of a physical testing (sensor compromise) and sensor maintenance (spare or battery charging) impossible.
 - **No base station found:** The sensor network must be designed to be a distributed network without a central management point. But if there is a design flaw, the organization of the network can become difficult, inefficient and fragile.

1.7.2. Security Requirements for the WSN

The properties of sensor networks facilitate operation and deployment, but make the communication system quite "tricky" with some failures. In order to deploy this technology on a large scale, it is necessary to mitigate security issues in the WSN. Security services in wireless sensor networks should protect data circulating on the network against any type of attack. Security objectives are classified as primary and secondary[20]. The main objectives are known as standard security objectives: *confidentiality*, *integrity*, *authentication* and *availability*. The secondary objectives are: *freshness*, *non-repudiation*, *access control*, *self-organization*, *synchronization* and *secure localization*. The most important safety requirements in sensor networks are discussed in this section.

- **Data Confidentiality:** This criterion requires that whoever receives the information cannot pass it on to others without the consent of the owner or manager. The confidentiality of the data guarantees that the exchange of data between two sensor nodes (sender and the receiver) will be completely secure and that no third party will be able to access it in reading or writing. For example, in military operations, the sensed data is of great importance so that it can be safely sent to ensure confidentiality and secure key distribution[21].

Ensuring confidentiality is hard work in wireless networks in general. The major problem is that the radio spectrum is an open resource and can be used by anyone equipped with a device that can send and receive the appropriate radio waves. A hacker can eavesdrop on data sent in the air as long as he is able to track the radio channels used in the communication[1]. The most common method of countering eavesdropping is called *data encryption*. It is about transforming the payload of the data into a special form that is understandable only to the authorized sensor nodes.

There are two data encryption techniques, the first is *symmetric key encryption* (see the following Figure 9) because all the sensor nodes of the network share the same encryption key K but keep it secretly between them. When a sending sensor node is about to transmit the plaintext message M to a receiving sensor node, it encrypts it by the algorithm E using the key K to find the ciphertext message $C = E(M, K)$. Then, the sending sensor node transmits the encrypted message C instead of the plaintext message M to the receiving sensor node. After receiving the ciphertext message C , the receiving sensor node uses the decryption algorithm D and the shared key K to retrieve the original plaintext message $M = D(C, K)$. The security in these operations is that the shared secret key K is not known to anyone else except the sensor nodes of the WSN, so no one else

can decrypt the ciphertext message C into plaintext message M and thus the confidentiality is carried out.

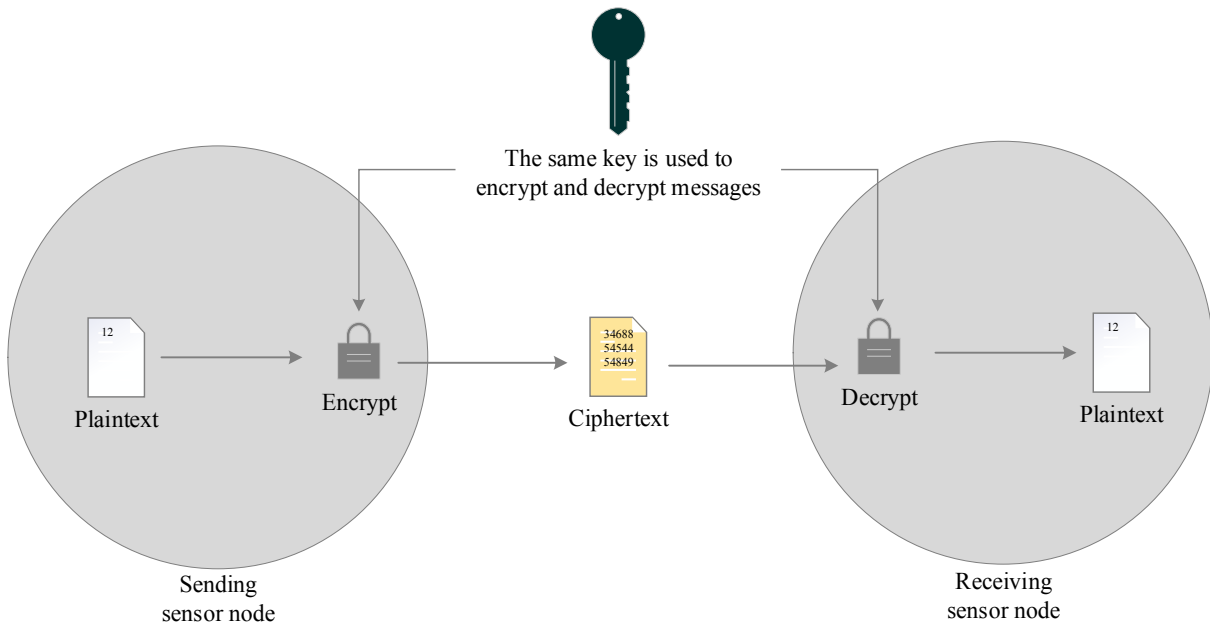


Figure 9 Symmetric Cryptosystem Diagram

On the Figure 10 below, there is another widely used technique which is called *asymmetric key encryption*. In such a cryptosystem, each sensor node has a pair of keys (K_s, K_p) , one of which is private K_s that each node keeps in secret and publishes the other key K_p . When a sending sensor node wants to transmit a plaintext message M to the receiving sensor node, it uses the public key of the receiving sensor node (K_p^r) to encrypt the message M using the algorithm E to obtain a ciphertext message $C = E(M, K_p^r)$ and then sends the ciphertext message C to the receiving sensor node. Since K_p^r is publicly known, anyone can send the message to the receiving sensor node but since K_s^r is secret only the receiving sensor node can decrypt the ciphertext message C using the decryption algorithm D into plaintext message $M = D(C, K_s^r)$.

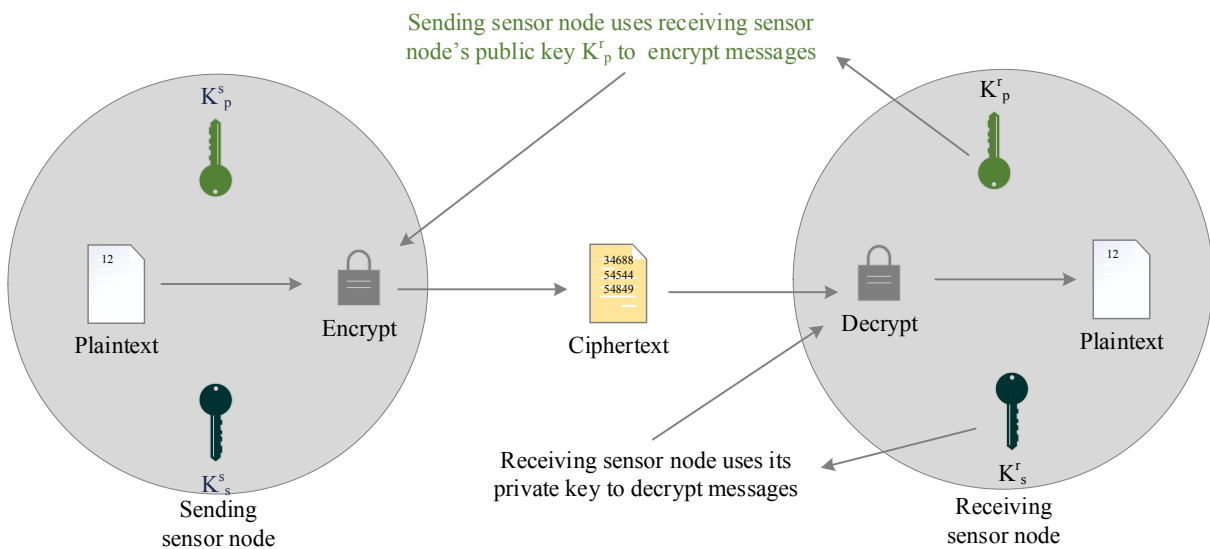


Figure 10 Asymmetric Cryptosystem Diagram

In short, symmetric encryption algorithms require simple computations (such as hashing, bit rotation, or scrambling) that can be efficiently implemented in software or

embedded in hardware. Asymmetric encryption algorithms often require more complex operations (exponential computations, modulo of very large prime numbers, etc.) than symmetric cryptosystems. As a result, symmetric key algorithms are more suitable for devices with limited processing, storage and energy resources such as sensor nodes. On the other hand, asymmetric encryption algorithms are more secure and support additional security services, for example, authentication or non-repudiation, while symmetric cryptosystems are mainly used for encryption.

In the wireless sensor networks, asymmetric key algorithms cannot be used to encrypt broadcast or multicast communications. Obviously because the key used to encrypt the data using an asymmetric key algorithm belongs to the receiving sensor node while in broadcast or multicast communications there are multiple receiving sensor nodes and each of them has its own pair of key for encryption.

- **Data Integrity:** Integrity is the property that ensures that information has not been tampered with or destroyed in an unauthorized manner. In other words, the integrity of the data is the confirmation that the data that has been sent, received or stored is complete and has not been modified. The data integrity of the wireless sensor networks guarantees that the data packets received by a receiving sensor node are exactly the same sent by the sending sensor node and no one in the middle modifies that packet during transmission[22]. Wireless sensor networks operate primarily on broadcast, so it is more vulnerable to such security attacks. Data integrity is a logically basic requirement for communications in the WSN because the receiving sensor node must know exactly what the sending sensor node wants it to know. However, it is not always an easy task when it comes to the wireless communication networks of sensors knowing its constraints.

Transmission errors are common in wireless communications due to unstable wireless channels due to many reasons, such as channel fading, time-frequency consistency, and inter-band interference. A packet containing errors is unnecessary and causes additional processing to the sending and receiving sensor node. It is obvious that no electronic device is perfect, errors can also come from any forwarding sensor node. The malfunction of electronic circuits embedded on sensor nodes can be influenced by ambient conditions such as temperature or humidity, which can inevitably cause errors in the packets. These errors may not be detected by the forwarding sensor node, which means that these erroneous packets may still be sent over the network causing problems on the downstream sensor nodes. Wireless sensor networks are typically deployed in hostile environments where an attacker is able to modify a frame before it reaches the receiving sensor node by adding a few bits to the transmitted frame to change its polarity. If the frame is unintelligible by the receiving sensor node it will be rejected and dropped, causing a denial of service attack[23].

Some error control mechanisms on the data link layer have been implemented to deal with these transmission errors. One of them is the *error detection code*, the idea is about to attach to each frame of the data link layer a few redundancy bits calculated according to an error detection algorithm usually called a *checksum*. By inspection of the checksum, each receiving sensor node determines whether there is an error in a received frame, if an error has occurred during transmission, then the receiving sensor node notifies the sending sensor node in order to retransmit the original frame (this feedback mechanism is called *automatic repeat request*, ARQ). In some cases where several redundancy bits are attached to each frame, the receiving sensor node can correct the errors to avoid the ARQ but the checksum is each frame is computed according to the error correction code algorithm (this mechanism is called *forward error correction*, FEC).

ARQ or FEC does not solve the problem of malicious attacks against data integrity in wireless sensor networks because error detection or error correction code algorithms are publicly known by everyone and therefore a receiving sensor node cannot detect the modification of the frame because the checksum of the modified frame remains the same as the checksum of the original frame. *Message integrity code* MIC and signatures are two cryptographic methods (these two methods are discussed in the next paragraph because they are also used in authentication process) globally used to ensure data integrity in the WSN.

- **Authenticity:** Within a wireless sensor network, it is important to check the authenticity of each sensor node. This is possible thanks to the authentication mechanism, which makes it possible to prove the identity of a sensor node via the identification process. Authenticity guarantees identities of communicating sensor nodes on the network. Each sensor node needs to know the identity of the sender of the received packet. Otherwise, the receiving sensor node may be cheated in performing incorrect actions. In some cases, if an attacker knows the packet format defined in the network protocol stack, he could inject his own packets in addition to modifying already existing packets. The receiving sensor nodes can accept these injected packets containing the false data which results, for example, in disrupting the routing protocols due to the erroneous routing information. The *Sybil Attack* is the perfect example of packet injection attack in the WSN[24].

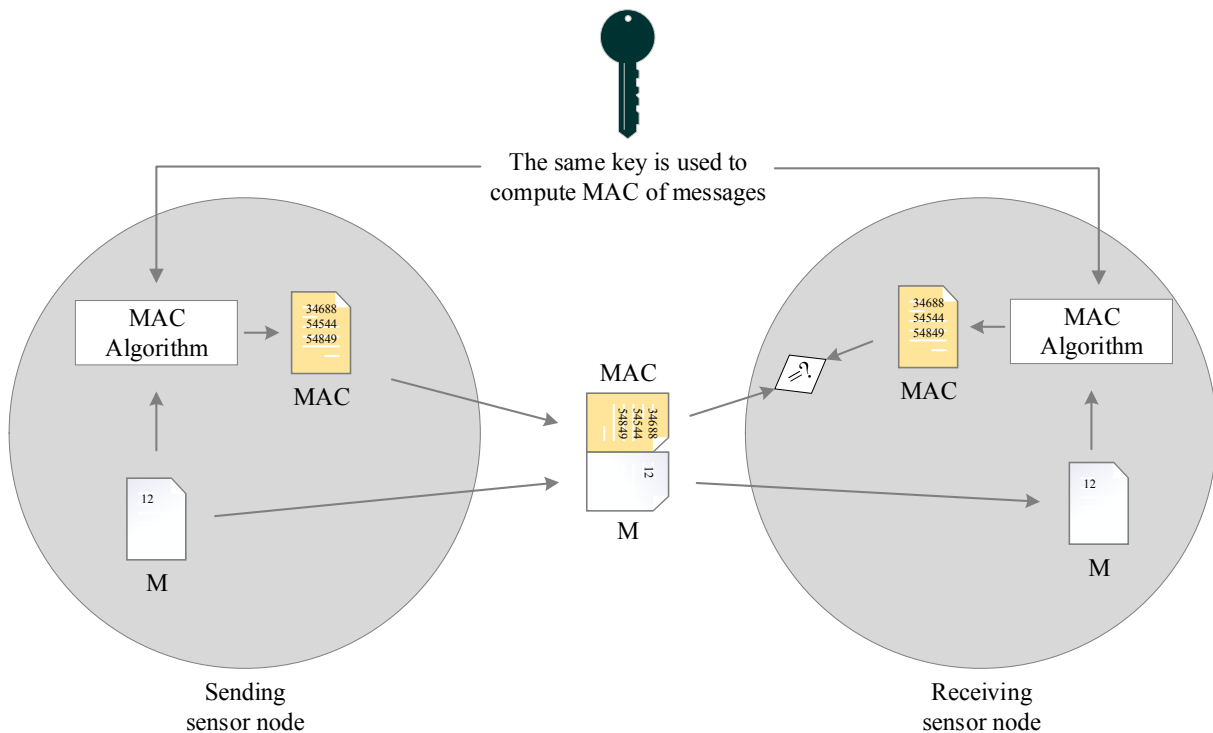


Figure 11 Message Authentication Code diagram

Message Authentication Code (MAC) is a method for handling these false packets to ensure authentication, which is essential to guarantee the origin of the received packets. This method can be also called MIC because it also guarantees the integrity of the data. In the process shown on Figure 11 of calculating a MAC, the requirement consists in acquiring a symmetrical K key shared between the receiving sensor node and the sending sensor node. Let $H(\cdot)$ be a collision-resistant hash function[25] and \parallel a concatenation operator, the sending sensor node concatenates the message (payload) with the key K and computes the MAC as $C = H(M \parallel K)$. Then the packet including

payload M and MAC C is sent to the receiving sensor node. This last node recalculates a Mac C' by using the payload M with the key K to check if C is identical to C' , then if there is equality the packet containing the payload M is authenticated and unmodified because only the sending sensor node knows the shared key. Otherwise, the package is not modified or injected.

In the authentication process, the widely used technique is called a *digital signature*. As shown on the Figure 12, when a sensor node wants to send the data in the form of a message M , it secretly keeps its private key K_s while another key K_p is publicly known by the other sensor nodes in the network. In order for the sending sensor node to authenticate the message M with the receiving sensor node, it uses its private key K_s to sign the message M in a signature $S = S(M, K_s)$ and then sends the signature S and the initial message M to the receiving sensor node. Only the sending sensor node is able of generating the digital signature because the key K_s is secret and as K_p is publicly known any receiving sensor node can check the validity of the digital signature S by obviously using the signature S , the public key K_p and the received message M in a verification algorithm to calculate $V(S, M, K_p)$. The message M is authenticated if the value at the output of the algorithm V is true, it is not otherwise.

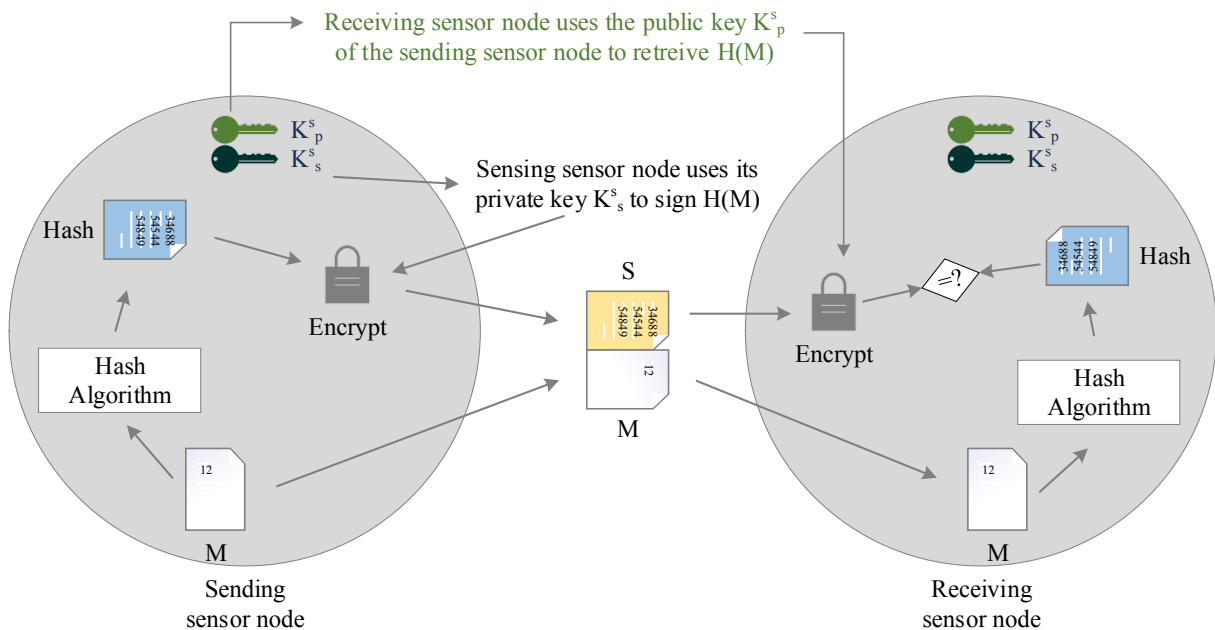


Figure 12 Digital Signature diagram

The signature-based authentication procedure requires that the private key K_s of the sending sensor node is secret and that the public key K_p of the same node is known to all other intended receiving nodes, but it is difficult for a sensor node to know whether a public key belongs to a key holder which results in serious attacks against this authentication system. The example of type of attack against authentication by public keys is called the *man in the middle*[26].

- **Availability:** This service designates the ability of the network to provide its services to maintain its proper functioning by guaranteeing the communicating parties the presence and use of the information at the desired moment. As the sensor nodes can act as servers, availability is difficult to ensure. In fact, a sensor node may not serve information so as not to exhaust its energy, memory and calculation resources, thereby causing bad behavior. The purpose of availability is to ensure that data from the wireless sensor network is accessible in a defined time. Degraded network functionality can be

caused by any problem to compromise the ability to offer the expected services as designed in advance, leading to the DoS[23].

By saving or using their energy properly, the sensor nodes can survive longer to ensure the availability of network resources. In a normal situation or when there is no activity in the network, the sensor nodes can save their energy by switching to the sleeping mode and use it only in case of absolute emergency. When an attack occurs, the base station is responsible for activating all the sleeping sensor nodes but when the situation is normal, only a few nodes continue to operate in active mode[21].

- **Nonrepudiation:** Non-repudiation ensures that the sender of the message will not be able to deny having sent the message in the future. Upon successful transmission of data in the WSNs, a sending sensor node should not deny the fact of having sent such data or messages and at the same time the receiving sensor node should not refuse to have received this data. So non-repudiation is divided into non-repudiation of origin and non-repudiation of receipt.

Non-repudiation is the best-known method to defend against cheating in WSN. When two sensor nodes (transmitter and the receiver of the message) present a dispute on the network they can address an adjudicator who judges which of these two sensor nodes is responsible for dispute, therefore each sensor node must present to the adjudicator his own evidence of the actions taken at the time of the communication.

In order to prove the identity of the source sensor node and the destination, non-repudiation evidences are obtained based on digital signatures, but it seems difficult to build a non-repudiation protocol using these evidences. The sending sensor node (sender of the message) generates its pair of keys (public, private). It broadcasts his public key and keeps secretly it private key. To sign the data as message, the sending sensor node starts by calculating the hash code of the message and then signs this hash code with its private key. The result of this last operation (encryption with private key) is the digital signature that will accompany the message. When the receiving sensor node receives the message and the digital signature, it recalculates the hashing code, decrypts the digital signature with the public key of the sending sensor node and compares the two hashes codes. If the two codes are similar then the signature is valid (see *Figure 12*). The receiving sensor node will not be able to deny in the future to have emitted the message since it is only him who can generate the digital signature with his secret private key.

When a destination sensor node receives a message, it uses its digital signature as proof of the non-repudiation of the origin and finally to complete the non-repudiation protocol, the receiver sensor node should respond to the sending sensor node by a signed acknowledgment of receipt as proof of non-repudiation. In some cases, the receiving sensor node may refuse the reception of message by not responding to the signed acknowledgment.

- **Freshness:** In the wireless sensor networks, the freshness of the data justifies that the date and time of a received packet is recent or not. Therefore, all data from the WSN is described by its temporary status and is valid in a limited time interval. In other words, when a node receives a packet, it must ensure its freshness otherwise the package is useless because it may be that the data it contains are not valid. In WSN, freshness is very important to ensure the network security and to avoid self-destructing packets because an attacker could inject an expired packet in order to waste the network's resources (bandwidth, processing time, etc.) to end up causing self-destruction[21]. In a wireless sensor network, the freshness of a packet can be ensured by attaching the timestamp to each packet transiting the network. Indeed, once a sensor node receives a packet, it can perform a timestamp comparison with its own clock to determine the

validity of this packet. To combat against Wormhole attacks[27], each packet is securely attached with the location information and the time of its emitting sensor node and as long as it progress on the network, the intermediate receiving sensor nodes compare the location and the time of its creating sensor node with their own location and time. Sometimes the wormhole attack occurs if the packet arrives earlier than the expected time.

- **Access Control:** This service consists of preventing external elements from accessing the network, by giving legitimate participants access rights to discern messages from internal network sources and external ones. Anything outside the system must be prevented from accessing the network. Access control helps detect messages sent from foreign sources.

No system is 100% safe, hackers can still find a trick to penetrate and attack the wireless sensor network from inside the network despite the efforts with which the network security infrastructure was designed. These hackers can simply spy on network traffic for not being detected but if they are actively violent to disrupt network communications, some anomalies appear indicating the existence of malicious attacks. These malicious intruders based on these anomalies can be detected by intrusion detection mechanisms[28].

- **Self-Organization:** As in an Ad-hoc network, a sensor network requires that each sensor be independent and flexible enough to self-organize and self-guide in various situations. The lack of a fixed infrastructure to facilitate the management of the wireless sensor network brings a great challenge related to securing the entire network. Depending on the evolution of the situation in the WSN as well as in case of sensor node failures or mobility, sensor nodes and sink nodes can be organized flexibly. When in some cases the sensor nodes are inactive or at very low energy, the neighbouring sensor nodes can arrange to adapt to the new situation.
- **Time Synchronization:** Most WSN applications have some form of synchronization. When a packet is traveling over the network, the communicating sensor nodes (two to two) may wish to calculate the end-to-end packet delay to estimate the time required for synchronization.
- **Secure Localization:** One of the utilities of a wireless sensor network relies on its ability to accurately and automatically locate each sensor node of the network. In order to precisely locate the location of a fault, a sensor network must be designed with the location information. By replaying signals or reporting false signal strengths, an attacker can easily fake information about unsecured location.

1.7.3. Frequent Attacks in the WSN

Because of the broadcast data transmission, sensor networks like other wireless networks are susceptible to multiple security attacks. In addition, wireless sensor networks have a specific feature that makes them more vulnerable than other conventional wireless technologies. For example, sensor nodes are typically located in hostile or dangerous environments that make them vulnerable to compromise attacks. The rest of this section is devoted to the main types of attacks.

- **Active attack:** An active attack attempts to alter the WSN resources or affect their operation, so it compromises the integrity or availability. When the hacker is in place, it causes the destruction of the resources of the network, the alteration of the data, and modification of the direction of the traffic or even the cut of the communication with the sink. It is easy to identify the attacker and restart the network recovery process.

- **Passive attack:** A passive attack attempts to learn or make use of data from the networks, but does not affect network resources, so it compromises confidentiality. The attacker who has a powerful equipment (computing speed, storage space, energy resource, etc.), collects the exchanged information in the wireless sensor network if they are not encrypted. In this type of attack, the hackers observe the activities taking place on the network while also checking the confidential information but without launching physical attacks or compromising the data. By observing the activities on the network, the attackers set a right time to take into action when they find the weakness and clues of the wireless sensor network. These passive attacks are more dangerous than active attacks and it is less likely to recognize the attacker because the attacker does not alter the exchanged data anymore. In this case the enemy eavesdrops on the confidential data or the discovery of the important nodes in the network.
- **Black hole attack:** The attacker can add malicious nodes into the network in order to inject falsified data into the network. A malicious or injected node modifies the routing information and forces the passage of data by itself. Then it creates a black hole in the network and places itself in a strategic location of routing then erases all the messages that it should return them, which allows the interruption of the service of routing of the network in the paths which cross the intruder node. Usually these malicious nodes will be more robust to attract the messages of the other nodes, because in the networks of sensors without the choice of the best path by the algorithms of routing is based on several parameters; among which are the energy and the computing power of the nodes.
- **Denial of service attack:** This attack can be defined as any reaction that reduces or destroys network capacity. In this attack, the hacked or the malicious node has the objective to keep busy the route to the base station by sending additional packets into the network without any specific need. Therefore, legitimate users of the network cannot take advantage of the resources to obtain any service.
- **Wormhole attack:** The attacker captures a message and, using a low latency channel, retransmits it to a remote location in the network. The channel thus created sends a message to a place in the WSN that it should not normally arrive at, or otherwise with greater latency. This redirection of data to a remote location causes the absence of data in another part of the network. This attack has a significant influence on the routing in the network.
- **Sybil attack:** In this type of attack, the malicious node takes a large number of identities that can be stolen or imagined. This attack can be used against routing protocols. To prevent an attacker from triggering a Sybil attack on the wireless sensor network, it is preferable to use authentication and encryption techniques as well as public key cryptography, but the latter is too exhausting in energy to be applied in wireless sensor networks rather than assigning for each node a pair of symmetric keys with the base station.
- **Node Replication:** Once the attacker can reach physically a sensor node from the WSN and extracts the legitimate information and transfers them to its own generic sensor node that uses the ID of a legitimate sensor node. At this point, the attacker can deploy this sensor node in the network in order to inject false data and delete legitimate data. This kind of attack is considerably dangerous because the replicated sensor node can pretend to be any legitimate sensor node or even the sink of the network. Once this replicated node is on the network, it is in a position to occupy a strategic node position so that it can easily retrieve the security keys[29].
- **Flood attack:** The "HELLO" packet is used by several routing protocols to discover the neighbourhood and establish the network topology. The simplest attack for an opponent is to send a stream of any messages to clutter the network and prevent other messages

from being exchanged. In this attack, "HELLO" packets are flooded throughout the network to destroy network resources.

1.8. Applications of Wireless Sensor Networks

Wireless sensor networks may comprise of numerous different types of sensors like low sampling rate, seismic, magnetic, thermal, visual, infrared, radar, and acoustic, which are clever to monitor a wide range of ambient situations. Sensor nodes are used for constant sensing, event ID, event detection & local control of actuators. As shown on the *Figure 13*, the applications of wireless sensor network mainly include health, military, environmental, home, and other commercial areas.

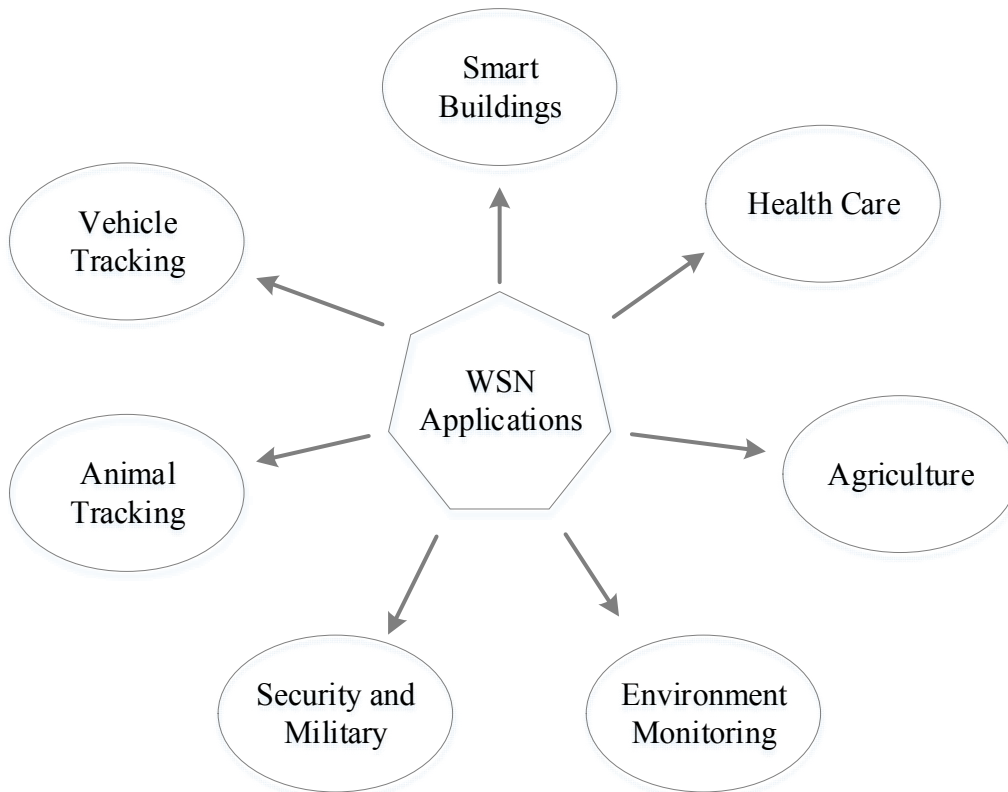


Figure 13 Applications of the Wireless Sensor Networks

1.8.1. Precision Agriculture

Precision agriculture aims to create more creative cultural activities while reducing the impact on the environment. The information gathered from sensors makes it possible to evaluate the most favourable sowing density, to estimate the need for fertilizers and other inputs and to more accurately predict crop yields. WSN plays an inevitable role in the field of agriculture. The architecture is composed of a number of wireless sensor nodes used for collecting and monitoring data such as temperature, humidity, carbon dioxide levels, soil moisture, etc. The detected and collected data is transmitted to cloud computing technology or via the Internet. Researchers at the central station can analyse and take additional steps to improve crop yields at lower cost.

There is a lot of research going on including sensors in agriculture. In [30], real-time monitoring of data using ZigBee to monitor climate and other environmental properties is done using wireless sensors. The main objective of the document is to report on the design, construction and testing of all environmental properties required for precision farming using wireless sensor

networks. Different topologies such as delay, rate, and load are calculated and simulated. The main benefits are improved quality of precision farming, reduced costs and ease of system deployment and maintenance.

A new technique called Integrated WSN Solution for Precision Agriculture is proposed in [31]. The system integrates crop data acquisition, data transfer to the end station and video surveillance. The system provides agricultural data monitoring, long-distance transmission and camera sensors to assess overall performance. The characterization of the product and the quality of the specialty culture using sensors are discussed in [32], [33]. A ground-based real-time monitoring system for the detection of plant diseases in field conditions is presented in [34]. It uses spectral reflectance measurements and fluorescence induction approaches to detect plant diseases. Detection of the disease by fluorescence imaging is performed by considering the patches present in the plants. The other parameters taken into account are the selection of the frequency band and the quadratic discriminant analysis.

Crop management proposed in [35] is a solution for farming accurately, for example by avoiding excessive pesticides or fertilizer. The sensors collect data (humidity, temperature, soil pH, moisture, atmospheric pressure) and send to the central coordinator. If for example the pH level of the soil is above the permitted threshold, the base station sends an SMS to the farmer through a modem connected to the base station.

Wireless sensor networks in agricultural water management are discussed in [36]. The data provided to the sensor is soil moisture, daily precipitation, sun hours, humidity and wind speed. The threshold value is predefined and once the value exceeds the limit, the system triggers an alarm for safety. [37] provides three different complementary approaches that exploit the psychometric properties of air in the drying chamber and temperature oscillations. Two experiments to find the dynamic characterization of the air inside the dryer and access to energy in solar radiation. A precision irrigation system is provided by integrating a central pivot irrigation system with a wireless underground sensor network, explained in [38]. The result of the proposed system shows that the wireless channel between the ground and the air is affected by spatio-temporal aspects such as location, depth of landfill, soil texture, soil moisture and canopy height.

A real-time monitoring system for monitoring the agricultural environment is detailed in [39]. The system is based on wireless sensor networks and paper details of the hardware used. The system has low power consumption, stable operation and high accuracy, which is an advantage. A system is proposed to analyse the use of programmable on-chip system technology as part of WSN to monitor various greenhouse parameters is discussed in [40]. The system helps monitor and control greenhouse parameters in precision farming. For long-term calibration and validation applications, the use of the wireless sensor network for automated monitoring of soil moisture is described in [41]. The network includes a number of automated measurement stations and facilitates the monitoring of soil moisture in real time. For example, the use of wireless sensor networks in the field of agriculture facilitates the collection of weather, crop and soil information as well as monitoring.

1.8.2. Environmental Monitoring

The use of wireless sensor networks in the environment extends its applications to coal mining, earthquakes, tsunamis, flood detection, forest fire prediction, gas leakage, cyclones, precipitation beach, water quality, volcanic eruptions, etc. As the network provides early detection and prediction of all these environmental calamities, it helps to take a security measure at a certain level. The data is collected using the sensors and transmitted to the processing station via the Internet. It helps to take precautions and to make people aware of the disaster that is

about to happen. The application of the wireless sensor network in the environmental monitoring application is discussed here.

- **Air Pollution:** Chemical reactions involving atmospheric pollutants create toxic gas of ozone that affects human health and can also damage the life of plants and animals. Thus, for the early detection of these atmospheric pollutants, wireless sensor networks are used. [42] focuses on elements of the wireless network of sensors that can be selected to monitor air pollution. This helps to measure harmful air pollutants and base particles of weather conditions. A network of wireless sensors for monitoring air pollution in Mauritius is presented in [43]. The system uses the air quality index. By comparing the data obtained with the value index, the system detects contaminated air.
- **Forest Fire:** In most countries, such as Australia, the incidence of fire is common due to the dry and warm climate. This will harm the wildlife. In order to avoid forest fires, in some cases wireless sensor networks are used. [44], [45] describe the use of wireless sensor networks during early detection of forest fires. [44] proposes a comparative study between the two forest fire detection methods such as the Canadian approach and the Korean approach. The method used to detect the fire using the sensor test bed. Considers that features such as energy efficiency, timely detection and accurate localization, predictability and adaptation to harsh environmental conditions. Two different algorithms based on information fusion techniques are proposed in [45], a framework for detecting forest fires using wireless sensor networks is proposed. The framework helps detect forest fire earlier. The power consumption of the sensor nodes is much smaller. The system can perform in any kind of environmental conditions.
- **Gas Leakage:** Gas leakage is another dangerous problem that can cause damage to human life as well as animal vitality. The application of wireless sensor networks to detect gas leakage is discussed in [46], [47]. A remote electronic carbon dioxide monitoring system is being developed in [46]. The system consists of a central processing unit, environmental sensors, GPS, receiver module, digital memory card storage, LCD and GPRS. The data will be saved and will be displayed and will also be able to provide a notification if the status is beyond control. A very wide band sensor network is used to explore oil and gas [47]. A seismic acquisition system is introduced to explore oil and gas in the oceans.
- **Coal Mines:** Another important application of the wireless sensor network is in coal mining. People working in coal mines face many dangerous dangers such as cavitation, gas explosion, collision or breakage of vehicles or equipment, chemical leakage, electric shocks and fires. Therefore, it is inevitable that these fields are constantly monitored for the safety of people working in the mines. [48], [49] describe in detail the application of wireless sensor networks in coal mines. [48] uses a wireless underground network of chain-type mine sensor. The sensors collect information about the environment and location. As parameters such as temperature and chemical rates are monitored continuously using the sensor, it prevents the possibility of danger to some extent. The [49] shows the reaction time for detecting a fire hazard in a panel of Board-and- Pillar coal. It uses wireless sensor networks (WSNs) and can be used to detect exact fire and dispersion location and also offers fire prevention to stop the spread of fire to save natural resources and to extract human resources from fire
- **Water Quality:** one of the most important implementation of WSN is the monitoring of water quality. Various water quality parameters such as pH, ammonia, dissolved oxygen, water level, and so on, can be monitored using WSN and discussed in [50], [51]. Because of people who drink water contaminated by microbes, authors in [52] worked on the water quality by collecting the pH of the water because the methods used

before detect only the particles present in water but not dissolved chemicals products. Their architecture allows collecting the data and sending them to the base station, which sends them to the web server. The sensed data are available in three modes: public users, registered users and the administrator. Everyone may have access to public information via the web browser. This web server sends alerts to all registered users in case of detection of abnormal data. These registered ones can locate the WSN on the map. The administrator has all privileges and is able to retrieve the archived data at a time of the past. The location of WSN is a key to those users to know which area where their water source is located and what the water quality there. This approach does not give an assessment on the performance of the proposed architecture and gives no clarification about displaying data to the user, so it is not possible to know if the user interface is updated once the new information came to the web server from the WSN.

1.8.3. Vehicle Tracking

Smart Transport is another WSN application. Networked cameras and other sensors used to monitor traffic flow to reduce congestion, monitor traffic in the city for traffic violations and detect illegal activities around critical infrastructure such as airports, train stations, etc. [53]–[55] are related to the implementation of WSN in intelligent transport. In [53], it provides the potential benefits of using the WSN-based traffic monitoring system to improve the quality and safety of vehicle transport. In [54], the document describes a smart transport solution that uses wireless sensor networks. In [55], he describes the design and execution of an ecological wireless sensor network that characterizes air quality in Asuncion, Paraguay. Mobile sensors on public transport vehicles provide a useful means of creating a well-organized result for this classification.

1.8.4. Health Care Monitoring

Currently, the use of WSN in the medical field is inevitable. The system consists of sensors for determining various physiological parameters. The measured parameters are then transferred to the practitioner for further analysis and diagnosis. Several papers have been done using WSN. Here are several WSN applications in healthcare in detail from [56]–[62]. In [56], a new home monitoring system has been developed, based on a cognitive sensor network for elderly care applications. The system consists of the optimal number of cognitive wireless sensors to determine the use of electrical devices, bed usage patterns and water consumption. In addition to detection, it includes a panic button for patients in an emergency. Multisensor system for monitoring the physical activity of the patient is described in detail in [57]. The system helps control physical fitness and performance, helps reduce risk factors such as obesity, blood pressure and diabetics, and also improves cardiovascular health and, therefore, helps increase longevity. [58] presents a new design for developing a wireless sensor network structure for monitoring several patients with a chronic disease. [59] represents a remotely controlled physiological monitoring system. The system is able to work without the intervention of the patient. A review of the health sector using a wireless sensor network is described in detail in [60], which discusses various methods that coordinate a wireless sensor network for healthcare applications. The WAP-based telemedicine system was developed in [61], which uses WAP devices as a mobile access point. [62] describes the requirements for ECG sensor design and system design, which is designed to monitor patients with chronic disease in real time. The patient will carry an ECG sensor and the sensor will continuously transfer data to the hospital or to the doctor without any errors, which will lead to continuous monitoring of the patient.

The architecture proposed in [63] is based on four agents: Aggregator Agent, Agent Patient, Doctor Agent and Nurse Agent. Each patient has his own cluster head joining him to the gateway. The patient Agent is located in cluster header and is used to transfer the data to Aggregator Agent situated in the gateway. When Aggregator Agent receives the data from the Patient Agent, it transmits them to the cloud for processing and storage. Then, a doctor and a nurse receive alerts via Doctor and Nurse Agent respectively installed on mobile devices. But this architecture has no real application because the authors conducted simulation. It lacks clarification on the results obtained and simulators used in the experiment.

1.8.5. Smart Buildings

An intelligent building must be able to control and control its functions in accordance with the structure of the building, the internal and external environment. The functionality and their characteristics are directly related to the application. [64]–[67] detail the use of WSN in the design of intelligent buildings. Authors in [64] designed and developed an intelligent system for monitoring and controlling household appliances in real time. The system controls the current and voltage consumed by household appliances. The advantages of this system are low cost and flexibility in operation. [65] provides a detailed WSN case study on supporting energy management using web services technologies and middleware. The approach involves integrating WSN with wireless technologies to support energy management in smart buildings. The wireless sensor network allowed smart home environments to create the ubiquitous and ubiquitous applications presented in [66]. The system provides scalable service and contextual awareness for the end user. The system also develops the application and reports its implementation in a genuine WSN to secure the remote home. [67] develops an automatic wireless sensor network for civil structures. This system is capable of measuring temperature and humidity in a concrete structure.

In Railway Bridge monitoring [68], the authors placed the nodes with accelerometers to measure vibration signals during a running train on the bridge. By averaging five measurements they got the minimum threshold indicating that the train arrived at the bridge. When the algorithm detects the approaching train, the base station stimulates all sleeping nodes so they can collect data by aggregation. After the passage of a train, the values sensed by the accelerometer decrease gradually to another threshold indicating that there is no train passing over the bridge, at the time the base station puts all nodes in sleep mode. There is a missing part in this approach because these authors were based only on data collection. This approach does not give details about how data are displayed and also if the data are sent over the Internet or stored somewhere.

1.8.6. Military Applications

WSN plays an important role in command, control, communication, computer, reconnaissance, surveillance, reconnaissance, and armed guiding systems (C4ISR). The fast operation, self-organization and distinctive characteristics of the sensor networks make them very encouraging for military C4ISR. Sensor networks rely on the intensive use of unused and inexpensive sensor nodes, the destruction of some nodes by aggressive actions does not affect military operations as much as the destruction of a conventional sensor, which makes the concept of sensor networks an advanced approach for the battlefields. [69], [70] details the use of WSN in military surveillance applications. Article [69] discusses the new multi-level architecture of sensor networks, which uses advanced wireless sensor network technology for military operations. The architecture leads to flexible monitoring systems focused on improving operational flexibility and usability. A wireless sensor network for tracking and tracking

opponents of the vehicle is described in detail in [70]. The system constantly monitors and detects the movement, location and position of the vehicle.

1.8.7. Animal Tracking

Animal tracking is another application of wireless sensor networks in which the sensor is attached to the body of the animal to determine the transport and position of the animal. The Zebra network is an example of using wireless sensor networks in animal tracking. It is used to track zebras on the ground. Sensors are attached to the animal's body so that they can monitor the position, location and type of food they consume. Zebra tracking is one of the main uses of WSN in livestock. The sensors on the collar are attached to the zebra's neck. This sensor will help track the zebra permanently. Many studies are conducted in the field of location and transport of animals using sensor networks. [71]–[75] proposes various methods for connecting and monitoring animals using wireless sensor networks. In [71], the article provides an arrangement of cattle in pasture fields using WSN. The main features of this system are low cost due to the potentially increased number of nodes needed to monitor a whole herd of cattle, well organized energy management to ensure the organization a reasonable time and independence to additional equipment that can increase costs, reduce mobility and reliability of mobile devices. An overview of the animal tracking system using WSN is described in detail in [72]. It provides detailed information about sensor nodes that can measure and collect environmental information, as well as limited choices, and can transfer measured data to the end user. Article [73] proposes and studies a system based on WSN for the general monitoring of target animals, for example the monitoring of animals in the neighbouring areas of the flora and fauna passages, created to create safe habits for the animals. animals when crossing transport infrastructure. In addition, it allows you to recognize targets through the use of video sensors associated with intentionally deployed nodes. In [74], the article proposes a simple recovery method that can locate lost targets when tracking a wildlife problem. The use of wireless sensor networks for real-time habitat monitoring is described in detail in [75]. In addition to habitat monitoring, the system is also able to predict system performance and network disturbances. In addition to the applications mentioned, wireless sensor networks are used in many other applications, such as safety and monitoring, chemical monitoring, monitoring of biological parameters, and so on.

1.9. WSN towards Internet of Things

Today, sensors are everywhere. We take that for granted, but there are sensors in our vehicles, in our smart phones, in plants that control CO₂ emissions and even on the ground to monitor soil conditions in vineyards. Although it appears that sensors have been around for some time now, wireless sensor network (WSN) research began in the 1980s, and it is only since 2001 that WSN has generated increased interest from of industry and research. This is due to the availability of inexpensive, low-power miniature components such as processors, radios, and sensors that were often integrated on a single chip (SoC).

The idea of internet of things (IoT) was developed in parallel to WSNs. The term internet of things was devised by Kevin Ashton in 1999 [76] and refers to uniquely identifiable objects and their virtual representations in an “internet-like” structure. These objects can be anything (check some of them on the *Figure 14*) from large buildings, industrial plants, planes, cars, machines, any kind of goods, specific parts of a larger system to human beings, animals and plants and even specific body parts of them.

While IoT does not assume a specific communication technology, wireless communication technologies will play a major role, and in particular, WSNs will proliferate many applications

and many industries. The small, rugged, inexpensive and low powered WSN sensors will bring the IoT to even the smallest objects installed in any kind of environment, at reasonable costs. Integration of these objects into IoT will be a major evolution of WSNs.

A WSN can generally be described as a network of nodes that cooperatively sense and may control the environment, enabling interaction between persons or computers and the surrounding environment [77]. In fact, the activity of sensing, processing, and communication with a limited amount of energy, ignites a cross-layer design approach typically requiring the joint consideration of distributed signal/data processing, medium access control, and communication protocols [78].

In an IoT system, all of the sensors directly send their information to the internet. For example, a sensor may be used to monitor the temperature of a body of water. In this case, the data will be immediately or periodically sent directly to the internet, where a server can process the data and it can be interpreted on a front-end interface.



Figure 14 Internet of Things (IoT) Concept

Conversely, in a WSN, there is no direct connection to the internet. Instead, the various sensors connect to some kind of router or central node. A person may then route the data from the router or central node as they see fit. That being said, an IoT system can utilize a wireless sensor network by communicating with its router to gather data. We can think of a wireless sensor network as more of a group of sensors or "a big sensor" and less like a "competitor" or "rival" to the Internet-of-Things.

IoT exists at a higher level than WSN. In other words, WSN is often a technology used within an IoT system. A large collection of sensors, as in a mesh network, can be used to individually gather data and send data through a router to the internet in an IoT system.

It's also important to note that the term "wireless sensor network" is not nearly as encompassing as "the internet of things." WSN consists of a network of only wireless sensors. If the network

was to include a wired sensor, it could no longer be labelled a "wireless sensor network." This is unlike IoT. Essentially any device that connects to the internet can be considered an IoT device. An "IoT system" can therefore be interpreted as a group of many IoT devices.

Chapter -2-

Cloud Computing Concept

2.1. Introduction

The cloud is everywhere, but its definition is a little understandable for many people. In its simplest version, cloud computing consists of using computing or remote storage resources, via the network (generally the internet), the cloud being a metaphor for the network of networks. This definition being broad, the cloud covers in fact many very disparate applications. Nobody really knows when, but one day, someone decided to draw a cloud to symbolize a network with an indefinite topology, probably on a diagram illustrating an interconnection of networks.

Since then, the new symbol has flourished. Everyone has taken up the idea, and especially to represent the largest network, the network of networks: the Internet. Indeed, how to find the right visual metaphor for something that is inherently vague, poorly defined, and therefore unrepresentable? In the minds of people, the concept of the cloud, a vaporous and somewhat shapeless volume, with vague contours and variable geometry, perfectly corresponded to the symbolization of a confused mass like the Internet. Quickly, the cloud has become the visual metaphor for the internet par excellence.

In the past, we used to access our data and programs on the hard disk of our computer, or, in the case of the company, often on a server located on the local network. Today, access to your data and programs can be done via the Internet. The immediate benefit is that you are no longer dependent on a computer or your office presence; your data is accessible from anywhere, anytime, with a wide variety of devices. They reside on a remote server, sometimes spread over several servers, or duplicated on different servers for security. It is said that they are "in the cloud", which is a part of truth for most users: you rarely know where your data are exactly, somewhere on a server rack in a huge "Server farm". When it comes to the cloud, users are often left in the fog. Also, many services for the general public use the cloud without always mentioning it explicitly. Many of us practice cloud computing in this way.

Cloud computing is an evolving model and its definitions, use cases, technologies, benefits and risks will be progressively refined. The cloud computing industry has a very large ecosystem of models, providers and specialized markets. The following definition, already mentioned above, tries to cover all the different approaches. Cloud Computing is an IT model that allows easy and on-demand network access to a shared set of configurable computing resources (servers, storage, applications, and services) that can be quickly provisioned and released with minimal management effort or interaction with the service provider

For companies, one of the advantages of the model is to be able to resize its IT resources without accounting capital. Do you need to deploy new servers, increased storage capacity or compute? You do not need to make any material investments: you rent what you need when you need it, in a flexible and fast way, requiring less administration. It is thus a delegation of infrastructure which allows to adjust your resources in very short times, according to your request which can be as fluctuating as unimportant.

2.2. The origins of Cloud Computing

To discover the roots of cloud computing, it is enough to observe the evolution of several technologies particularly in hardware (virtualization, multi-core chips), internet technologies (Web services, service-oriented architectures, Web 2.0), distributed computing (clusters, grids) and systems management (automatic computing, data center automation)[79].

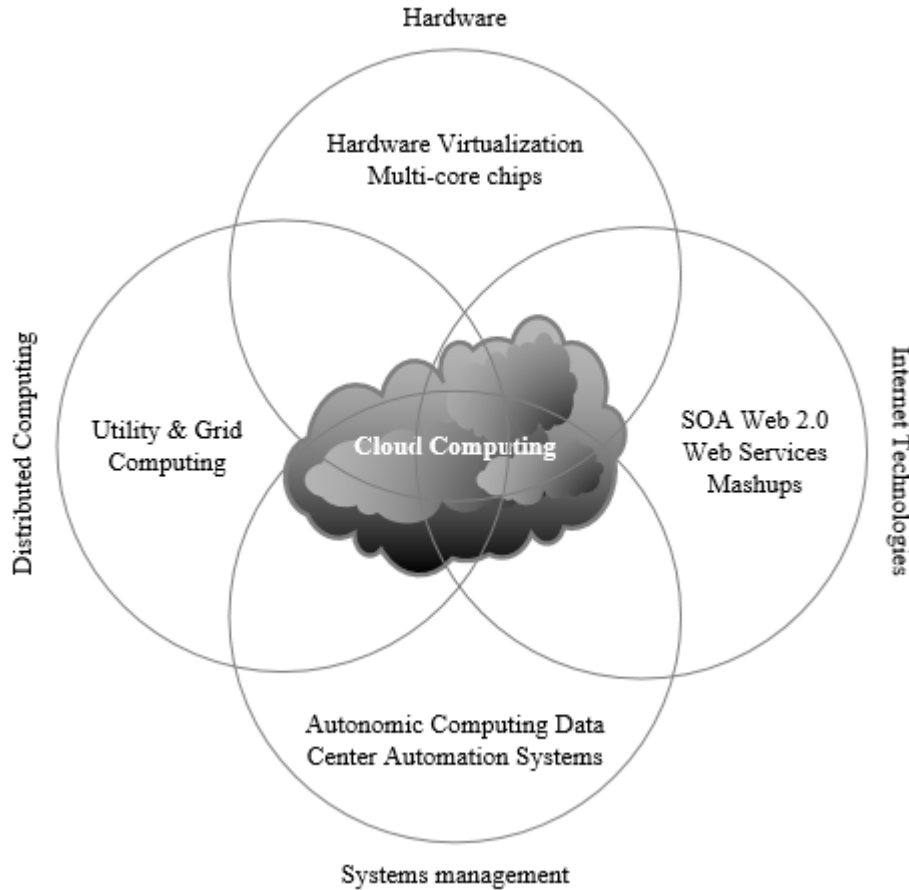


Figure 15 Convergence of various advances leading to the advent of cloud computing.

Consumers and IT service providers benefit from the advantages of this cloud computing model. Consumers are getting the IT services they need at low cost from external providers instead of investing heavily in infrastructure themselves and hiring staff to run the infrastructure. On-demand service enables consumers to meet growing or unpredictable computing needs. IT service providers benefit from the best costs because hardware and software infrastructures are developed to meet the multiple needs and many users by increasing the efficiency and benefits leading to a faster return on investment[80].

2.3. Reasons for migration to the Cloud Computing

Faced with the complexity of infrastructures and technologies, it is not always easy for SMEs to manage their information system on a daily basis. Medium-sized companies rarely have the time to invest a lot of energy in managing infrastructure, applications and users. However, they have the same requirements as all other companies: They are also dependent on their information system and demand as much agility, flexibility and security to accelerate their growth.

2.3.1. Optimize the IT budget

This is often one of the first reasons why medium-sized companies think about going into the cloud. Before the advent of the Cloud, to offer a quality application and necessary to its business, the company had to invest in the purchase of hardware, software but also in the installation and maintenance of the solution. Not to mention the need to strengthen the IT team to manage the entire chain with specific skills.

With the Cloud, the company subscribes to a subscription that usually includes the hardware, hosting, application and part of its installation. It also offers a pay-per-use system, adapting the environment to the growth of the company. It therefore reduces the expense of updates, support and equipment, since these settings are supported by the cloud provider.

2.3.2. Improve the work of users

Users are becoming more demanding with the computer tool because it has become an indispensable everyday tool. They tend to seek more and more IT service. But for financial reasons, it is not always extensible. One of the keys is to free up time for IT to be available to users.

The Cloud allows you to free yourself from certain tasks such as hardware failures, hardware evolutions, management of the computer room etc ... to focus on business applications and users. The business of IT evolves, it becomes closer to business than hardware. IT managers are increasingly taking the role of Project Manager to manage all contracts and ensure compliance.

2.3.3. Access data anywhere and anytime

The company can securely access their data anywhere and anytime of the day. Unless the company wants to put special restrictions for security reasons. SaaS applications (explained in future sections) are usually accessible from any web browser. As for the other applications, there are different solutions to make them available such as, for example, the virtual office or the application publication.

2.3.4. Secure company's data

If security remains one of the main obstacles to migration to the cloud, this is not always justified. Data backup is critical for the business, yet it's not always a priority. Poorly done, it can waste a lot of money on the company. Cloud services provide automatic backup of systems and supervise the elements to ensure the company's assets. On the other hand, data and applications benefit from an often more secure environment than that of the company on several points:

- Security of the computer room: The data centers have security systems at the electrical, climatic, etc. They are designed to face many risks.
- Security of the data and applications: Cloud providers, like Amazon, implement redundancy systems to ensure a minimum level of availability of 99.9% up to 99.999%.
- A security team: Security is a complex subject that requires specific expertise that is not always available to the company.

2.4. Desired features of Cloud Computing

The Cloud Computing model is differentiated by the following five essential characteristics:

- **On-demand self-services:** The notion of self-service on demand is paramount for cloud service users. On-demand self-service allows the user to be able to provision, but also to release remote resources in real time as needed, and without the need for human intervention. The implementation of the systems is fully automated and it is the user, through a control console, who sets up and manages the remote configuration.
- **Broadband network access:** These processing centers are usually connected directly to the Internet backbone for excellent connectivity. Large vendors are distributing processing centers around the globe to provide system access in less than 50 ms from any location.
- **Resource pooling:** It is often possible to choose a geographical area to put the data "near" users. The hardware resources of the provider are shared between the different users of the service. This sharing makes it impossible to determine the exact location of user data, which can be a real problem for companies that are subject to many regulatory constraints regarding the location and control of data. In return, the economies of scale achieved make it possible to reduce the costs of the supplier and therefore the expenses of the users.
- **Rapid elasticity:** The upload of a new instance of a server is done in a few minutes, shutdown and restart in a few seconds. All these operations can be done automatically by scripts. These management mechanisms make it possible to fully benefit from usage billing by adapting computing power to instantaneous traffic.
- **Measured services and billing for use:** The use of resources can be controlled and measured by the provider, essential conditions for billing, access control, optimization and planning of the deployment of IT resources and guarantee of transparency for the service provider and the user of the service. There is usually no cost of commissioning (it is the user who performs the operations). Billing is calculated based on the duration and quantity of resources used. A stopped processing unit is not charged.
- **Virtualization:** Virtualization is an indispensable feature with many benefits. The hardware is replaced by software with all the advantages of the software: to create a new machine or to save its state is to copy a file from which a huge saving of time and money. The virtual machine hardly ever fails, which seriously increases the reliability of the systems. We can continue to use machines that are no longer manufactured. The actual usage percentage of a physical server rarely exceeds 15%. On the same computing power it is possible to run several servers. When a configuration is used for development, recipe operations or load tests, it is possible to free up resources by archiving the configuration. Putting the system back online when needed is done in minutes.
- **Cheap software:** Most public platforms use free open source software. The costs of proprietary software are often billed for use without requiring the purchase of licenses. Most of these programs are already preinstalled and preconfigured which saves a lot of time as explained in the example of use of the last chapter.
- **Geographical distribution:** Large public platforms have centers around the world to reduce risks and place data closer to users.
- **Advanced security features:** Security is a major concern for organizations that use cloud services. These platforms usually have many protection systems out of reach of most organizations.

2.5. Service models of the Cloud Computing

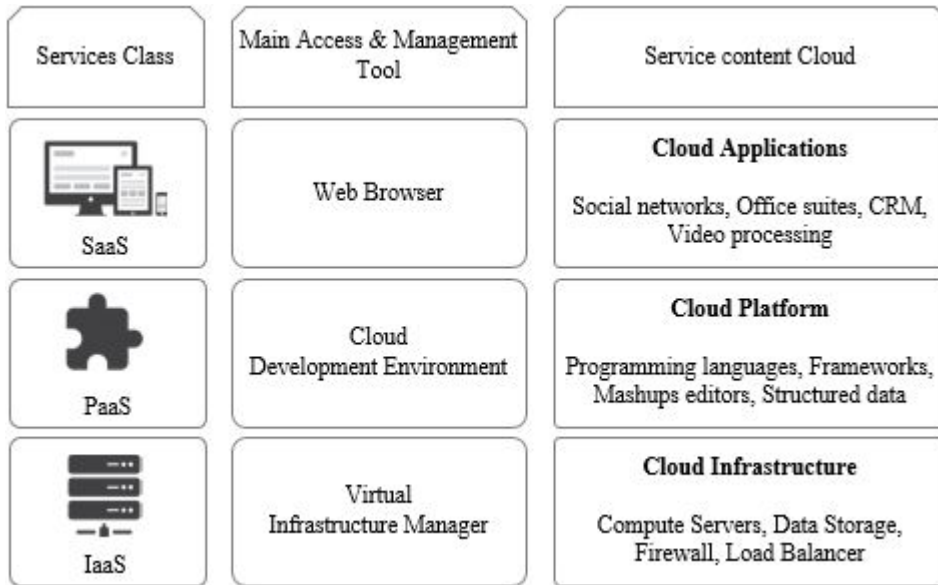


Figure 16 The cloud computing stack

Cloud computing gives developers and IT departments the ability to focus on the essentials and avoid undifferentiated tasks such as provisioning, maintenance, and capacity planning. As cloud computing has grown in popularity, several different deployment models and strategies have emerged to meet the specific needs of different users. Each type of cloud service and deployment method offers different levels of control, flexibility, and management. Grasping the differences between cloud applications, cloud platform and cloud infrastructure to determine which deployment strategies to use can help to choose the set of services you need.

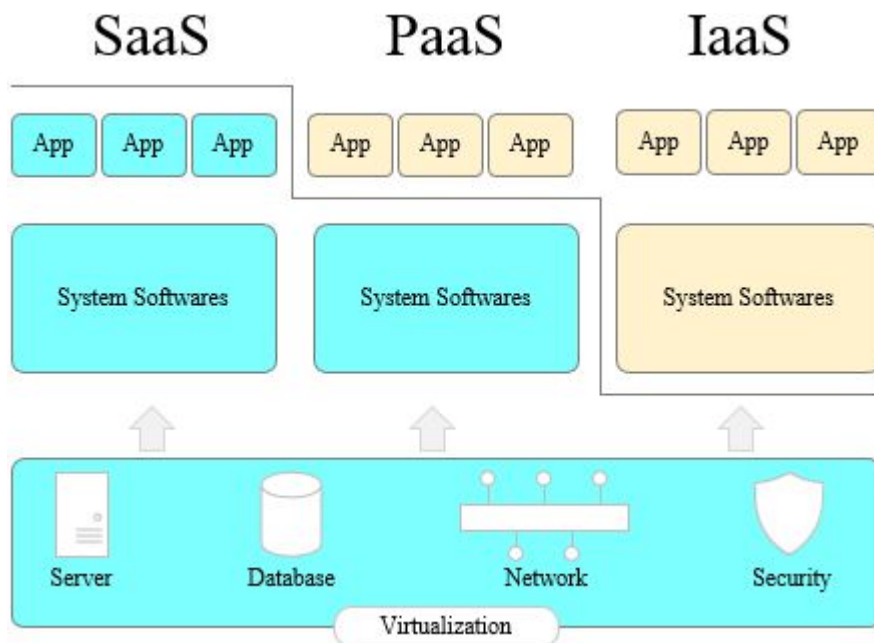


Figure 17 Types of clouds based on services

According to the NIST (National Institute of Standards and Technology)[81], three service models can be offered on the cloud (*Figure 16* and *Figure 17*): Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). These three service models must be deployed on infrastructures that have the five essential characteristics mentioned above

to be considered as Cloud Computing. Often referred to as the main components of the cloud, they help determine different degrees of outsourcing.

2.5.1. Software as a Service (SaaS)

This service model is characterized by the use of a shared application that runs on a cloud infrastructure. The user accesses the application through the network through various types of terminals (often via a web browser). The application administrator does not manage and control the underlying infrastructure (networks, servers, applications, storage). It does not control the functions of the application except for setting a few limited user functions. The provision of a SaaS solution may be billed by subscription or proportionally to the use and may sometimes include customization and service provision. In the field of web marketing, email campaign management platforms, web analytics tools and advertising servers are generally offered in SaaS mode.

Good examples of SaaS are the messaging software through a browser like *Gmail* or *Yahoo mail*. These infrastructures provide the messaging service to hundreds of millions of users and tens of millions of businesses.

- **Advantage:** no more installation, no more updates (they are continuous with the provider), no more data migration etc. Pay with use. Test new software with ease.
- **Disadvantage:** limitation by definition to the proposed software. No control over the storage and securing of data associated with the software. Reactivity of web applications not always ideal.

2.5.2. Platform as a Service (PaaS)

The user has the ability to create and deploy on a cloud PaaS infrastructure its own applications using the vendor's languages and tools. The user does not support or does not control the underlying cloud infrastructure (networks, servers, storage) but the user control the deployed application and its configuration.

Example of PaaS include the *Google App Engine* (GAE) which servers to deploy Web services. In both cases, the user of these services does not have to manage servers or systems to deploy their applications online and to size resources adapted to the traffic.

- **Advantage:** the deployment is automated, no additional software to buy or install.
- **Disadvantage:** limitation to one or two technologies (e.g. Python or Java for Google App Engine, .NET for Microsoft Azure, etc.). No control of the underlying virtual machines. Only for web applications. The targets are the developers. Google App Engine is the main player offering this kind of infrastructure.

2.5.3. Infrastructure as a Service (IaaS)

The user rents computing and storage resources, network capabilities and other necessary resources (load sharing, firewall, cache). The user has the opportunity to deploy any type of software including operating systems. The user does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and applications. It can also choose the main features of network devices such as load sharing, firewalls, and so on.

This is the provision, on demand, of infrastructure resources, most of which are located remotely in Datacenters. IaaS allows access to servers and their configurations for corporate

administrators. The customer has the opportunity to rent clusters, memory or data storage. The cost is directly related to the occupancy rate.

The prime example of this type of service is *Amazon Web Services*, which provides computing (EC2), storage (S3, EBS), online database (SimpleDB), and many other basic services. It is now imitated by many suppliers.

- **Advantage:** great flexibility, total system control (remote administration by SSH or Remote Desktop Protocol, RDP), which allows you to install everything type of business software.
- **Disadvantage:** need system administrators as for traditional server solutions. The targets are the IT infrastructure managers. Amazon EC2 is the main provider of this kind of infrastructure.

2.6. Deployment Models of the Cloud Computing

Thanks to cloud computing, many resources can be connected via private or public networks. This technology simplifies infrastructure deployment by providing dynamic and scalable support for hosting services that are often difficult to predict over time. Organizations can choose to deploy their cloud infrastructure in-house, working with partners who share a common vision, using the services of a public cloud service provider, or by combining many of these solutions. Four deployment models are listed below, although these models have little influence on the technical characteristics of deployed systems.

2.6.1. Public Cloud

In a public cloud, the environment is entirely owned by the company that provides its cloud services (*Figure 18*). Users and customers of a public cloud have no rights to the infrastructure, hardware, software, or anything else. The cloud provider provides resources to users. It designs, manages, maintains and evolves these resources based on customer needs over time. In the cloud, and the public cloud in particular, security is the keystone.

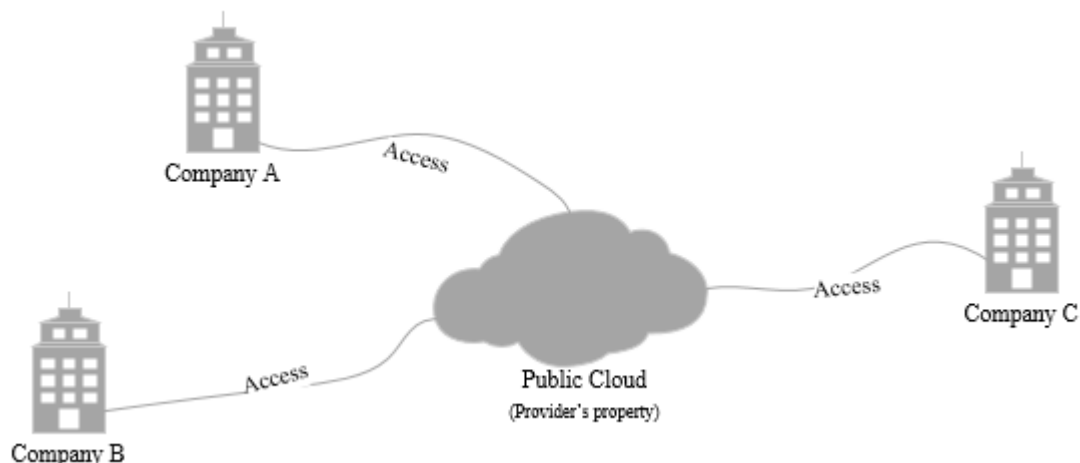


Figure 18 Public cloud deployment model

Since the infrastructure is shared with many customers, security is the responsibility of the provider who manages it. This point is all the more crucial because the customer has only a very weak degree of control and monitoring of the physical and logical aspects of security over the resources that are made available to him. The supplier must therefore make every effort to maintain the trust of its users.

2.6.2. Private Cloud

The term private cloud is used to describe the infrastructure that emulates cloud computing on a private network. The private cloud aims to offer some benefits of cloud computing while limiting its disadvantages. A private cloud is owned by the user company (*Figure 19*), this requires buying, building and maintaining all of its constituents, which means supporting a very large initial investment.

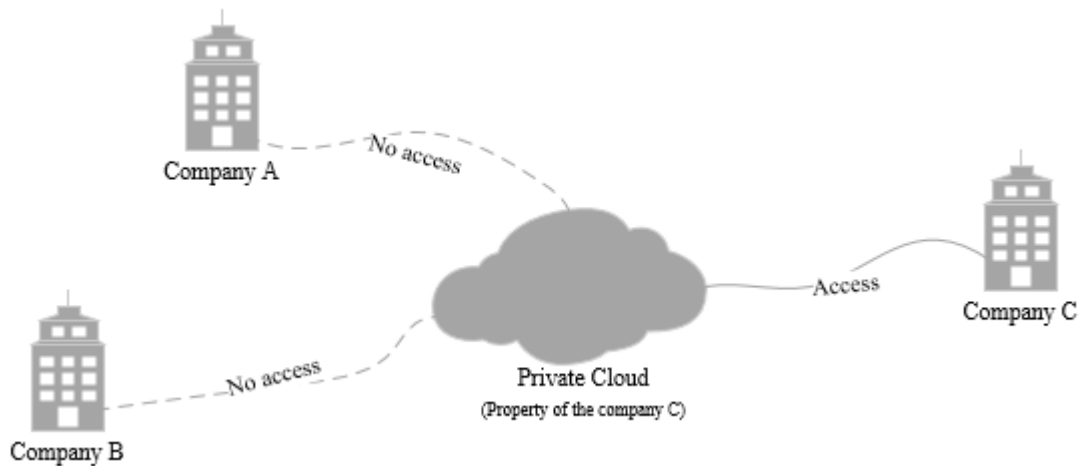


Figure 19 Private cloud deployment model

Private clouds differ from public clouds in that the networks, servers, and storage infrastructures associated with them are dedicated to a single company and are not shared with others. Since the cloud is fully controlled by the enterprise itself, the security risks associated with a private cloud are minimized. This high degree of control and transparency makes it easier for the owner of a private cloud to comply with standards, security policies or regulatory compliance that may be required in certain areas.

2.6.3. Community Cloud

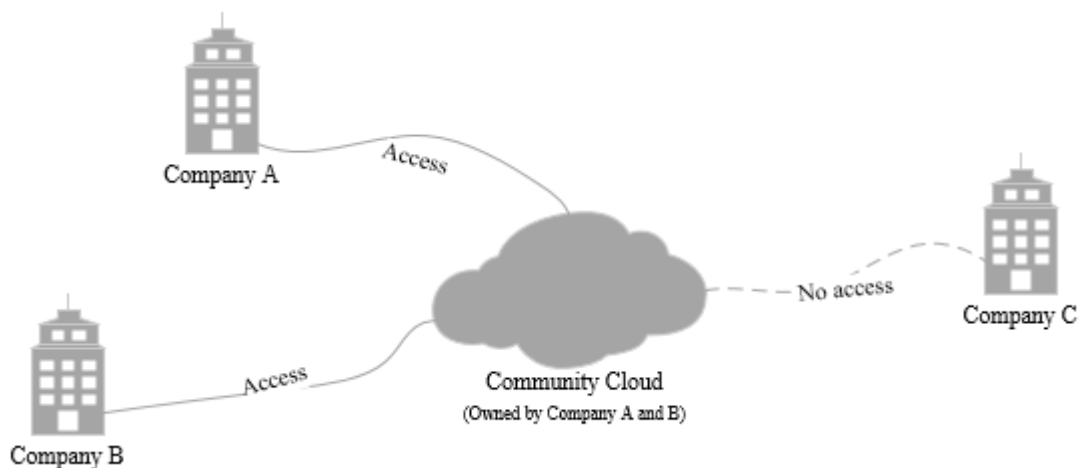


Figure 20 Community Cloud Deployment Model

The community-based cloud is a multi-tenant deployment model that is shared among multiple organizations or companies and is governed, managed, and secured by all participants or a service provider (*Figure 20*). A community cloud is a hybrid form of private cloud built and operated specifically for a small and targeted group. These communities have similar

requirements and bring together their human and financial resources to achieve their common goals.

The common infrastructure is specifically designed to meet the requirements of a community; for example, government agencies, hospitals, or telecommunications companies with similar network, security, storage, computing, or automation constraints might find common interests in collectively deploying a community cloud.

2.6.4. Hybrid Cloud

As the name suggests, a hybrid cloud is the combination of multiple cloud deployment models (Figure 21). With a hybrid cloud, a business can take advantage of the simplicity and low cost of a public cloud to host traditional services that require no special precautions while creating its own private cloud for applications that are tightly integrated with existing systems or storage of sensitive data. It also has the opportunity to privilege the use of its private cloud while keeping the possibility of overflowing on a public cloud offer in case of temporary need.

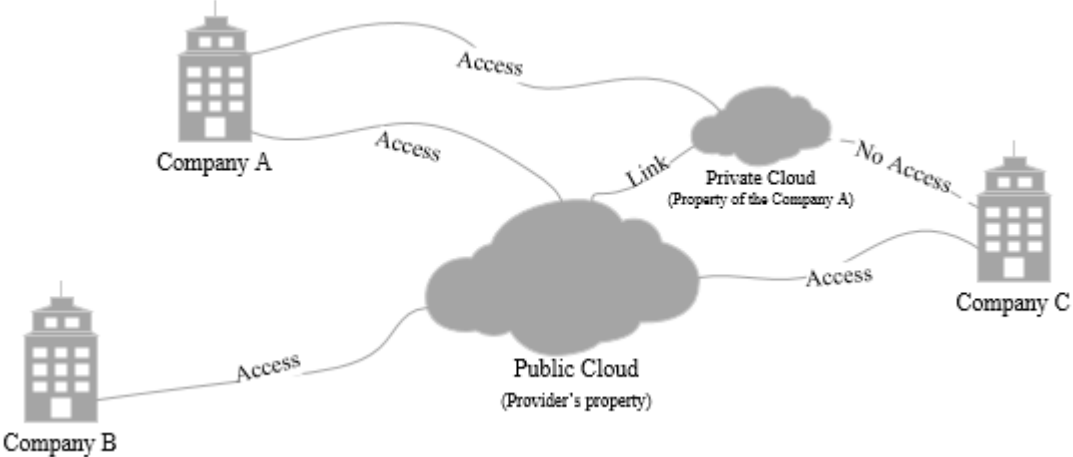


Figure 21 Hybrid cloud deployment model

In a hybrid cloud, public, private or community clouds remain unique entities, but are interconnected by a standard or proprietary technology that allows the portability of data and applications. Due to the complexity of combining multiple types of clouds, designing, managing, and maintaining a hybrid cloud can be challenging.

2.7. Virtualization and the basis of the Cloud Computing

A server is a computer used remotely from different workstations, or other servers. It has hardware resources, mainly CPU, memory, disks and network interfaces. These resources are used by applications, not in a direct way, but by relying on an operating system. Server virtualization is a set of techniques and tools for running multiple operating systems on a single physical server. Virtualization covers all the hardware and software techniques that make it possible to operate on a single machine several operating systems, several different and partitioned instances of the same system or several applications, separately from one another, as if they were running on separate physical machines[82]. The principle of virtualization is therefore a sharing principle: the different operating systems share the resources of the server. To be operationally useful, virtualization must respect two fundamental principles:

- **Partitioning:** Each operating system has independent operation, and cannot interfere with others in any way.

- **Transparency:** operating in virtualized mode does not change the operating system's operation, let alone applications.
- **Compatibility:** all applications can turn on a virtualized system, and their operation is not changed.

Cloud Computing is a technique for managing resources (servers) and adapting an infrastructure very quickly to load variations in a completely transparent manner for the administrator and the users. The applications offered in Cloud Computing mode are no longer necessarily on a computer server hosted at the user but in a cloud formed of the interconnection of geographically separate servers performed at giant server farms (also called datacenters). This is made possible by the virtualization process of operating multiple operating systems and their associated applications on a single physical server. Virtualization makes it possible to recreate several virtual machines on one and the same physical machine.

Virtualization has been the cornerstone of the cloud computing era. Indeed, this notion makes it possible to optimize the material resources by sharing them between several environments in order to be able to execute several "virtual" systems on a single physical resource and to provide an additional layer of abstraction of the material. Cloud computing is the juxtaposition of these technologies to move up a gear on the exploitation of data across the Internet.

Virtualization is changing the relationship between hardware and software, and is one of the cornerstones of cloud computing technology, which makes full use of the capabilities of cloud computing. Unlike virtualization, cloud computing is primarily based on service resulting from this change. Most of the time, the big confusion between virtualization and cloud computing occurs because they are complementary in order to offer users different types of services.

2.7.1. Virtualization Mechanism

- A main operating system called *host operating system* is installed on a single physical server. This system serves as a host for other operating systems.
- A virtualization software called *hypervisor* is installed on the main operating system to allow the creation of closed and independent environments on which will be installed other operating systems called "*guest operating systems*". These created environments are called *virtual machines*.
- A guest operating system is installed in a virtual machine that runs independently other guest systems in other virtual machines. Each virtual machine has access to the resources of the physical server (memory, disk space, etc.).

2.7.2. Understanding Hypervisors

A hypervisor is a virtualization platform that allows multiple operating systems to work separately on the same physical machine at the same time[82]. Hypervisors are becoming more prevalent in enterprise architectures as virtualization grows and spreads rapidly. The solutions are multiple and more or less expensive, but for hypervisors, there are two types. However, a server must have a hardware configuration that supports Intel-VT or AMD-V technology to virtualize.

2.7.2.1 Type 1 Hypervisor Type

A Type 1 hypervisor (*Figure 22*) is a system that installs directly on the server hardware layer. These systems are lightened so as to "focus" on the management of the guest operating systems that are those used by the virtual machines they contain. This frees up as much resources as

possible for virtual machines. However, it is possible to run only one hypervisor at a time on a server.

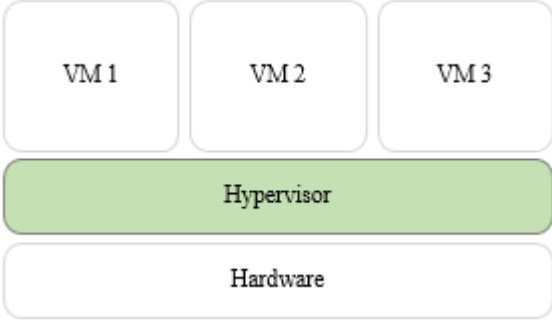


Figure 22 Type 1 hypervisors run directly on bare metal

The Type 1 hypervisor is a lightweight and optimized host kernel to initially run only kernels of guest operating systems that are adapted and optimized for this specific architecture, these guest systems being "aware" of being virtualized. On processors with hardware virtualization instructions (AMD-V and Intel VT), the guest operating system no longer needs to be modified to run in a type 1 hypervisor. Some examples of such hypervisors more recent are Xen, Oracle VM and VMware's ESX Server.

2.7.2.2 Type 2 Hypervisor Type

A Type 2 hypervisor is software that installs and runs on an operating system already in place. As a result, more resources are being used because the hypervisor and the operating system that supports it are running, so there are fewer resources available for virtual machines. As shown on the *Figure 23*, the interest that can be found is the fact of being able to run several hypervisors simultaneously as they are not linked to the hardware layer.

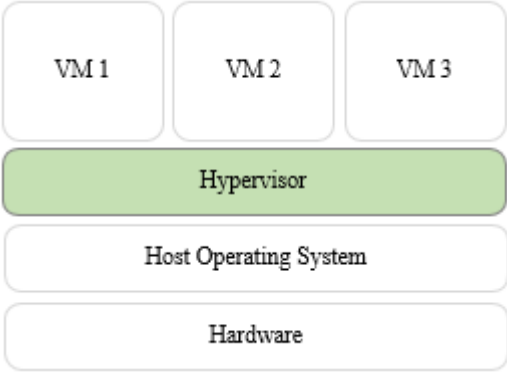


Figure 23 Type 2 hypervisors run within an operating system environment

A guest operating system runs at the third level above the hardware. Guest operating systems are not aware of being virtualized, so they do not need to be adapted. Some examples of such hypervisors are VMware Workstation, VMware Fusion, QEMU open source hypervisor, Microsoft Virtual PC and Virtual Server products, Oracle VirtualBox, as well as SWsoft Parallels Workstation and Parallels Desktop.

2.7.3. Types of Virtualization

2.7.3.1 Full Virtualization

Full virtualization is an operating system that runs hypervisor software (*Figure 24*). The hypervisor will allow the execution of several virtual machines on the physical machine. It manages memory access, CPU allocation and all the resources needed for virtual machines. In full virtualization, the hypervisor manages all virtual machine requests, which allows virtual machines to operate without any change to their kernel. In other words, virtual machines do not know that they are running virtually. The most known product is VMware Infrastructure. At the boot of the machine, a Linux launches to first load the administration console of the machine and secondly the OS dedicated to the hypervisor. VMware also offers VMware Server running on either Windows or Linux. Microsoft offers Virtual Server running Windows 2003 Server and soon Hyper-V integrated with Windows 2008 server.

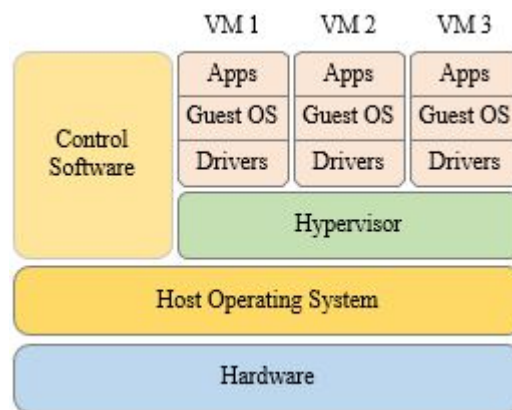


Figure 24 Full virtualization

The so-called full virtualization makes it possible to operate any operating system as a guest in a virtual machine. For the end user, this type of virtualization is the simplest to set up and is the most practical. The principle of complete virtualization allows the hypervisor to create a complete virtual environment literally simulating a new complete computer, with "fake hardware". With few exceptions, the guest operating system (installed in the virtual machine) only communicates with this fake simulated hardware, sealing the virtualized environment.

The limitation of this type of virtualization is based on virtualizing only operating systems intended for the same hardware architecture as the physical processor of the host computer[83]. For example, a computer equipped with an Intel x86 processor will be unable to virtualize an operating system intended to run in a PowerPC architecture. To overcome this limit, it is necessary to use emulation: the hypervisor then creates a complete virtual environment, going as far as to simulate a microprocessor which can then have a hardware architecture different from that of the host CPU. The main disadvantage of this type of solution is then the level of performance, often poor.

2.7.3.2 Para-Virtualization

Para-virtualization and complete virtualization are pretty close. They rely on a hypervisor layer, which fully manages the interface with the hardware resources, and on which one can install different operating systems. The para-virtualization presents to the operating system a special generic machine, which therefore requires special interfaces integrated into the guest systems, in the form of drivers (*Figure 25*). Para-virtualization is a lower-level virtualization technique than isolation. It shares with the latter the need to use a modified OS. More precisely, in para-

virtualization it is no longer only the host OS that must be modified, but also the OSs that are to run on the virtual environments.

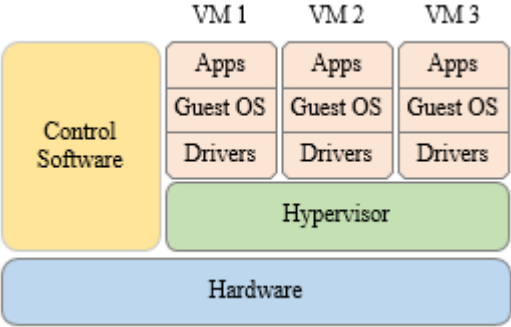


Figure 25 Para-virtualization

The core of para-virtualization is a hypervisor that works closely with the hardware, and provides an interface that allows multiple host systems to access resources concurrently. Each virtual system must be modified to use this interface to access the hardware. Unlike isolation, many OSs from different families can run on the same physical server. This makes it possible to run Linux, NetWare, Solaris (and others) simultaneously on the same machine. Each OS will then have access to its own storage devices, its own memory, its own network interface (s), or its own processors, and each virtualized hardware resource being shared with other environments. The need for small changes to the guest operating system excludes support for closed systems, especially Microsoft Windows.

2.7.3.3 Isolation

An isolator is a software to isolate the execution of applications in what are called contexts, or execution areas. As shown on *Figure 26*, the isolator allows to run several times the same application in a multi-instance mode (multiple instances of execution) even if it was not designed for that. This solution is very efficient, due to the lack of overhead (time spent by a system to do nothing but manage), but virtualized environments are not completely isolated. The performance is at the rendezvous, however we cannot really talk about virtualization operating systems. Only related to Linux systems, insulators are actually composed of several elements and can take many forms.

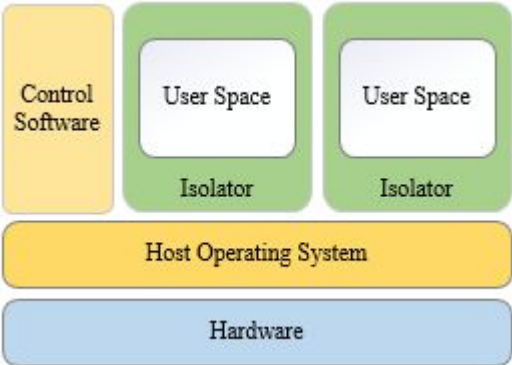


Figure 26 Isolator

In this situation, a kernel runs as an application in the user space, which gives it its own memory space to manage and allows it to control applications. This virtualization method is not very efficient because the guest system is not completely independent of the host system. On the other hand, the two kernels stacked in the same physical system which makes the two systems highly dependent.

The isolation is used under UNIX to protect the systems. Via mechanisms like *chroot* or *jail* it is possible to run applications in an environment that is not belonging to the host system, but a mini system containing only what the application needs, and having only access limited to resources. It is also possible to launch programs in a distribution other than that of the main system. With isolation, kernel space is not differentiated, it is unique, shared between different contexts. But we define multiple partitioned user spaces. This is how you can make different operating system distributions coexist, as long as they share the same kernel.

Context isolation is a lightweight solution, especially in Linux environments. But for the most common needs of virtualization, the simplicity of implementation and the low overhead are excellent arguments. The main solution for isolation is Linux-V Server, the most mature and the most advanced. OpenVZ is an alternative, which presents itself in the same way and offers almost the same features. It is the basis of the Virtuozzo commercial product.

2.7.4. Advantages of Virtualization

- **Minimize hardware costs:** It's easy to see that virtualization reduces the burden of having to buy hardware for any new system. The first benefit of virtualization is obvious: since many virtual machines can run on a single physical server, the number of servers to buy and maintain will be reduced. With a traditional IT infrastructure, many servers are oversized to guard against potential load spikes. Grouping multiple VMs in one place, a virtualized server is better exploited. The total number of machines needed for the proper operation of the IS can be revised downward.
- **Availability:** Today, virtualization solutions enable hot migration of virtual machines. Understand that it is possible to move a VM from one physical server to another, without even having to stop it. This feature is a key element to improve the availability rate of your services. The use of two physical servers makes it easy to double a virtualized infrastructure (redundancy). In case of failure of one of the two servers, the VMs will be automatically moved to the second one.
- **Performance:** Another benefit of hot-migration of virtual machines between physical servers is that it distributes the workload among the servers. When a VM loads extremely, the others will be able to fall back on a less busy physical server. Critical tasks may also work within a VM with more CPU cores (virtual), (virtual) memory, and (virtual) disk space than others. It is thus possible to modulate the size of the VMs according to the tasks they will have to perform.
- **Security:** In traditional small and medium enterprises' IT infrastructure, ERP, file sharing, the messaging system, and even the web server, all run on the same server. However, if the mail is infected by malware, all applications hosted on the machine is endangered. Virtualization can be used to separate the different tasks of a physical server into as many separate virtual machines, which will then be isolated from each other, allowing the services to be partitioned.
- **Anti-obsolescence guarantee:** A part of your IS, for example your ERP, is likely to work on a dedicated server, because of a specific configuration. Ensuring the renewal of this dedicated server, which is only used by one application, is often unprofitable. P2V tools (physical to virtual) make it possible to transform most physical servers into virtual machines. Once this is done, your virtualized ERP server will be able to switch from a machine at the end of its life to a new server, simply by hot migration of the VM.
- **Gain on licensing costs:** You must have one operating system license per server. However, virtualization can sometimes benefit from license packs covering the OS of the physical server and its virtual machines. Get closer to your supplier to check this point, which varies by solution. For existing VMs, it will not be necessary to reimburse

a license when the physical server making them work will be changed. A plus, which also applies to applications running within these VMs.

- **Simplified backups:** With a virtualized infrastructure, the physical server is the only one physically present in the machine room. VMs are pure software for their part. This aspect greatly simplifies data backup operations. It is indeed possible to directly make a backup of the contents of the virtual hard disk of a VM. And even during its operation, creating a snapshot of the virtual machine and its data. In case of problems, this snapshot will restart the VM in a previous state.
- **Manageable DRP:** Virtualization can simplify the Disaster Recovery Plan (DRP) by facilitating the implementation of complex resumption plans. For example, launching a database server before the ERP server that accesses it. Be careful, because virtualization will not solve SI design errors. Example: a server A requiring a server B to be launched, which cannot start without a database of the server A. Separating the separate VM services should, however, limit the occurrence of such errors.
- **Free testing:** Virtualization makes it possible to create a blank VM in minutes. As long as your physical server does not display complete, it will be possible to add new virtual machines to manage. Developers or system administrators can exploit this feature to try new services without spending a penny. Where a test server was previously needed, a simple virtual machine on an existing server of the company is enough today.
- **Step to the private cloud:** Server virtualization allows you to deploy new services in your computer system - virtual machines. But also to size these VM according to the expected criticality and use.

2.8. Cloud Challenges and Risks

Security is a problem for the entire cloud and is therefore implicitly easier to manage. For example, through virtualization (we can safely assume that all IaaS platforms are based on it) can identify resource-related patterns to identify non-normal usage situations. Managing infrastructure updates is easy, so it's easy to have a system up-to-date at the latest patch level. Security in some modes (e.g. SaaS) is fully charged for the Cloud Provider and, therefore, is not a problem for organizations using these services.

There are many risks associated with resource sharing and isolation management (networks, disks, and hypervisors). Cloud infrastructures can be misused to carry out attacks or to manage botnet networks. The legislation is inadequate and inappropriate: it is still debatable in terms of servers "physically" and well located. The security of the APIs exposed by cloud services is essential (overall security depends on the correct design of APIs in terms of security model and ability to exploit them for attacks). Lack of physical control and resource sharing can lead to industrial spying, loss or manipulation of data, exposure of data to third parties.

In general, it is not clear that Cloud Computing is implicitly more or less certain of "traditional" systems. In fact, like everything else, Cloud Computing creates new opportunities and introduces new risks. The novelty is that, in addition to the traditional risks (which remain), there are also other "new" derivatives of this paradigm (declined on the three models of delivery). The problems thus extend from more "traditional" security domains to contractual and legal aspects that were partially absent in traditional architectures and which, in some regions, are not at all secondary.

In summary, Cloud Computing includes *all traditional risks for information security and adds more*. Cloud computing based on the technologies of multi-renting and virtualization introduces new risks and security vulnerabilities specific to cloud computing in addition to the risks incurred by traditional environments[84]. The pooling of resources allows savings on

equipment and therefore indirectly a reduction in electricity consumption thanks to virtualization and multi-lease technologies, but these technologies introduce some risks into the system. Sharing infrastructure between multiple clients leads to the risks of data visibility by other users. In addition, cloud users want to ensure that critical data is not accessible and used illegally, even by cloud providers. The on-demand service is provided to clients through web-based management interfaces that causes the probability of unauthorized access to the management interface higher than traditional systems.

Chapter -3-

Background and Literature Review

3.1. WSN and Internet Integration Approaches

There are many approaches that can be used to make wireless sensor network services accessible to the worldwide community. By behaving like the adjacent "capillary networks" [85], wireless sensor networks are not fully integrated into the internet but its services can be provided via a standard interface, or they can behave as any other IP-based network whose sensor nodes can communicate directly with other hosts located on the internet. In addition, there is a third alternative of leaving independence to wireless sensor networks but allowing the establishment of direct connections. These three approaches (see the *Figure 27*) can be summarized as follows: Front-End Approach, Gateway Approach and TCP/IP Approach.

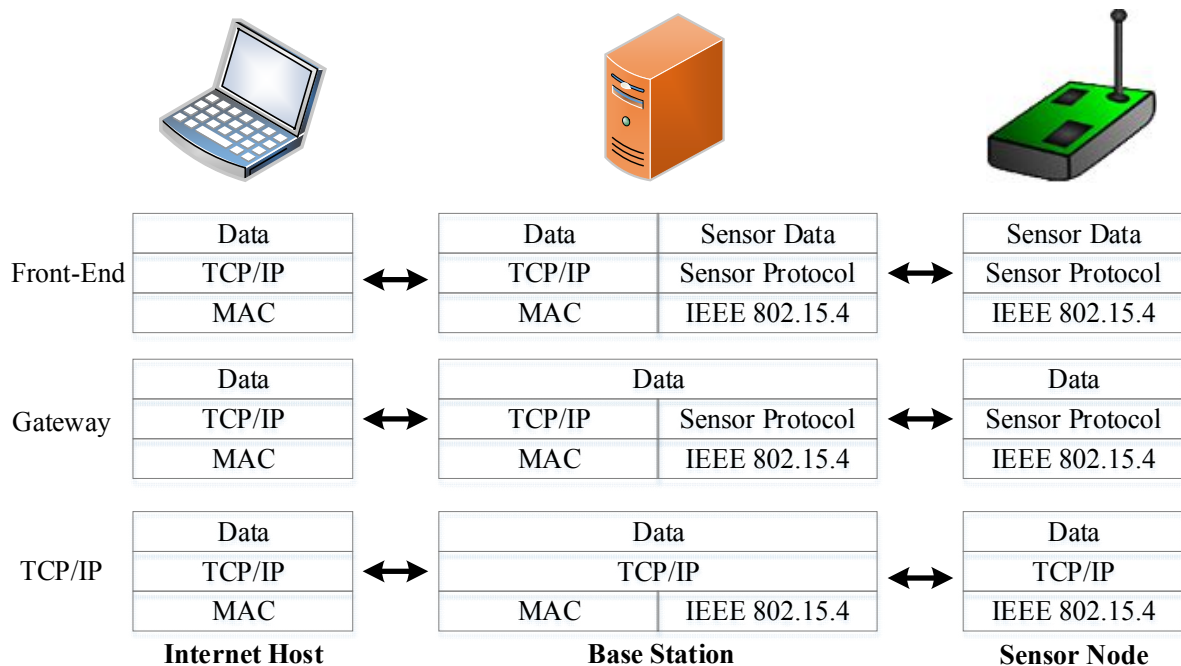


Figure 27 WSN and Internet integration strategies

3.1.1. Front-End Approach

In Front-End solution, the base station mediates between the data acquisition network (the wireless sensor network) and the data dissemination network (the Internet). In this approach, data collected by the wireless sensor network is gathered at the base station where it is stored and the base station also sends any control information to the sensor nodes. For this approach, there is no direct connection between the sensor nodes and the internet because all incoming or outgoing information is passed through the base station for analysis. Since the wireless sensor network is completely independent, its own protocols and algorithms can be implemented. From a functional point of view, the base station can use standard mechanisms such as web

services [86] or web feed [87] to provide the services of its sensor nodes to hosts situated on the internet.

3.1.2. Gateway Approach

In this approach, the base station acts as the application layer gateway that is used to translate the lower layer protocols of two networks (for example, TCP/IP and ZigBee). As a result, internet users can exchange information with sensor nodes directly through this gateway. From its infrastructure, the wireless sensor network can still retain some of its independence, provided that it is necessary to establish a translation table for mapping sensor node addresses to IP addresses of internet's hosts. TinyREST [88] is one of the web services solutions that takes advantage of this approach. However, there is a small variant of this solution, known as the Delay-Tolerant Network Gateway (DTN) where the base station can be able to store and forward packets between networks. In the case of link breakage between the base station and the sensor nodes, the packet is not transmitted, but it is stored for later transmission [89].

3.1.3. TCP/IP Approach

Finally, for TCP/IP solution, the sensor nodes communicate with each other using the TCP/IP model. Therefore, the base station behaves like a router that transfers packets to and from the sensor nodes. In this approach, the sensor nodes must implement the protocols and standards used on the internet, such as the TCP/IP stack or the OSI model as well as the web services interfaces. Since a decade, there are implementations and specifications for IPv4 addresses [90] and IPv6 addresses for sensor nodes, although the future deployments will be based on IPv6 addresses[91]. By declaring its interfaces using the web service description language (WSDL), the sensor nodes are able to provide web services through connections to other hosts using the HTTP protocol[92].

3.2. Infrastructural Issues of the Integration

While the benefits of connecting a network of sensors to the Internet are obvious, it is unclear whether sensor nodes must be fully integrated with the Internet or must remain independent within a cloud of "adjacent" networks. Some of the infrastructure issues to consider when deploying a sensor network connected to the Internet are described below:

- **Addressing:** To send a message to a specific sensor node in the front-end proxy solution, it is necessary to incorporate such functionality at the interface of the base station (for example, as a web service). This is not the case for other solutions, in which a sensor node can be directly addressable using IP addresses. Nevertheless, the way in which IP addresses could be used in sensor networks that identify its nodes using location information or other data-centric protocols is unclear.
- **Protocols:** By providing interoperability at the network and application layers (TCP/IP and web services, respectively), it is possible to integrate a sensor network with the Internet. In addition, this is also beneficial for achieving interoperability between different providers. On the other hand, by using specific protocols tailored to the requirements of the application and the scenario in which the sensor network is deployed, this network may be able to respond adequately and optimally to application-specific problems.
- **Data availability:** When a sensor node is not available, it is impossible to obtain either data streams from its sensors or historical values stored in its flash memory. However, if the data stream is accessed via a proxy or gateway, there may be specific network

mechanisms to provide data regardless of the state of the node (for example, by measuring data from nodes in the same context as the interrupted node).

- **Network-specific issues:** A sensor network has very specific features that differentiate it from other network paradigms. Its nodes are battery powered and, in most scenarios, they must provide their services as long as possible. In addition, in order to save power, the data rate of the wireless channel is not high (i.e. 250 kbps in IEEE 802.15.4). Therefore, the sensor network protocols and the applications that use its services must take into account these specific constraints.

3.3. Security Challenges

3.3.1. WSN-specific Security Challenges

Sensor networks are not intrinsically secure. Its nodes must be deployed near the event source and they use wireless communication channels to exchange messages. As a result, any malicious opponent can manipulate the sensor nodes, the environment, or the communication channel to their advantage. In addition, if this malicious outsider manages to access one or more sensor nodes, it may be possible to manipulate the information that passes through the nodes. Therefore, a sensor network must be prepared from hardware from its nodes to their application layer to prevent or reduce the effects of such attacks[93].

It is essential to provide basic security primitives to sensor nodes in order to minimize the flow of information and to create secure protocols. These security primitives are symmetric key encryption (SKE) schemes, public key cryptography (PKC), and hash functions. Since most sensor nodes are very limited in resources, it is difficult to implement them effectively. These security primitives also require certain security credentials, such as secret keys, to function. The task of creating and providing these keys, thus creating a secure key infrastructure, is performed by the key management system (KMS). Building these KMS is not a trivial task, as they must respect properties such as scalability, communication overhead, connectivity, and so on.

This underlying security infrastructure is essential to protect the network from attack, but not enough to protect the entire infrastructure against attacks from within the network. Therefore, the most critical protocols of a sensor network must be prepared to handle malicious activities and node failures as part of their core functionality. Routing algorithms must support full connectivity and network coverage while being fault-tolerant. Data aggregation protocols must handle false data received from faulty nodes or nodes controlled by an adversary. Time synchronization protocols must reduce errors should reduce errors related to malicious activities or accumulated propagation delays to a minimum. Finally, other protocols such as clustering or locating and positioning nodes must be as secure as possible.

Other aspects of sensor network safety should also be taken into account. For example, a sensor node must be able to detect any abnormal events occurring in its vicinity, detecting suspicious behaviour in other nodes of the network and protecting the network against any malfunction. This self-knowledge could serve as a basis for complex security services, such as intrusion detection systems (IDS) and trusted architectures. In addition, other aspects of security, such as secure management of mobile nodes and base stations, delegation of tasks, data privacy, secure network agents, secure code update, security Code certification, the generation of secure random numbers and many others, should also be taken into account. Note that the importance of all these security solutions must be consistent with the importance of the processed data and the security requirements of the scenario and the application.

3.3.2. Integration security challenges

As mentioned above, some security issues need to be resolved to create the foundation for a secure sensor network. However, once a secure sensor network is integrated with the Internet, new security issues will emerge that will need to be addressed by both parties. As network users will connect remotely to the services provided by the nodes, it is necessary to protect the flow of information. In addition, opening sensor network services to the global community will facilitate the existence of unauthorized users attempting to access the network. Therefore, there must be mechanisms to authenticate both sites and users, and to check if a user is allowed to access the directory. Other factors such as availability and responsibility must also be taken into account.

When a sensor node and an Internet host communicate, it is important to configure a secure channel that supports end-to-end integrity and privacy services. If the integrity of the sensory data is protected, the attacks targeting the data flow will not be able to falsify the readings. In addition, once the confidentiality of the sensory data is assured, no adversary will be able to read the sensitive information contained in the data stream. Creating a secure communication channel requires a common secret key between the two peers, which should be negotiated using standard Internet security protocols such as SSL/TLS. Note that in most cases, the use of pre-shared keys (as in TLS-PSK, for example) is not an option: sensor nodes must be able to services to any user with a certain degree of security.

It is also important to support device and user authentication. An Internet host must be certain that he or she reads the sensory data of the right network in the right network, while a sensor node needs to know that it is providing its services to the right customer. Again, standard security protocols such as SSL / TLS provide authentication, although in most cases sensor nodes must handle digital certificates and belong to some public key infrastructure (PKI). However, authentication of interacting devices may not be sufficient for specific scenarios, in which the user attempting to access the node must also present certain credentials.

The authorization problem is closely related to the authentication problem. Once a user of the network (human user or machine) proves his identity, it may be necessary to check if this user has the necessary rights to access the sensory data. For example, everyone can access public data such as the temperature of a university building, while other data streams such as the radiation level of a nuclear power plant should be accessible only to those with sufficient authority. Not only the access to data must be controlled, but also the granularity of the data. Beyond the data, it is also necessary to monitor the control operations: the sensors must be reprogrammed and rebroadcast remotely, and only authorized personnel must be able to do so.

Responsibility and availability are other specific issues that can arise when integrating sensor networks into the Internet. Because a heterogeneous set of users will access sensor network services, it would be important to record interactions with these users to detect or re-create security incidents. Unlike Internet servers, most sensor nodes have a limited amount of memory for storing transaction status information and descriptions of data changes. Therefore, it is necessary to devise an optimized way to store this log data, either by storing the information in a more powerful server or by choosing the type of interactions to be logged.

Finally, since the main objective of a sensor network is to provide sensory data to its users, the availability of this data is another sensitive subject that needs to be analysed in detail. If a sensor node is unavailable due to a hardware error or other malfunction, no external host will query it for data flow. In addition, the node may be subject to external attacks by malicious hosts and these attacks can affect the node in different ways: from the saturation of its scarce resources to

the depletion of its battery. It is therefore essential to design mechanisms that can protect nodes and ensure the availability of data flows and other network services.

While protecting the interactions between sensor networks and the Internet, it is essential to keep in mind the inherent limitations of sensor node hardware. In most cases, constrained sensor nodes may not have the resources to implement complete Internet protocols. Even more, sensor nodes must spend as little energy as possible during normal operation if they are deployed for long periods of time. Therefore, the protocols must be adapted to work optimally in these limited devices. Even more, consider whether deploying the sensor network can delegate security tasks to more powerful devices such as the base station.

3.4. Security Achievements and Open Issues

3.4.1. Securing Wireless Sensor Networks

The researchers thoroughly studied the safety of sensor networks. Although outstanding issues remain to be solved, it is now possible to create a sensor network that meets a minimal set of security properties. In addition, the secure integration of sensor networks and the Internet is an underdeveloped research topic, although it is indeed possible to put in place an infrastructure that is solid enough for sensor networks to interact safely with Internet hosts. The purpose of this section is to show these results, as well as some outstanding issues.

To comply with a minimal set of security properties on their internal operations, sensor networks must use cryptographic primitives, support key management systems, and provide automatic configurability. The sensor hardware is fully capable of using cryptographic primitives, such as symmetric key cryptography, public key cryptography, and hash functions. The IEEE 802.15.4 standard provides hardware support for the AES-128, although this algorithm and other symmetric key algorithms can be fully implemented in SW. In [94], the implementation of AES-128 requires 8 KB of ROM and 300 bytes of RAM. Note that there are other block codes and flow codes such as Skipjack[94] and RC4 [95] that have lower memory requirements, 2,600 and 428 bytes of ROM, respectively.

While sensor nodes have generally been considered highly constrained devices that cannot implement any type of PKC, new research findings have challenged this assumption. Using Elliptic Curve Cryptography (ECC), it is possible to support Encryption and Decryption (ECIES), Signature and Verification (ECDSA) and Key Establishment (ECDH) in a sensor node. However, the memory and computing requirements of the ECC are quite high: in a standard class II machine, the execution time of an ECDSA signature is about 2 seconds and the algorithm consumes 17 KB of ROM and 1.5 KB of RAM[96]. Finally, sensor nodes can implement hash functions such as SHA-1 in only 3 KB of ROM.

By implementing ECDH in sensor nodes, it is possible to solve the problem of distributing link layer keys in a sensor network. In addition, ECC can be an extremely useful tool for integration with the Internet. However, the functionality of the application may be so complex that the sensor nodes do not have enough memory to implement this feature, or the application requirements do not require the complexity of ECDH. Key management systems are still a hot topic of research, although with the current state of knowledge it is possible to meet the requirements of simple and small networks.

Cryptography can serve as the basis for essential security services, such as confidentiality, integrity, and authentication. However, these services are not enough to support one of the specific properties of the sensor network: the ability to configure itself. To be completely autonomous and able to support itself, it is essential that the nodes are aware of their

environment, that is to say that they recognize certain events that can affect the behaviour of the network. Currently, there are light situational detection mechanisms for detecting abnormal events occurring in the sensor network[97]. These mechanisms can also be used for accountability purposes and support for securing the "core protocols" of a sensor network: routing, aggregation, and time synchronization.

3.4.2. Securing the Integration Strategies

3.4.2.1 Front-End Solution

Although sensor networks must be sufficiently secure on their own, it is necessary to protect interactions between networks and the Internet. In a front-end proxy solution, the base station acts as a representative of all sensor nodes. It provides all the functionality of the network, behaving like an Internet host. Therefore, most protection mechanisms can be implemented and deployed in the base station. Since the Internet and the sensor network are logically separated, it is possible to protect the exchange of information between an Internet host and the base station using one of the existing security standards, while any interaction between base and sensor nodes can use simpler security approaches. In addition, the sensor network can implement specific mechanisms (for example, sensor redundancy) that best exploit the specific characteristics of the network.

Note that in this case, the base station becomes a single point of failure: if the base station is attacked successfully, an adversary can have access to all information flows. In addition, in case of malfunction of the base station, the sensor network will be completely inaccessible. One possible solution is to use multiple base stations to improve network availability in the event of base station failure and to include new features such as load balancing. However, new challenges, such as maintaining the consistency of information between all base stations, will need to be addressed.

In this front-end solution, Internet hosts can negotiate and establish end-to-end secure channels with the base station using standard protocols such as TLS in the transport layer while the base station and the sensor nodes may use simpler mechanisms such as pre-negotiated shared keys, or even public key cryptography if it is available. With respect to authentication, all nodes being considered under the control of the base station, this base station can authenticate (for example, present its own digital certificate) on behalf of its sensor nodes. User authentication is also managed by the base station, using public key certificates or other authentication mechanisms such as pairs (user, password) or complex protocols such as Kerberos.

About authorization, the base station can analyse the credentials presented by users (for example, attribute certificates) or check if the user is allowed to perform certain operations (using lists access control, for example). Then, the base station can change the provisioning mode of the network services: filtering the data provided by the sensor nodes, allowing the user to modify only certain configuration values of the motions, etc. With respect to liability, there may be close collaboration between the sensor nodes and the base station to monitor and monitor the actual state of the network. The base station can store any interaction between the hosts and the nodes, and can also extract and analyse any behaviour related information from the nodes themselves. Finally, the front-end proxy solution mitigates some issues such as storing historical data and the availability of failed nodes. The base station may behave as a "cache server" by storing the data of the sensor nodes, although it requires polling of the sensor nodes even if there is no request for data from external hosts. In addition, it can monitor whether a given node is accessible or not, transmitting the control information if the node becomes available again.

3.4.2.2 TCP/IP Solution

Considering the Internet and the sensor network as separate entities, it is not necessary to use the already limited resources of sensor nodes to implement expensive Internet standards. However, this situation is changing in the TCP/IP overlay solution, where the sensor nodes become Internet hosts. As a result, the sensor network should no longer be treated as an independent entity and the protocols and security mechanisms used by the Internet hosts should also be supported by the sensor nodes.

Starting in 2008, it is possible to send IPv6 packets over an IEEE 802.15.4 network using the 6LOWPAN specification [98]. For the security of the link layer, 6LOWPAN recommends using the security mechanisms defined at the link layer level by the IEEE 802.15.4 standard. However, for network layer security, IPsec is currently not supported and it is unclear whether the sensor nodes will be able to support the use of these low-level security mechanisms with end to end properties. In fact, using a full IPsec implementation for sensor nodes is discouraged, and even some inbound standards only define a simple transport level security greater than UDP. 6LOWPAN advises identifying the relevant security model and a preferred set of encryption suites appropriate to the constraints.

Although no end-to-end secure channel is supported at the network layer, most applications may still need to create such a channel to protect the flow of information. This problem can be partially solved by providing security at the transport layer, using the TLS/SSL standard. A prototype called Sizzle, developed by Sun Microsystems in 2005 [99], serves as proof of concept. To save enough resources, Sizzle only implements a relatively small set of cryptographic algorithms. In addition, there is no certificate analysis code. Customers may need to authenticate using other mechanisms, such as passwords.

Specifically, with respect to user authentication, it may not be possible to store the user's identity information (i.e., user and password pairs) in a sensor node. In this case, all sensor nodes belonging to the same network must create a mechanism to store and manage these credentials. The same problem applies to the authorization of the user: it would be necessary to maintain the access control model in a distributed form, which is an extremely difficult task. One possible solution is to use Kerberos. For authentication, sensor nodes simply need to check the ticket provided by the ticketing server. Kerberos can also pass authorization information generated by other services. Of course, Kerberos requires the permanent availability of a central server and all devices must keep their clocks synchronized loosely. Note that PKC solutions (certificates of identity, attribute certificates) can also be applied if they are supported.

The low storage capacity of highly constrained nodes significantly hampers network accountability. The sensor nodes must be intelligent enough to detect abnormal conditions caused by hosts accessing its services and store only information related to these incidents. In addition, the storage capacity partially influences availability: only one node can store its readings for a limited time. As a result, the historical data must either be stored elsewhere or summarized in one way or another.

3.4.2.3 Gateway Solution

Some of the issues associated with the TCP/IP solution can be partially resolved by using the gateway solution. The gateway (i.e., the base station) can play the role of storing the responsibility information relating to the interactions between hosts and nodes. It can also store the historical data of the nodes, if they run the risk of running out of storage space. In addition, it can improve network availability by acting as a "cache server" or as an intelligent redirector, as in the front-end proxy solution. On the other hand, if end-to-end secure channels are

negotiated, it should be necessary to implement non-trivial mechanisms in the gateway to analyse the information between an Internet host and a sensor node.

3.5. Related Works to the WSN - Cloud Integration

The potential of gathering the data of WSN is very high but has limitations in terms of storage and processing power. On the other hand cloud computing does not have any restriction in terms of storage and processing power. So if the advantages of both the technologies are considered, then the WSN - Cloud integration can figure out many problems. We discuss here some issued related to the integration of WSN and Cloud and the detailed investigation of current work in this area.

3.5.1. Integrating Wireless Sensor Networks with Cloud Computing

A novel framework integrating cloud computing with the WSN has been proposed in a research project entitled “*Integrating Wireless Sensor Networks with Cloud Computing*”[100]. This project aims to facilitate the shift of data from the WSN to the cloud environment for use for scientific and economic purposes. Users can send their requests via three service models (IaaS, PaaS and SaaS) from the archive, archiving is performed periodically by collecting data from WSN to Data Center (DC). As shown on the *Figure 28*, this framework consists of four

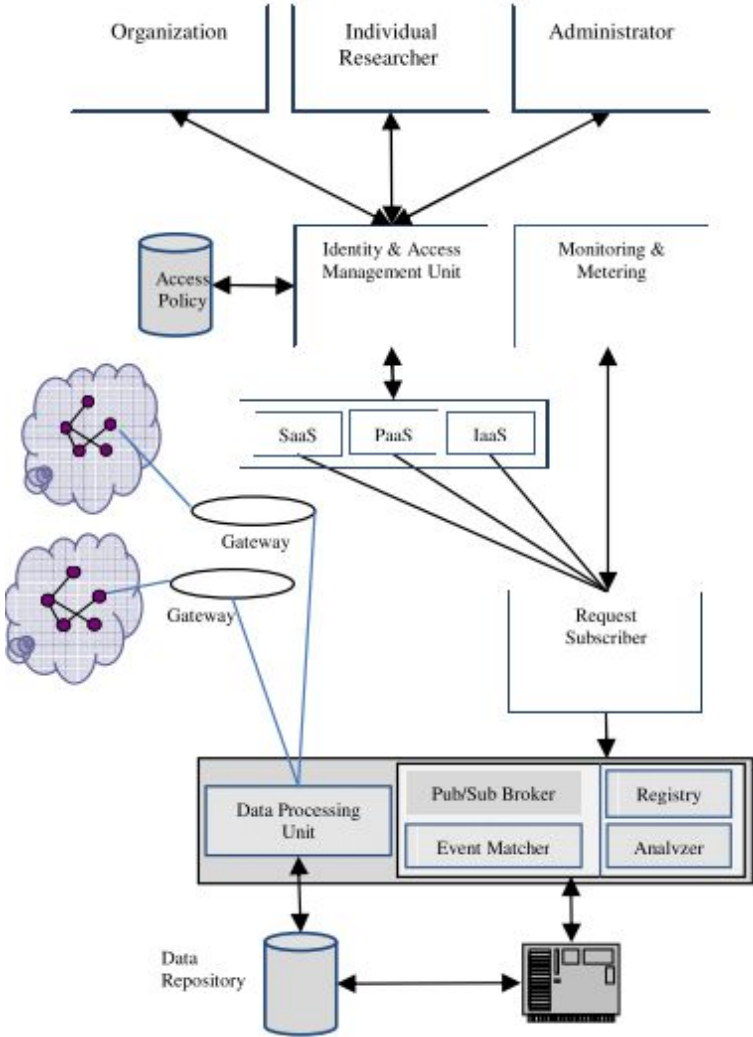


Figure 28 Sensor-Cloud Integration Framework [100]

components namely the Data Processing Unit (DPU), the Pub/Sub broker, the Request Subscriber (RS), the Identity and Access Management Unit (IAMU) and the Data Repository (DR).

Whenever the data is available in the WSN, they pass through a gateway to the DPU, this processing unit puts them in a storage format before sending them to the DR. To be able to use the system, a user must first connect to the cloud through a secure IAMU in order to obtain the access based on the security policy of his/her account. After a user has been granted access to the cloud, he/she can send another request to access the data. This data request must be sent to the RS in order to create a subscription on the basis of the request and RS forwards this subscription to the Pub/Sub Broker. The DPU identifies the data received in the cloud from the WSN and then creates an event of the data ready to be published to the queue at the Pub/Sub Broker. When an event is published, each subscription to the Pub/Sub Broker is evaluated by an event matcher and when the user request corresponds to the published event, this data is made available to the user after some processing is carried out if necessary.

3.5.2. WSN Integrating with Cloud Computing for Patient Monitoring

Another proposed architecture is untitled as “*Wireless Sensor Network Integrating with Cloud Computing for Patient Monitoring*”[63], this project is in the context of medical surveillance for example when patients need to be followed during the way in the ambulance to the hospital or during the surgical operations, recovery rooms, intensive care or even in emergency situations. Thanks to high speed internet, the hospital needs a basic level of subscription in the cloud known as Software as Service (SaaS) because it is easy to pay as the hospital pays its internet bills or electricity by paying only what has been used. In this cloud-based hospital management project, the use of an application programming interface is needed to connect emergency workers in order to have access to stored data in the cloud and to connect the ambulance. Doctors are allowed to visualize important data from the ambulance via the hospital’s emergency ward and record it into the patient's databases.

The proposed system known as the Wireless Body Sensor Network (WBSN) provides the different functionalities like patient identification or physiological parameters by using wireless sensor nodes and storing the obtained results into QualNet software located in the cloud in order to improve the monitoring. The main contribution brought by this project is based on the programming of the agents to carry out some processing on the sensed data leading to reduce the network traffic as well as the response time of the network. This project uses a community cloud under the control of multiple organizations that have a common interest such as the health care facilities.

The proposed architecture includes four agents namely aggregator agent, patient agent, nurse agent and doctor agent. A group of patients wearing the sensor nodes collecting the physiological parameters is connected to a patient agent acting as a cluster head, and the different cluster heads are connected to a wireless access point conveying the information directly to the base station acting as an aggregator agent. When the Aggregator detects anomalies in the received data, it immediately alerts the doctor agent and the nurse agent while sending these data to the cloud for processing and storage. The doctor and nurse agent are situated on a handheld mobile device to be reachable at any time, when a patient is assigned to a doctor, this information is displayed on his mobile device. These doctor and nurse agents are used to send alerts to medical agents to allow queries regarding patient information including current and past physiological parameters.

3.5.3. Cloud Computing System Based on Wireless Sensor Network

In the “*Cloud Computing System Based on Wireless Sensor Network*”[101], the authors wanted to propose to users a system applicable in agriculture to monitor pH, temperature, humidity, etc. using web services to store these big data into the distributed SQL database proposed in their cloud system.

The proposed cloud system in this project consists of using a master server computer and four other virtualized slave server computers for data storage and distributed computing process. By programming stored procedures in T-SQL, basic instructions for insertion, update, deletion, and selection can control the data in the relational database. This project uses C# as the programming language of web services, to make easier the communication of this language and the database, the authors have used LINQ[102] making T-SQL easily readable in C#. Thanks to the web services interconnecting the different platforms, the user requests arrive at the cloud and the results are formatted in XML.

3.5.4. Integrating WSN into Cloud services for real-time data collection

“*Integrating Wireless Sensor Network into Cloud services for real-time data collection*”[103] is an extensible and flexible research project integrating WSN with the cloud using an interoperable application based on RESTful web services. The authors of this project proposed an architecture subdivided into three layers (see the *Figure 29*): the Sensor Layer, the Coordinator Layer and the Supervision Layer. The sensor layer is mainly composed of sensor nodes (XBee ZB platform) directly connected to the environment. These nodes form a Mesh topology to gather the data to be sent to the coordinator layer through the sink node. The management of the data coming from the sensor network is carried out by the coordinator layer, these data are temporarily stored in a buffer before being sent to the supervision layer at the predefined intervals. The supervision layer contains the base station and a web server connected to the internet in order to let the data from the sensors can be accessed remotely. These data are stored in a database on that server which offers a web interface to allow end users to perform a set of the management operations and statistics on the sensors data. In this layer, the authors used an HTTP service from the Open.Sen.Se cloud platform that provides a RESTful API for publishing and accessing sensor data.

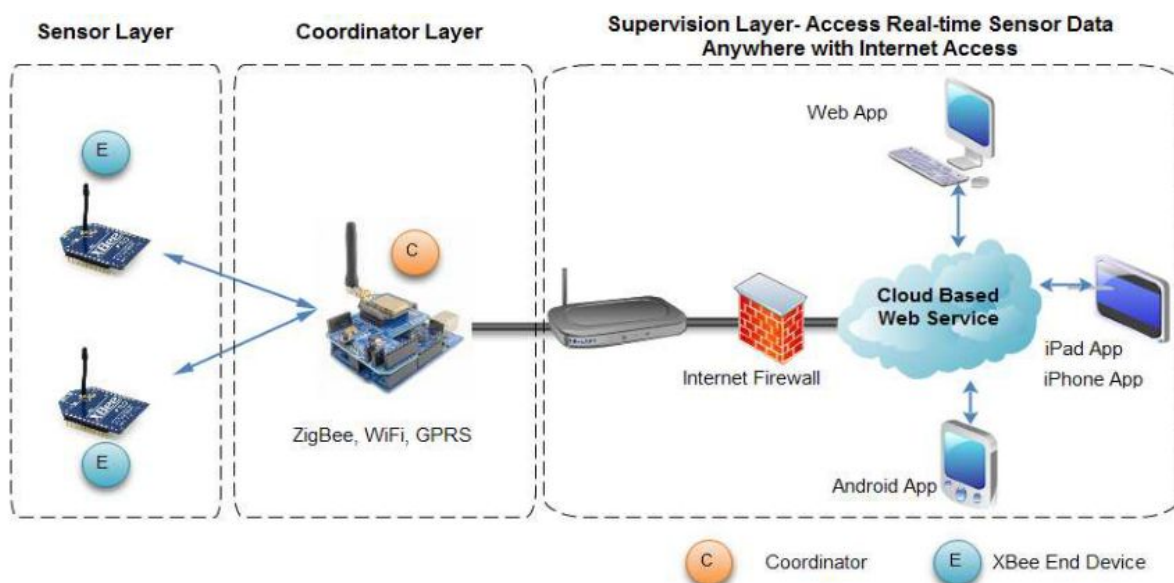


Figure 29 Layer-based WSN-Cloud integration for real-time monitoring [103]

According to the authors, the access to cloud services must be easy, direct and interoperable. Accordingly, they proposed an application programmable interface (API) allowing easy implementation on any platform and on any development environment. For this, they have chosen web services as the most open and interoperable means of communication to provide remote services or allow applications to communicate with each other. As was previously written, the authors used Open.Sen.Se which is an open source Internet of Things platform offering APIs for storing and retrieving data from objects and sensors formatted in either JSON, XML or CSV. In addition to storing data, this platform allows numeric data processing such as average, median, summation, etc. As we know, there are two types of web services namely SOAP and REST but for some unexplained reason, the authors have adopted for REST web services using standards such as POST and GET requests returning responses in JSON format for communicate the base station and the Open.Sen.Se server. The authors explained the choice to use JSON as the data format in that JSON is easier to parse by the machine than the XML format, unlike human beings who easily analyze XML than JSON. As soon as the data has arrived to the Coordinator from the sensor nodes, an HTTP POST request is sent from the base station to a specifically predefined URL containing that data. Each entry in the database is stored with date, timestamp and unique identifier.

3.5.5. Combining Cloud Computing and Wireless Sensor Networks

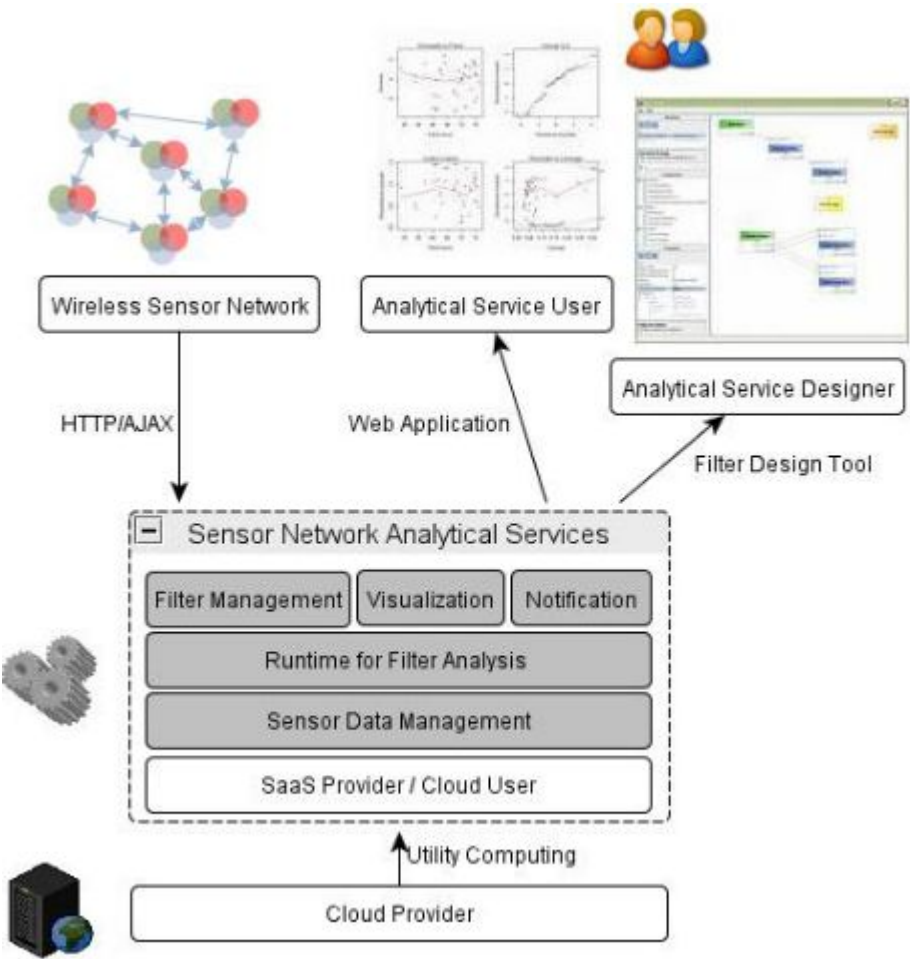


Figure 30 Cloud computing based sensor data analysis environment [104]

Another model which combines the concept of wireless sensor networks with the cloud computing paradigm has been presented in “*Combining Cloud Computing and Wireless*

Sensor Networks”[104], showing how both can benefit from this combination. The proposed cloud computing model is mainly based on pipes and filters. Pipes do not transform any data, they buffer the data to provide a uniform interconnection mechanism of filters. Filters do the specific processing and transformations to input data such as deletion. A filter takes the data as input and makes it available at the output, for example a filter that stores data in a database when a specific values exceed a certain threshold. In order to collect and analyse data from the WSN, the proposed architecture must provide the following services (check the *Figure 30*): Sensor Data Management, Runtime for Filter Chains, Filter Chain and Filter Management, Visualization and Notification Service.

As the cloud (PaaS) provides the various ways of storing data, for example Google App Engine (GAE) offers the BigTables for persistence of data and Microsoft Azure offers Blobs, queues and tables, the Sensor Data Management service is responsible for managing the sensor data within the cloud by creating a flexible data access layer leading to a high level of abstraction so that the mechanism for data persistence can be easily exchanged. Runtime for Filter Chains are designed to represent specific analysis and a filter chain is configured for each filter. A filter chain is a reliable runtime environment for filters and runs various user-defined filters. In order for a user to add, delete, and aggregate existing filters in a combined filter, a management service is needed to manage filters called Filter Chain and Filter Management. When a number of the filters becomes too big, the filters are combined to provide an aggregation mechanism capable of running the filter chains in parallel providing a flexible configuration mechanism. Since cloud computing is a storage and data-processing environment, the visualization service offers the various pre-defined and user-designed views of data and the analysis of results. By using language like OLAP and data warehouses, visualization can be implemented. In order for external applications and services to be informed when an unusual event such as fire is occurring, the notification service is able to perform this task. This notification service provides a set of predefined user-designed rules. Such an event could be, if several sensor values exceed a certain predefined vibration threshold, it could be an indication that a gas pump failure is approaching.

To demonstrate the powerful of this approach, the authors have implemented in an application to collect and send to the cloud data on the power consumption from different devices. At the basis of the WSN, they used Sentilla's JCreate sensor nodes that measure power consumption. Through the use of wireless links built from the ZigBee protocol, each sensor node sends the sensed data to its near neighbour as a message called PowerMessage containing the identification number, the message number which is continuously increased and the electrical potential provided by the current sensors. As the sensor nodes are organized into a mesh topology, each sensor node receives and retransmits the received data to the base station. A USB gateway attached to a computer resembles the data sent by the sensor nodes in this computer serving as an intermediate point between the WSN and the remote cloud computing infrastructure.

The data pushed up to the cloud is first processed and then stored by the appropriate filter chains, a simple filter chain for example could put the data of the sensors into the persistent storage. Finally, the data can be used by analysis and visualization services implemented from web services using Java or .NET as the authors had chosen Google App Engine and Microsoft Azure as a cloud platform.

3.5.6. Synthesis of the Proposed WSN-Cloud Architectures

From the architectures previously presented, some of them have common points with which it is possible to establish a comparison in order to know what the common core to build such

architecture. The following *Table 1* gathers the comparable points for the research works previously mentioned.

Table 1 Comparison between presented architectures related to this thesis

Architecture	Middleware	Cloud Infrastructure	End user applications	Security
[100]	Web Services	-	-	AAA by Kerberos
[63]	Web services	SaaS	Web browser	-
[101]	.NET based web services	IaaS (private cloud based on virtualization)	C# and Java Interface	-
[103]	RESTful web services	PaaS (Open.Sen.Se)	Web browser	-
[104]	.NET and Java based web services	PaaS (Microsoft Azure and Google App Engine)	C# and Java Interface	-

The previous section summarizes some research work related to this thesis, it seems obvious that the authors have some common points about their proposed architectures as the use of web services for the interconnections of the different components of the system. However, it is obvious that no system among them is 100% efficient considering its architecture or its implementation. According to an analysis of each of them, there are some remarkable weakness for each presented architecture:

- First of all, the use of web services as the middleware to interconnect WSNs, the cloud and end users. All authors choose the web services as a way to communicate the different sites of their architectures but without specifying the reason for this choice. According to comparative studies between the most used and the most well-known middleware (Web Services, Java RMI and CORBA), the results have shown that web services are slow, consume more memory for programs and use too much bandwidth compared to other middleware [105], [106]. However, the web services are the most flexible, the most scalable and the easiest to implement, from these reasons the authors would have favoured the ease of implementation rather than the reliability.
- Secondly, the authors did not want to develop the security concept in their architectures, for this reason their architectures seem to be incomplete or vulnerable in term of data privacy. The proposed architectures combine the WSN, the cloud as well as end-user applications, and the authors do not specify whether the data transiting from one side to another are protected by any kind security technique.

Chapter -4-

Integration of the WSN with the Cloud Computing

This chapter proposes a WSN integration architecture with cloud based on interoperability technologies. For the proposed architecture, it was considered that the WSN, the cloud as well as the end-user applications are components of the same distributed system. In order for these heterogeneous components to communicate each other, an interoperable technology is required. Consequently, before choosing an interoperability technology to be used for interconnecting all components aforementioned, a comparative survey was carried out between CORBA, Java RMI and the web services in order to pick out the technology that best fits the challenges of WSNs. The results obtained showed that Java RMI consumes fewer memory resources, less bandwidth on the network and faster response time, which allowed to choose Java RMI as middleware to communicate the Cloud with other components of the system.

4.1. Distributed System based on Cloud and WSN

Wireless Sensor Networks (WSN) has received considerable attention from universities and industry around the world. A large number of research activities have been conducted to explore and resolve various design and application issues, and significant progress has been made in the development and deployment of these networks. It is expected that in the near future, WSN will be widely used in various civil and military fields, and will revolutionize the way we live, work and interact with the physical world. Access to the applications and data are two important features of the WSN-Cloud architecture. Ubiquitous applications can be transmitted via WSNs, while automation can be used to facilitate low-cost data collection and distribution. Cloud is an efficient and cost-effective solution that can be used to connect, manage and track everything using integrated applications and custom portals.

As the WSN can be built by thousands of sensor nodes, it includes a large number of information sources, which generate a huge amount of semi-structured or unstructured data. The cloud is considered one of the most cost-effective and appropriate solutions for managing the huge amount of data created by the WSN. In addition, it creates new opportunities for integrating, aggregating and sharing data with third parties. WSN platforms are characterized by limited processing capabilities that prevent complex on-site data processing due to lower computing capabilities and low energy. Instead, the collected data is transferred to nodes with high capabilities (cluster head) for aggregation and processing. The Cloud offers unlimited virtual processing capabilities, predictive algorithms and data-driven decision-making ready to be integrated into the WSN in order to increase revenue and reduce risk at a lower cost.

Sometimes the WSN is composed of heterogeneous sensor nodes and protocols. As a result, reliability, scalability, interoperability, security, availability, and efficiency can be very difficult to establish. Integrating WSN into the cloud can solve most of these problems. In addition, the cloud offers other features such as ease of use and ease of access, with low deployment costs. However, the management of thousands of sensor nodes in a network, the processing of the sensed data, their storage and backup for future use must rely on new models of IT infrastructure in order to go faster, to be more agile and safer. To enable WSNs to communicate with other information system environments IaaS platforms should play a key role. In addition, the cloud

provides the processing power needed to analyse data from WSNs, both in volume and in real time.

With the Internet of Things, the objects connected to the network transmit their data to the cloud for storage and processing but the constraints are not identical with the wireless sensor network. In the Internet of Things, an object (refrigerators, cars, electric lamp, etc.), sends directly its data (temperature, fuel level, electric current, etc.) to the processing or storage center to make a meaningful interpretation and make a decision based on the received data. If each object is considered as a communicating node on the network, there would be a difference with the nodes of the wireless sensor network because each node of the WSN does not send its data directly to the processing or storage center, rather to the base station that behaves like the gateway of the wireless sensor network with the external networks. Each connected object has a connection interface often equipped with an IP address to communicate, the manufacturers of these objects integrate communication middleware for the traditional networks. In other words[107], the WSN were conceived mainly for local networks to collect data from neighbourhood but its applications may take advantage of Internet even without being an essential requirement for WSN. However, IoT integrates several technologies that already exist such as Cloud Computing, Middleware and end-user applications. Unlike the WSN, the internet is essential requirement for IoT.

Wireless sensor networks, the cloud, and end users need to be viewed as a distributed system where each component is used to complete a well-defined task. By definition, a distributed system is a system having a set of communicating entities, installed on an architecture of independent computers connected by a communication network, in order to cooperatively solve a common application functionality. In other words, a distributed system is defined as a set of physical and logical resources geographically dispersed and connected by a communication network in order to achieve a common task.

A distributed system consists of autonomous devices (computers, tablets and smartphones), connected through a network and distribution middleware, which enables devices to coordinate their activities and to share the resources of the system (In programming, middleware enables computer applications to manipulate remote objects by invoking their methods). The typical application of distributed system requires the exchange of messages formatted according to a pre-established convention; *it can be a system moving data produced in one geographical area and required in another area as the case of wireless sensor networks toward the end users.*

4.1.1. Overview of Distributed System Architecture

Distributed systems are built from the operating systems and network software that already exist. In a distributed system, there is a collection of autonomous computers, interconnected by a computer network and distribution middleware. For computers to become autonomous, there must be a clear master/slave association between them. In the same distributed system, users must perceive the system as a single, integrated computing facility through middleware that allows computers to coordinate their activities and share available resources in the system.

In order for the different and remote physical locations to communicate each other (*Figure 31* shows a simple architecture of a distributed system), a middleware must act as a gateway connecting the distributed applications developed in different programming languages to be executed on different hardware platforms, different network technologies as well as different operating systems. The middleware is developed under the agreed standards and protocols to give standard services like naming, persistence, concurrency control that ensures the production of accurate results for simultaneous processes and obtain the results as quickly as possible, the event distribution, the authorization by granting rights access to resources, security, etc.

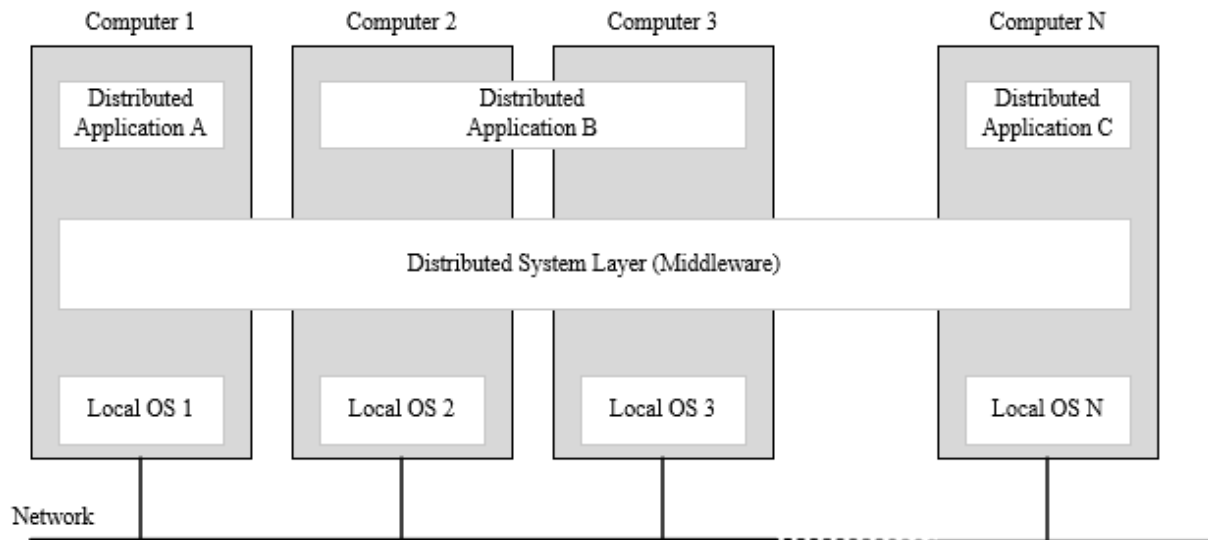


Figure 31 A Distributed System

A distributed system can be visualized either as by physical components or as a computation point. The first is called the physical view while the second is called the logical view.

- Physically, a distributed system consists of a set of nodes communicating with each other (computers, smartphones, PDAs, tablets, etc.) via a computer network. In this model, loosely coupled nodes do not share their memories and communicate by transmitting messages from one node to another over the network by the use of communications protocols.
- Logically, the core of the network is considered as fully connected. This model contains a set of concurrent processes and communication channels between them (The processes communicate by sending messages to each other).

A distributed system can be synchronous when a process continuously executes the scheduled operation within a specified time, it is asynchronous otherwise. To detect the failure in the synchronous distributed system, it suffices to notice the lack of response to a process execution. That's why there are time-delay-based techniques for discovering communication failures in a distributed system.

4.1.2. Main Characteristics of a Distributed System

For the users to have the best performances, a distributed system must possess at least the following characteristics:

- **Fault-Tolerant:** The goal of fault tolerance is to increase reliability and availability. In a distributed system, there is a large number of hardware and software that can fail at any time. Such a fault tolerant system should be able to recover from failure of components without performing erroneous actions. Usually, fault tolerance is achieved by performing redundancy in two ways: hardware and software. Hardware redundancy consists of adding extra hardware components to the system to take over in the case of a malfunctioning component. Software redundancy consists of adding extra code instructions for managing the extra hardware components so that they can properly use the interrupted services in the case of component failure.
- **Scalability:** A distributed system must be operational even if some aspects of the system are scaled up. Scaling can be done in many ways such as the number of users or the distance between the nodes furthest away from the system. Three techniques are used

to make a scalable system: replication, distribution, and caching [108]. Replication creates multiple copies of system resources in order to reduce the load on individual servers and improve availability. The distribution firstly spreads the information managed by a distributed service on several servers of the system and secondly spreads the loads across the servers by reducing the number of requests processed by each server. Caching gradually decreases the load on the servers and on the network. The cache is faster accessible when a new request is received. The caching is usually performed by the user, thus reducing frequent requests for services across the network.

- **Openness:** A system is said to be open if it makes possible the interacting with other systems. For example, Web services are software systems designed to support interoperable interaction between machines on a network. An open, scalable system has advantages over a fully enclosed, self-contained system. A distributed system independent of the heterogeneity of the underlying environment, such as hardware and software platforms, is the property of openness. Therefore, each service is also accessible to all users (local or remote) of the system. Implementing, installing, and debugging new services should not be very complex in a system with open characteristics.
- **Security:** Distributed systems must allow communication between programs, users and resources on different computers by imposing the necessary security provisions. A distributed system must at least provide these three security functions: confidentiality, integrity, and availability. Other important security issues are access control and non-repudiation. The security mechanisms implemented should ensure appropriate use of resources by the different users of the system.
- **Transparency:** Users and application developers should perceive a distributed system as a whole rather than a set of cooperating components. The locations of the computer systems involved in a distributed system are hidden for users. Transparency hides the distributed nature of the system from its users and shows the user that the system appears and functions as a normal centralized system. Transparency can be used in different ways in a distributed system:
 - Access transparency: makes it easier for users of the same distributed system to access the resources (locally or remotely) of the system by using the identical operations.
 - Location transparency: describes names used to identify network resources (IP address for example) so that the users of the same distributed system can easily access the system resources from anywhere on the network without knowing where the requested resource is located.
 - Concurrency transparency: allows multiple processes to run concurrently using the shared resources of the system without interferences between them (ATM network for example). In this case, a user does not notice the existence of other users on the network even though they access the same system resource at the same time.
 - Replication transparency: enables the system to have additional copies of the system resources in order to increase performance and availability without users noticing. When a system resource is replicated among multiple locations (Website mirror for example), they must appear as a single resource for the user.
 - Failure transparency: assures the applications that they can complete their tasks even if the failure occurs in one of the components of the distributed system (for example if a server does not respond, the user requests are immediately redirected to another server without users noticing it). It is difficult to achieve

this kind of transparency because in some cases the server responds slowly but the system believes that the server is failing.

- Migration transparency: facilitates the system resources to move from one physical location to another without changing their names. System users are not aware that the resources they use have physically moved from one location to another.
- Scalability transparency: allows the system to remain efficient even with a large number of users accessing it at the same time or a large number of resources is added to the system.

4.1.3. Middleware as the basis of the Distributed System

Middleware is a generic term to describe any application used to bring together or to connect (as a mediator) two separate applications. It is a set of network layers and software services, which allow the dialogue between different components of a distributed application. This dialog is based on a common application protocol, defined by the middleware API. The Gartner Group defines the middleware as a universal communication interface between processes. It truly represents the keystone of any client-server application. The main objective of the middleware is to unify the applications so that the access and the manipulation of all the services are available on the network, in order to make their use almost transparent.

The middleware widely used (also used as technologies of interoperability) are the basis of the objects distributed with Java RMI (Remote Method Invocation), objects brokers to the CORBA standard (Common Object Request Broker Architecture) and Web services based on SOAP standards.

- **Java RMI:** It is only a Java-to-Java technology that allows an application to call methods of remote objects physically away from each other, which are running on separate Java Virtual Machine (JVM). Despite its ease of use, RMI has a limitation because it is almost impossible for integration with objects or applications written in another language different from Java.
- **CORBA:** Unlike RMI, CORBA middleware is not tied to a programming language and it can integrate with legacy systems written in ancient language as well as future languages that support CORBA. CORBA is not tied to a single platform and shows great potential for use in the distributed computing. CORBA allows structures and primitive data (Integer, Double, String, etc.) to be transferred between two remote systems while RMI allows Java objects to be passed and returned as parameters. This allows a new instance to be passed by virtual machines for being launched (mobile code) that does not exist in CORBA for security reasons, so no real code.
- **Web Services:** Unlike RMI and CORBA, there is no concept of referenced objects in Web Services (Web services concerned are Java API for XML-based RPC); rather, architecture based on a stateless server interconnecting the ubiquitous technologies often accessed through a web browser. Thanks to the standards and open protocols, Web services provide interoperability between different software running on various platforms. Despite the advantages of web services, they are less efficient comparably to other distributed computing approaches such as RMI, CORBA.

4.1.4. Comparative Survey between Technologies of Interoperability

Among the middleware aforementioned, a comparative survey has been carried out for this thesis in order to pick out one with the best performances required for real-time monitoring to be used while integrating WSN to the cloud and end users. In this study, the experiment goal

was about to invoke a method “*saveData(Datastore data)*” on a distributed object and to repeat the same experiment ten times for each interoperability technology: CORBA, Java RMI and JAX-WS. This method sends as parameter an object “*Datastore*” containing the fields of the sensed data to be stored in the database. The concerned fields are:

- *nodeName*: a string denoting the data source node’s name. It could be the mac address of the mote.
- *collectingDate*: a string indicating the date and time which the data are collected.
- *dataType*: this field is a string that gives the precision of the sensor type concerned (temperature, humidity, etc.).
- *dataValue*: a float, this field contains the value of *dataType* (it can be a temperature or humidity value).

This study focuses on three criteria, initially the performance related to the time taken to invoke remote “*saveData(Datastore data)*” method of the distributed object (check the *Figure 32*), then the memory used by the program that distributes object (*Figure 33*) and finally the amount of data sent over the network in order to estimate the bandwidth cost (*Figure 34*). Below (on the *Table 2*) are the computer configurations used for the experiments.

Table 2 The experiment configuration of Middleware Comparative Survey

Component	Processor	Processor speed	RAM	OS	JDK version	Network Connection	Bandwidth
Local machine	Intel Core i3	1.8 GHz	4 GB	Windows 8.1 x64	7	Wired via RJ45	1 Gbps
Remote machine	Intel Core i5	2.7 GHz	8 GB	Mac OS X Yosemite	7	Wired via RJ45	1 Gbps

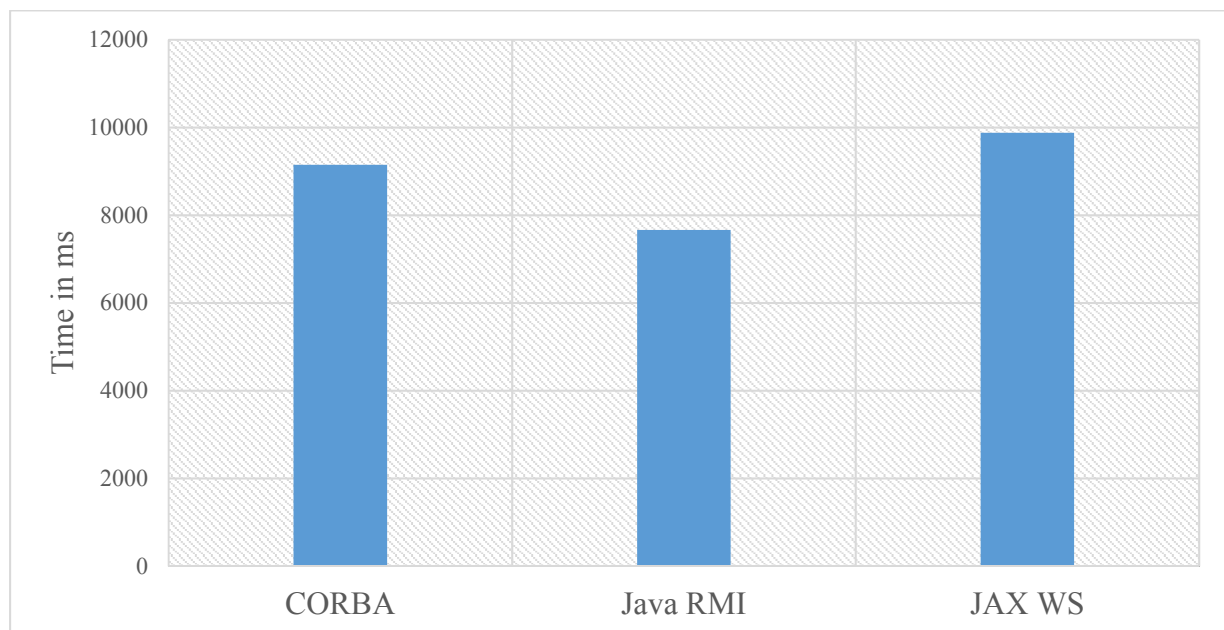


Figure 32 Time required to invoke remote method

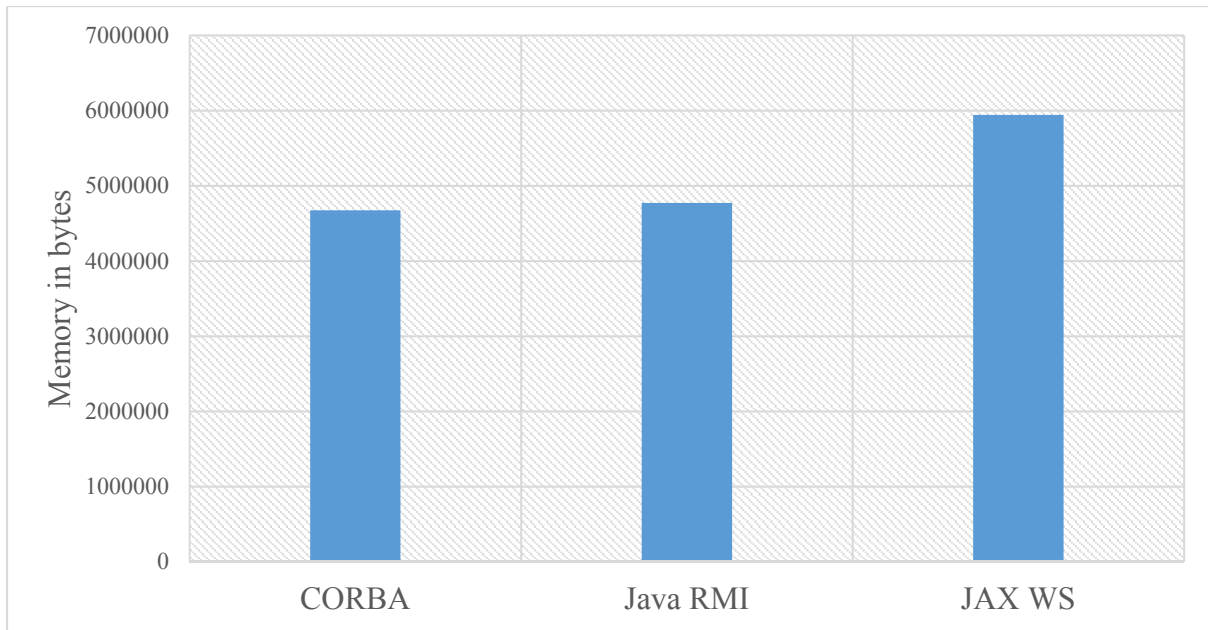


Figure 33 Memory used while invoking remote method

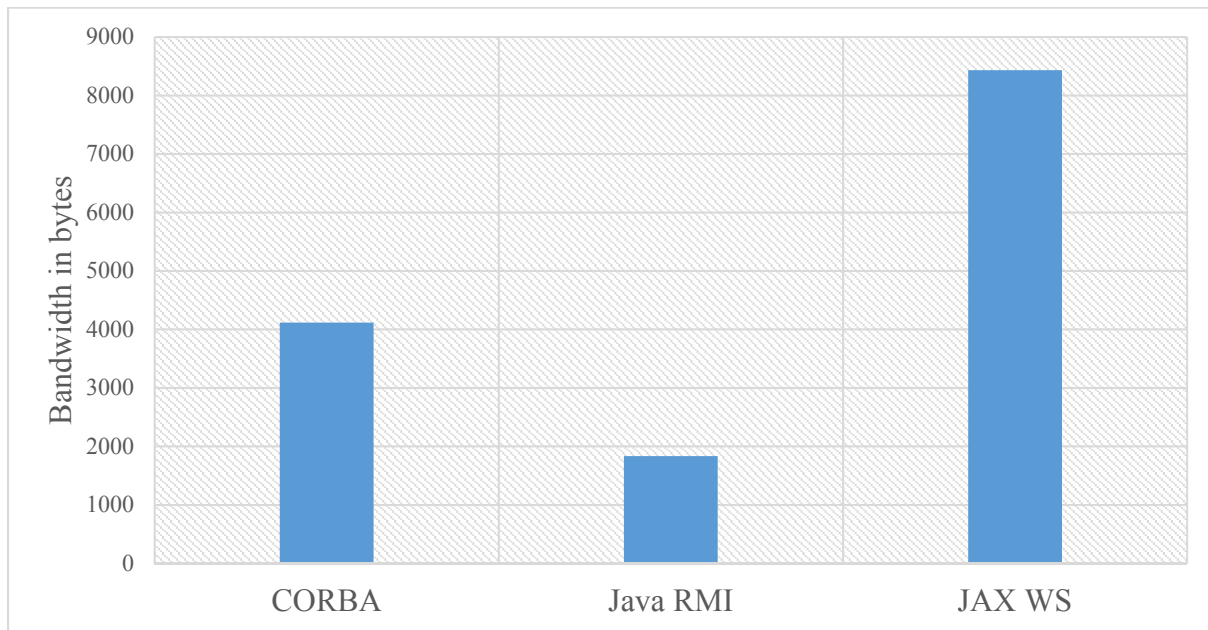


Figure 34 Consumed bandwidth while invoking remote method

Java RMI provides better performances compared to other technologies in terms of invocation time and consumption of network resources. The choice of RMI is also the fact that it can be deployed on different operating systems via the JVM. CORBA uses less memory compared to Java RMI and Web services, CORBA can also integrate multiple programming languages but its GIOP/IOP protocol has some drawbacks such as blockages of the data flow through most firewalls on the Internet. Based on the *Figure 32*, *Figure 33* and *Figure 34*, web services are slower, consume more memory and network bandwidth, thus the web services are not well suited for real-time monitoring application.

4.1.5. Presentation of Java Remote Method Invocation

RMI stands for Remote Method Invocation. It is a mechanism that allows an object residing in one system (JVM) to access/invoke an object running on another JVM. RMI is used to build distributed applications; it provides remote communication between Java programs. It is provided in the package *java.rmi*.

4.1.5.1 RMI Architecture

In an RMI application, there must be at least two programs, a server program (resides on the server machine) and a client program (resides on the client machine). The following *Figure 35* shows the architecture of an RMI application.

- Inside the server program, a remote object is created and reference of that object is made available for the client (using the registry).
- The client program requests the remote objects on the server and tries to invoke its methods.

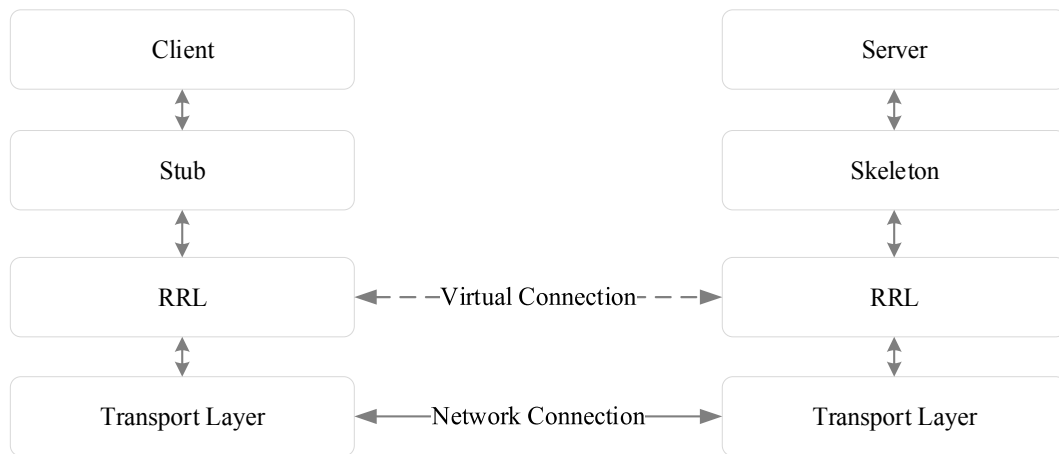


Figure 35 Java RMI Architecture

- **Stub**: is a representation (proxy) of the remote object at client. It resides in the client system; it acts as a gateway for the client program.
- **Skeleton**: is the object which resides on the server side. Stub communicates with this skeleton to pass request to the remote object.
- **RRL (Remote Reference Layer)**: is the layer which manages the references made by the client to the remote object.
- **Transport Layer**: connects the client and the server. It manages the existing connection and also sets up new connections.

4.1.5.2 Implementation of RMI

To create an application with RMI just do the following:

- Define the remote class. It must derive from *java.rmi.server.UnicastRemoteObject*.
- Define the interface for the remote class. This one must implement the interface *java.rmi.Remote* and declare the global public methods of the object, i.e. the declared methods are shareable. In addition, these methods must be able to throw an exception of type *java.rmi.RemoteException*.
- Create the classes for the stub and the skeleton.
- Launch the RMI registry and launch the server application to instantiate the remote object. This one during instantiation will create a link with the register.

- Create a client program that can access the methods of an object on the server using the *java.rmi.Naming.lookup()* method.
- Compile the client application.
- Instantiate the client

4.1.5.3 Running an RMI program

- When the client makes a call to the remote object, it is received by the stub which eventually passes this request to the RRL.
- When the client-side RRL receives the request, it invokes a method called *invoke()* of the object *remoteRef*. It passes the request to the RRL on the server side.
- The RRL on the server side passes the request to the Skeleton (proxy on the server) which finally invokes the required object on the server.
- The result is passed all the way back to the client.

Whenever a client invokes a method that accepts parameters on a remote object, the parameters are bundled into a message before being sent over the network. These parameters may be of primitive type or objects. In case of primitive type, the parameters are put together and a header is attached to it. In case the parameters are objects, then they are serialized. This process is known as *marshalling*. At the server side, the packed parameters are unbundled and then the required method is invoked. This process is known as *unmarshalling*.

4.1.5.4 RMI Registry

RMI registry is a namespace on which all server objects are placed. Each time the server creates an object, it registers this object with the RMI registry (using *bind()* or *reBind()* methods). These are registered using a unique name known as bind name. To invoke a remote object, the client needs a reference of that object. At that time, the client fetches the object from the registry using its bind name (using *lookup()* method). All processes are illustrated in the *Figure 36*.

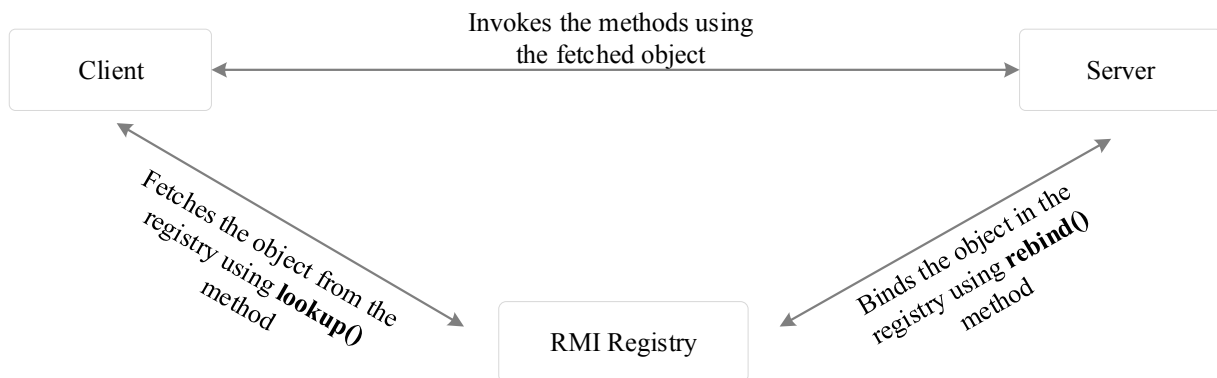


Figure 36 Illustration of RMI Registry process

4.2. The Proposed WSN-Cloud Architecture

4.2.1. Physical Architecture

Like all other architectures presented in the literature review 3.5, this proposed architecture follows quite the same guidelines but with some specifications about the operational and service cases. Physically, the proposed architecture (shown in *Figure 37*) is divided into four layers namely the *Sensing Area (SA)*, the *Gateway (GW)*, the *Cloud Infrastructure (CI)* and the *End User (EU)*.

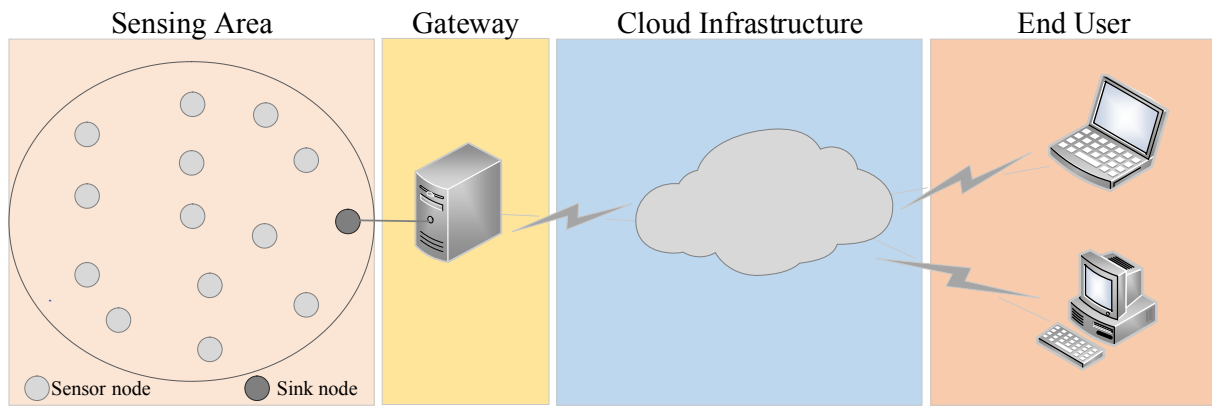


Figure 37 Physical WSN-Cloud Architecture

The Sensing Area is generally composed by the wireless sensor network platforms, which can be deployed anywhere in order to interact directly with the environment for collecting useful data required by various and multiple applications. In fact, the sensor nodes communicate with each other to route data end-to-end to the sink node. There may be more than one platform for multiple deployment areas, so multiple sink nodes. However, it can be a simple sensor network deployed on a site with several sink nodes to avoid congestion of the network caused by excessive transmissions. These sink nodes are directly connected to the gateway (which can be a workstation) via its COM or USB port used to transit the data so that they be acquired by the GW.

The Gateway is a single workstation or powerful computer that mediates between Sensing Area and the external networks. On the one hand, this gateway must be able to understand the protocols used on external networks such as the Internet and even the intranet. On the other hand, the gateway must be able to acquire data from Sensing Area through one or more of its serial ports. Therefore, the gateway must run an application on its behalf to route the sensed data toward the End Users through the Cloud Infrastructure. The Gateway's application serves as well as to send the requests from the End Users to the Sensing Area. Since there can be multiple WSN platforms, the number of Gateways must be less than or equal to the number of WSN platforms.

There are many ways to define the cloud when considering the services that it offers, according to the NIST (National Institute of Standards and Technology)[81], three service models can be offered on the cloud (see the *Figure 17*): Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). These three service models must be deployed on infrastructures that have the five essential characteristics to be considered as Cloud Computing.

In this architecture, we are interested in IaaS as our Cloud Infrastructure. So, the prime example of this type of service is *Amazon Web Services (AWS)*, which provides computing infrastructure called Elastic Compute Cloud (EC2)[109] allowing third parties to rent cloud servers on which they can install and run their own applications. Our cloud infrastructure is generally composed of an application running on a type-1 hypervisor (shown on the *Figure 38*), this application is connected locally (or to the separate virtual machine) to a database for storing data from the sensing area. In addition, it allows end users to access the sensed data (real-time or stored data).

In this case, we have to rent computing and storage resources, network capabilities and other necessary resources (load sharing, firewall, cache) so that we have the opportunity to deploy any type of software including operating systems. We have not to manage or control the underlying cloud infrastructure but we have the control over operating systems, storage, and applications. We can also choose the main features of network devices such as load sharing, firewalls, and so on.

When considering the deployment models of the cloud computing, many resources can be connected via private or public networks. This technology simplifies infrastructure deployment by providing dynamic and scalable support for hosting services that are often difficult to predict over time. Organizations like the city administration can choose to deploy their cloud infrastructure in-house, working with partners who share a common vision, using the services of a public cloud service provider, or by combining many of these solutions. They are at least four deployment models [109] namely Public Cloud, Private Cloud, Community Cloud and the Hybrid Cloud.

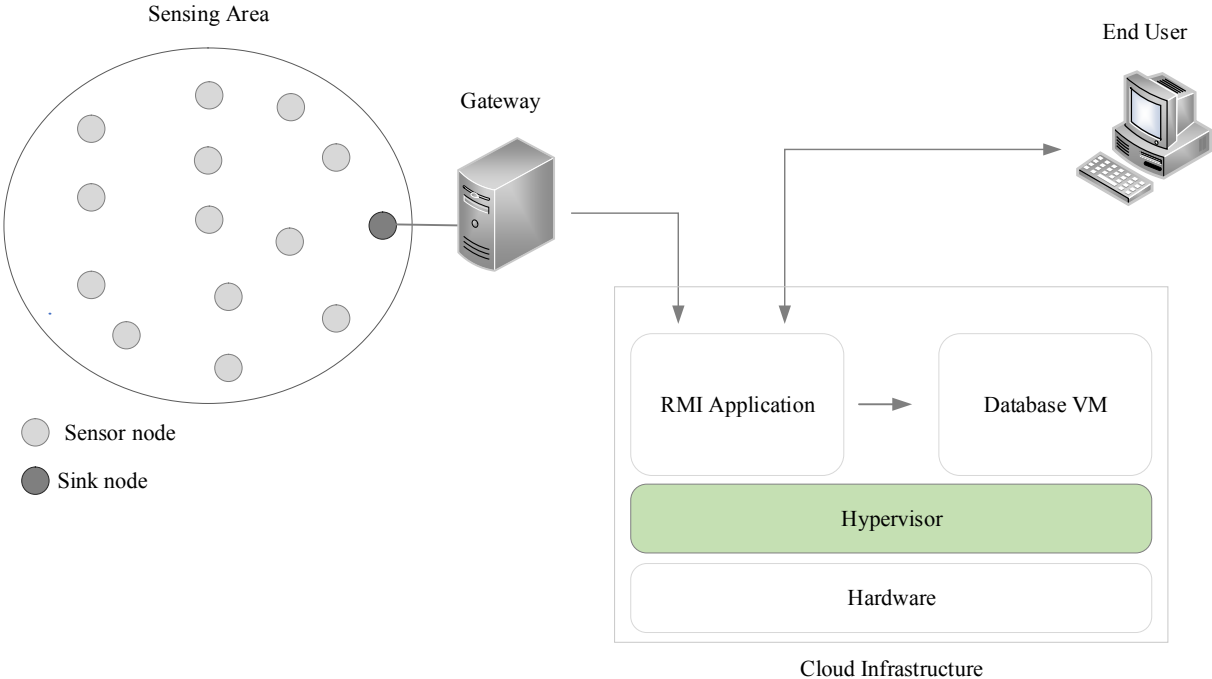


Figure 38 Hypervisor Type-1 based Cloud Infrastructure

The choice of deployment model of the cloud in our architecture can be based on the importance of WSN application that we want to implement. In this thesis we are interested in applications of the smart cities where the administration of a city is considered as the manager of smart city projects. Accordingly, the community-based cloud is a multi-tenant deployment model that is shared among multiple organizations (high education institutions, universities, researchers, weather stations, etc.) or companies (telecom carriers, internet providers, etc.) and is governed, managed, and secured by all participants or a service provider (Administration of the city). A community cloud is a hybrid form of private cloud built and operated specifically for a small and targeted group like the size of the city. These communities have similar requirements and bring together their human and financial resources to achieve their common goals. The common infrastructure is specifically designed to meet the requirements of a community; for example, government agencies, hospitals, or telecommunications companies with similar network, security, storage, computing, or automation constraints might find common interests in collectively deploying a community cloud.

This cloud infrastructure serves end-users to have the services with ease and transparency. The End-User layer can have multiple actors depending on the access levels they have in the distributed system. End-user may be a citizen of the city, an institution, a company, etc. Generally, End-User is an application or several applications facilitating end users to have access to services provided by the WSN-Cloud. It features graphical interfaces to view real-time data and information, retrieve old data, and monitor the WSN. These applications can be

installed on personal computers or other mobile devices or not integrating seamlessly into the system.

4.2.2. Logical Architecture

Logically, the proposed architecture considers that the GW, the CI and the EU are components of a same distributed system (Figure 39). According to the study carried out previously, Java RMI has been chosen as the best fit middleware between the gateway, cloud and the end users. In this architecture, each component must be able to provide services to others. So, we consider that each component must behave like a server in one case and a client in another. Therefore, each component must be able to create a single object and make it available to others in order to be requested to provide services when needed. That single object of each component must be distributed with RMI on a central server so that all other components can reach it. When any component needs a service from the other component, it must use the RMI reference of the distributed object and run an appropriate remote method for the requested service.

Logically, the Gateway hides the WSN behind itself. No other component can communicate with the WSN without going through the Gateway because it can convert the raw data from the WSN into a data format understandable by other components of the system. The fact that the WSN sink node is physically attached to the Gateway allows us to see the Sensing Area and the Gateway as a single component in this distributed system. As the Gateway hides the WSN behind itself, logically it becomes the source of the data. To achieve this combination of two physical layers into a logical layer, an application running on the Gateway is required. This application must be able to receive incoming data from the side of the WSN and get them out on the other side as objects toward the Cloud Infrastructure.

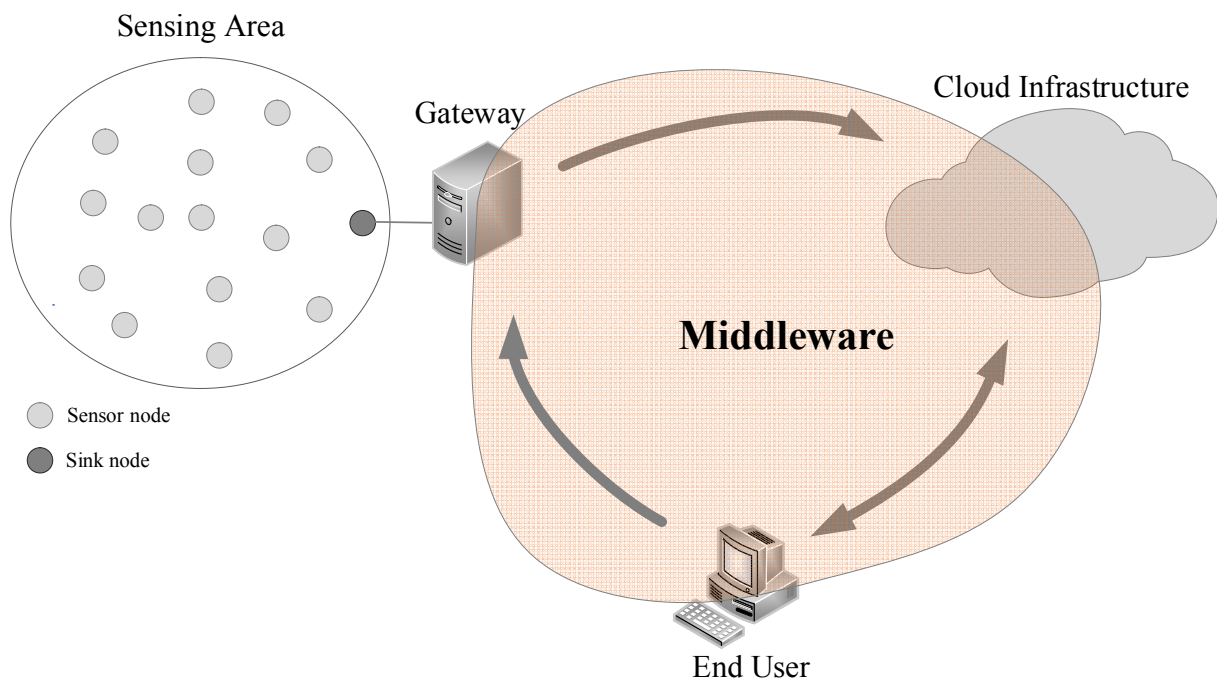


Figure 39 Logical WSN-Cloud Architecture

Previously, we discussed on the physical layer of the cloud infrastructure that it is an IaaS, but logically is an application that runs continuously in the cloud, which is connected locally to the database. It is obvious to consider it as a server because it is responsible for receiving fresh data from the WSN and responding to client requests. However, this application has its client face

because it can establish connections to clients to deliver information or notifications. On the other hand, under no circumstances should the application running on the cloud infrastructure need Gateway as a server, we cannot imagine what kind of service the Cloud Infrastructure might need from the Gateway.

Applications dedicated to end users are logically more clients than servers. Their roles include receiving and displaying data and information from the cloud as well as interacting with WSNs. This possibility that a user can communicate with the WSN makes this architecture more interesting in terms of services on demand. Depending on the access level of an end user, the end-user may send a request in the WSN through the Gateway to make some changes in the configuration or configuration of the network. For example, a researcher in any institution wanting access to data that was not yet collected despite the presence of sensors already deployed in the Sensing Area.

4.2.3. System Design and Algorithms

In a Java RMI environment, system components communicate with each other by invoking methods on objects distributed in the system. These objects are declared, instantiated, and distributed so that they are accessible throughout the system. The distributed objects must be hosted in RMI Registry and are accessible by their references. In this thesis, the design of the software to communicate the gateway, the cloud infrastructure and the end users is proposed in three steps:

- Definition of interfaces to be used by each component and methods for each interface.
- Implementation of the services to offer through the proposed architecture.
- The setup of the RMI registry where the distributed objects must be hosted.

4.2.3.1 Definition of Remote Interfaces

The first thing to do is to define the methods that must be exposed through RMI: those are declared through an interface. These interfaces will have to extend the interface *java.rmi.Remote*, this interface does not contain any method but simply indicates that the interface can be called remotely. The communication between the client and the server during the invocation of the remote method may fail for various reasons such as a server crash, a break in the link, etc. Hence, for each method called remotely will have to be able to propagate an exception of type *java.rmi.RemoteException*.

Among the methods to be declared, there are common methods for the operation of the system and the specific methods for each type of application. For example, if the application requires collecting the temperature in the different neighbourhoods of the city using the proposed architecture, an end user needs a specific method calculating the average of the daily temperature. But if the application deployed in the WSN is about to collect the vibration data of several bridges in the city, it makes no sense to average the vibration on several bridges because each bridge is unique but the neighbourhoods of the city shares the same atmosphere, the reason why it is possible to look for the temperature average.

In this architecture, each component can behave like a client in some cases and as a server in other cases, for this reason each component of the system there must be an interface with the name of the component and the necessary methods. For example, when the gateway receives data from the sink node, it must invoke a cloud object method to store it in the database. In this case, the gateway behaves as a client and the cloud as a server but in the event that the end user sends the instructions in the WSN (for example, changing the data collecting interval), he/she

must use the distributed gateway object on RMI registry to execute a necessary method. In this case, the gateway acts as server and the end user as a client.

4.2.3.1.1 The Gateway interface

As shown on the *Figure 40*, for the Gateway interface, three common methods have been declared:

- ***startCollectingData()***: When an end user invokes this method on the gateway object distributed with Java RMI, the wireless sensor network begins to collect the data, i.e. through the sink node, the application running on the gateway workstation sends a broadcast to all WSN’s sensor nodes to wake up and start collecting data.

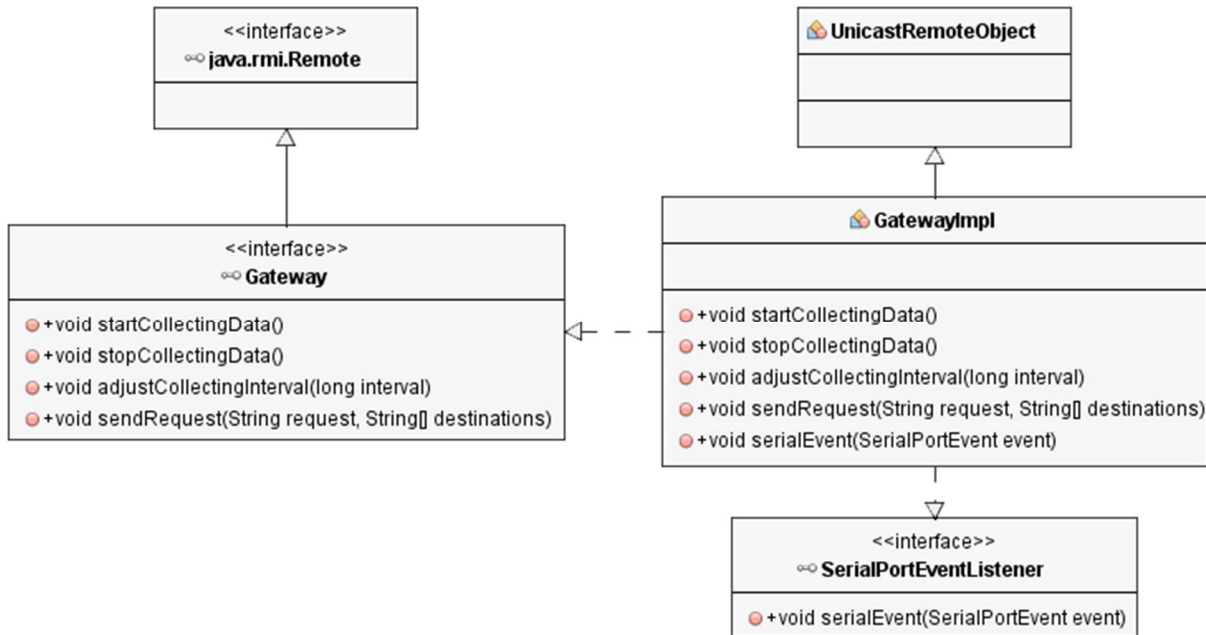


Figure 40 Diagram of the Gateway Interface Definition and Implementation

- ***stopCollectingData()***: Unlike the *startCollectingData()* method, this method puts all sensor nodes in sleeping mode whenever an end user invokes it on the distributed gateway object with Java RMI. These sensor nodes remain idle until the next invocation of the *startCollectingData()* method. It is possible to overload these two methods so that it can transmit the parameters as method arguments, these parameters can be related to the type of sensor nodes that must be slept or to be awakened or thresholds of battery energy below which a sensor node can be kept idle.
- ***adjustCollectingInterval(long)***: This method allows the end user to change the collection interval of the data. When invoked on the distributed gateway object with Java RMI, it sends a broadcast to all WSN sensor nodes to adjust their collection interval. This method sends an argument in milliseconds indicating the time it takes to get fresh data again.

4.2.3.1.2 The Cloud interface

The Cloud object distributed with Java RMI must have at least the following declared common methods (see the next following *Figure 41*):

- ***saveData(Datastore)***: When the data from the WSN arrives at the gateway, the application running on the gateway’s workstation is woken up to parse the received data and then creates a *Datastore* object that should contain only one type of data

(temperature, humidity, pressure, etc.). The gateway application then retrieves the *Cloud* object distributed with Java RMI in order to execute *saveData(Datastore data)* method and store the data passed as argument into the database.

- **registerEndUser(EndUser)**: When an end user launches his/her application, the *EndUser* object is registered with RMI registry so that it can be used by other system components. The same *EndUser* object distributed with Java RMI must be registered in the list of end users available in the *Cloud* object also distributed with Java RMI. The purpose of this recording will be explained later in this thesis.

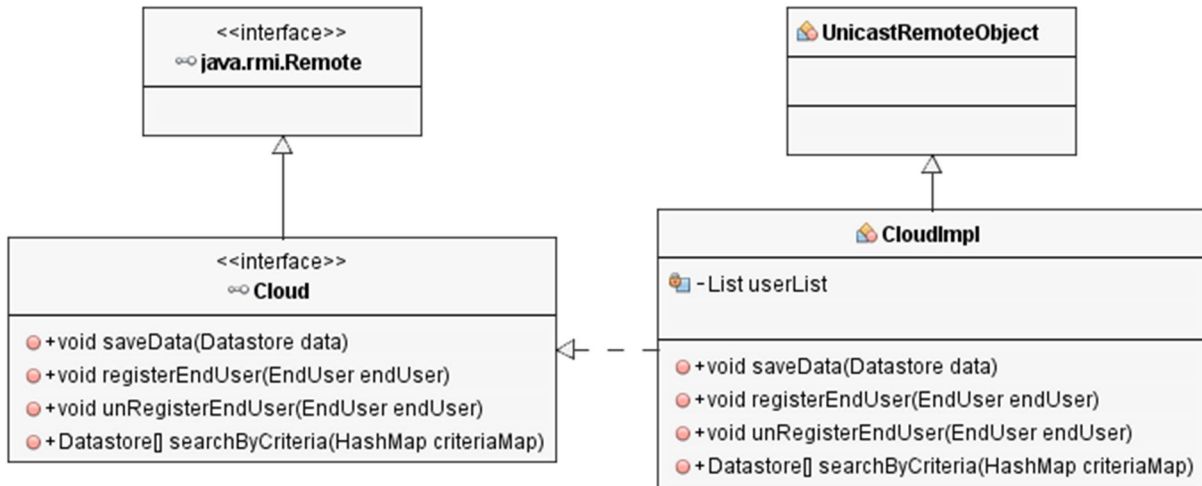


Figure 41 Diagram of the Cloud Interface Definition and Implementation

- **unRegisterEndUser(EndUser)**: This method is executed when an end user quits or closes his/her application. Before exiting the application, a closing event triggers this method so that the instance of the previous *EndUser* object registered is removed from the end users list of the *Cloud* object.
- **searchByCriteria(HashMap)**: This method is an example of how to retrieve data already stored in the cloud infrastructure’s database. When an end user needs to retrieve old data, he/she uses this method (or an alternative of this method with more arguments for searching) that returns an array of *Datastore* objects. The *criteriaMap* argument contains the map (key, value) of the search criteria (data type, date, location, the limit of items to be returned, etc.).

4.2.3.1.3 The EndUser interface

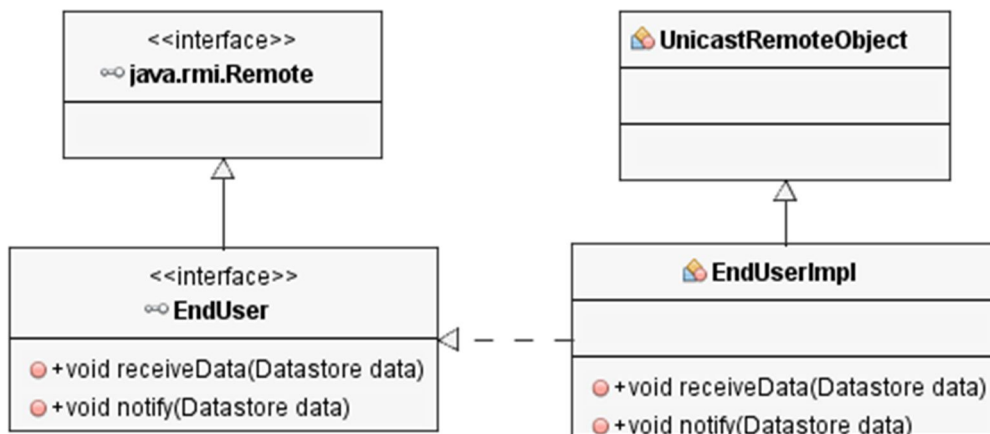


Figure 42 Diagram of the EndUser Interface Definition and Implementation

At least two methods must be declared in the EndUser interface shown in the figure above:

- **receiveData(Datastore)**: the sensed data arrived in real-time to the end user through this method. More explanations will be provided a little later in this document.
- **notify(Datastore)**: to be notified of an event observed by the WSN, an end user could choose the thresholds at which he will start to receive notifications, it is through this method that this process will happen.

4.2.3.2 Registration in the RMI name registry

The following operation is to prepare the RMI Registry where the distributed objects will be hosted. When an object is created and bound into the RMI Registry, a name is assigned to it. This name is provided to the registry as a URL consisting of the prefix *rmi://*, the server name (hostname or IP address followed by the port number for which the RMI server replies on), and the name associated with the object preceded by a slash. The server name can be supplied as a string constant or can be dynamically obtained using the *java.net.InetAddress* class for local use.

This name will be used in a URL by the client application to obtain a reference to the remote object. The registration is performed using the *rebind()* method of the *java.rmi.Naming* class. It expects as parameter the URL of the name of the object and the object itself.

On the server, the RMI name register must run before you can register an object or obtain a reference. This registry can be started as an application provided in the JDK (*rmiregistry*) or launched dynamically in the class that registers the object. This launch should only take place once. The registry is run by the *createRegistry()* method of the *java.rmi.registry.LocateRegistry* class. This method expects a port number on which the server will respond.

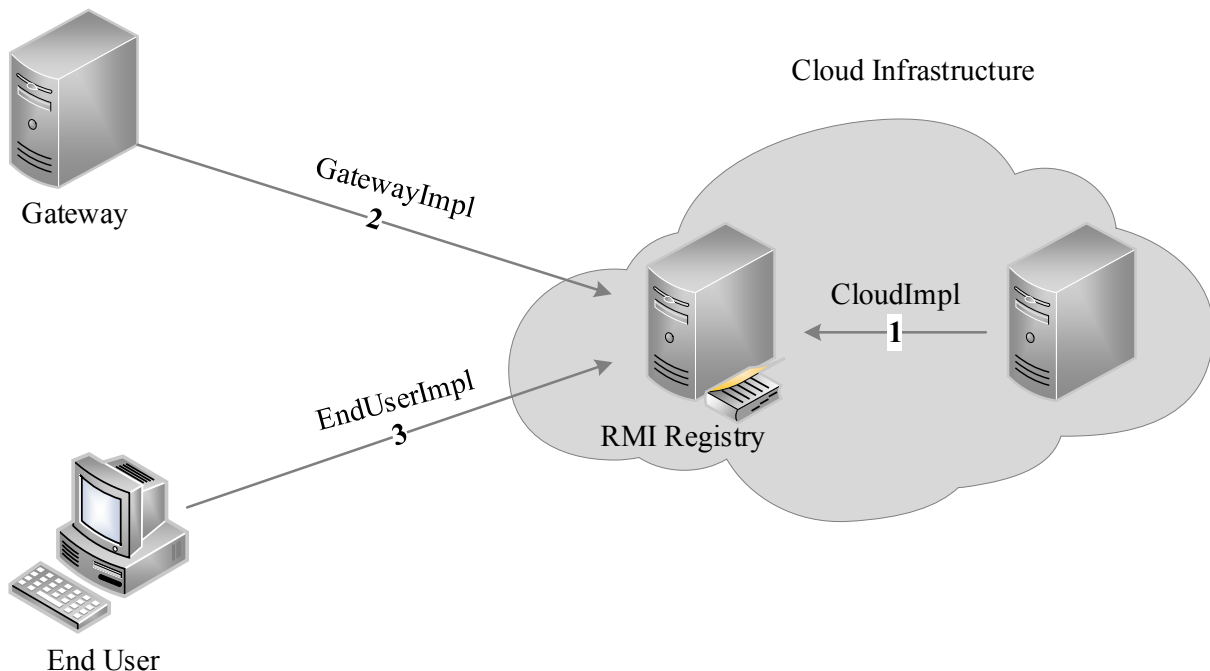


Figure 43 Registration sequence of remote objects to the RMI registry

To get a reference to remote objects from their names, the static method *lookup()* of the *java.rmi.Naming* class is used. This method expects a URL as a parameter indicating the name that references the remote object. This URL is composed of several elements: the prefix *rmi://*,

the name of the server (hostname) and the name of the object as it was registered in the register preceded by a slash.

The *lookup()* method will find the object in the server's registry and return a stub object. The returned object is from the Remote class (this class is the parent class of all remote objects). If the name provided in the URL is not referenced in the registry, the method throws the *java.rmi.NotBoundException* exception.

For the proposed architecture, the RMI registry must be situated on the considered core component, the cloud infrastructure. When the cloud-running application is launched, the registry RMI is started at the same time, closing at the same time too. Once started, other system components can launch their applications by registering their objects in the RMI registry. The *Figure 43* shows the registration sequencing operations on the RMI Registry.

4.2.3.3 Implementing Intended Features and Services

The interface implementation creates classes corresponding to the remote objects. These classes must therefore implement the interfaces previously defined and contain the necessary code. As shown on the previous figures (*Figure 40*, *Figure 41* and *Figure 42*), those classes must inherit from the class *java.rmi.server.UnicastRemoteObject*, which contains the various elementary processes for a remote object whose call by the client's stub is unique. The stub can only get one reference on a remote object inheriting from the *java.rmi.server.UnicastRemoteObject* class. It can be assumed that a future version of RMI will be able to make *MultiCast*, allowing RMI to choose from among several identical remote objects the reference to provide to the client.

As stated in the interface, all remote methods, but also the class constructor, must indicate that they can throw the *java.rmi.RemoteException* exception. Thus, even if the constructor does not contain any code, it must be redefined to inhibit the generation of the default constructor that does not throw this exception.

Be aware that for a complete implementation we could have the data displayed in a graphical user interface with controls to manipulate the data. The implementation examples of this proposed architecture are detailed later in this document.

1.2.3.1.1. Real-time Monitoring and Notification Services

Real-time monitoring is a challenge for most applications communicating over the network. In this architecture, real-time monitoring must be performed through the cloud. Each time the data is sensed by the WSN, it is sent to the gateway through the sink node. At this point, the gateway creates a *Datastore* object, fills that data, and then retrieves the stub of the cloud object distributed with Java RMI to invoke the remote method by passing the *Datastore* object as an argument to this method.

Notifications are alerts that display on end-user computers to alert them to a new activity. The cloud object distributed with Java RMI to warn them for example when the temperature increases or the pressure drops in the sensing area launches them. In fact, end users must setup the lower or upper thresholds with which notifications must be sent. It is obvious that each user records their own alarm levels in the cloud to be notified whenever declared thresholds are reached or exceeded. As a result, whenever the cloud-running application receives data from the WSN, it compares them against the thresholds stored in the database for each end user. If the collected data matches at least one of the saved thresholds, the cloud-running application sends the notification by calling the *notify(Datastore)* method, which must be added to the *EndUser* object distributed with Java RMI. Indeed, the real-time monitoring occurs in three important steps:

- **WSN towards Gateway:** The gateway runs a simple JavaSE application connecting the WSN and the cloud using remote invocation concept. This application uses a Java extension library called RXTXcomm allowing us to access the serial ports of the computer where the sink node of the WSN is connected. When launching the gateway application, a dialog frame prompts to choose the name or number of the serial port

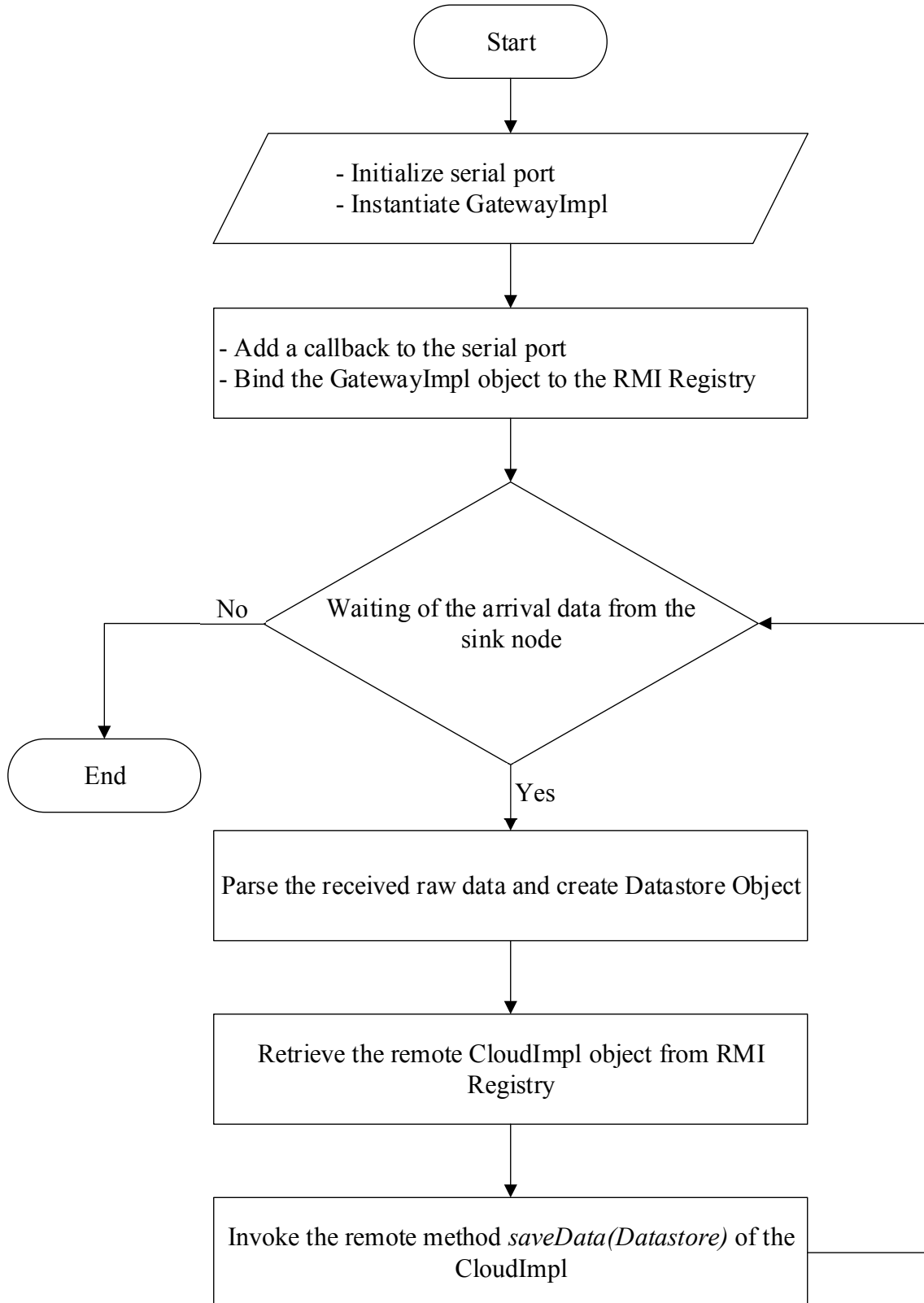


Figure 44 The flowchart of the Gateway's Algorithm

(COM 4 for example) on which the gateway is plugged in, then the serial port is initialized in the application and an event call-back is added to the serial port allowing the application to be notified whenever the data are available at that port. When the frame sent by any sensor node arrives at the sink node, it is routed through the serial port of the gateway and the listener immediately reports the arrival of data flow to that serial port. At the moment, the gateway application reads the received raw data through the input of the serial port instance and it maps them to a new *Datastore* object. Then the base station retrieves the instance of the object distributed by the cloud and executes the remote *saveData(Datastore)* method for sending the *Datastore* object as parameter. The base station waits again until the arrival of another frame and so on.

- **Gateway towards the Cloud Infrastructure:** The cloud infrastructure is the central element of this system. Therefore, the cloud connects all components in the distributed

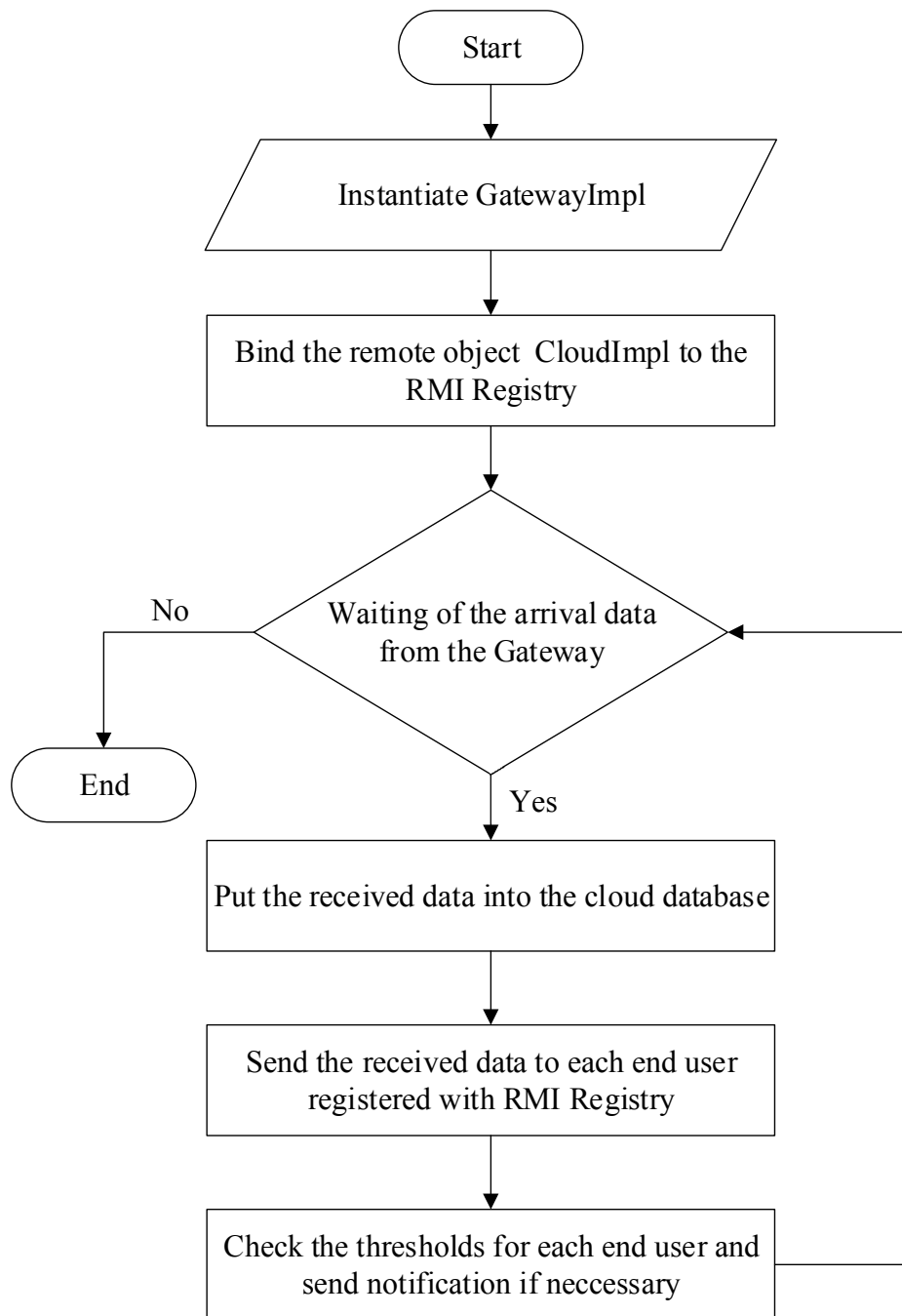


Figure 45 The flowchart of the Cloud Infrastructure's Algorithm

system; but an end user can access the gateway as well without using the cloud. In fact, the cloud has a Java interface called “*Cloud*” extending the RMI’s “*Remote*” interface. This interface contains methods for receiving data from the base station and the utility methods of registration of end users, disconnection of users, querying of the database, etc. The primary element of cloud application is the class that implement the methods declared in the previous interface (i.e. this class extends RMI’s *UnicastRemoteObject* object and implement the previously mentioned interface), this class also called “*CloudImpl*” contains as global variable a list of the end users, this list registers at any time all available users and ready to receive data, i.e. when an end user launches her/his application, the reference of his/her distributed object is added immediately in that list and removed when closing the end user application. After the implementation of all methods, the object of *CloudImpl* is instantiated and bound on the local RMI registry in the cloud so that this object is available throughout the distributed system via his name “cloud”. When data arrives in the cloud as parameter of *receiveData* method, the cloud application executes firstly a method that receives the data for each object (the same objects containing in the collection) distributed by the user, then cloud application also makes a copy of the data to be stored into the local database (cloud database).

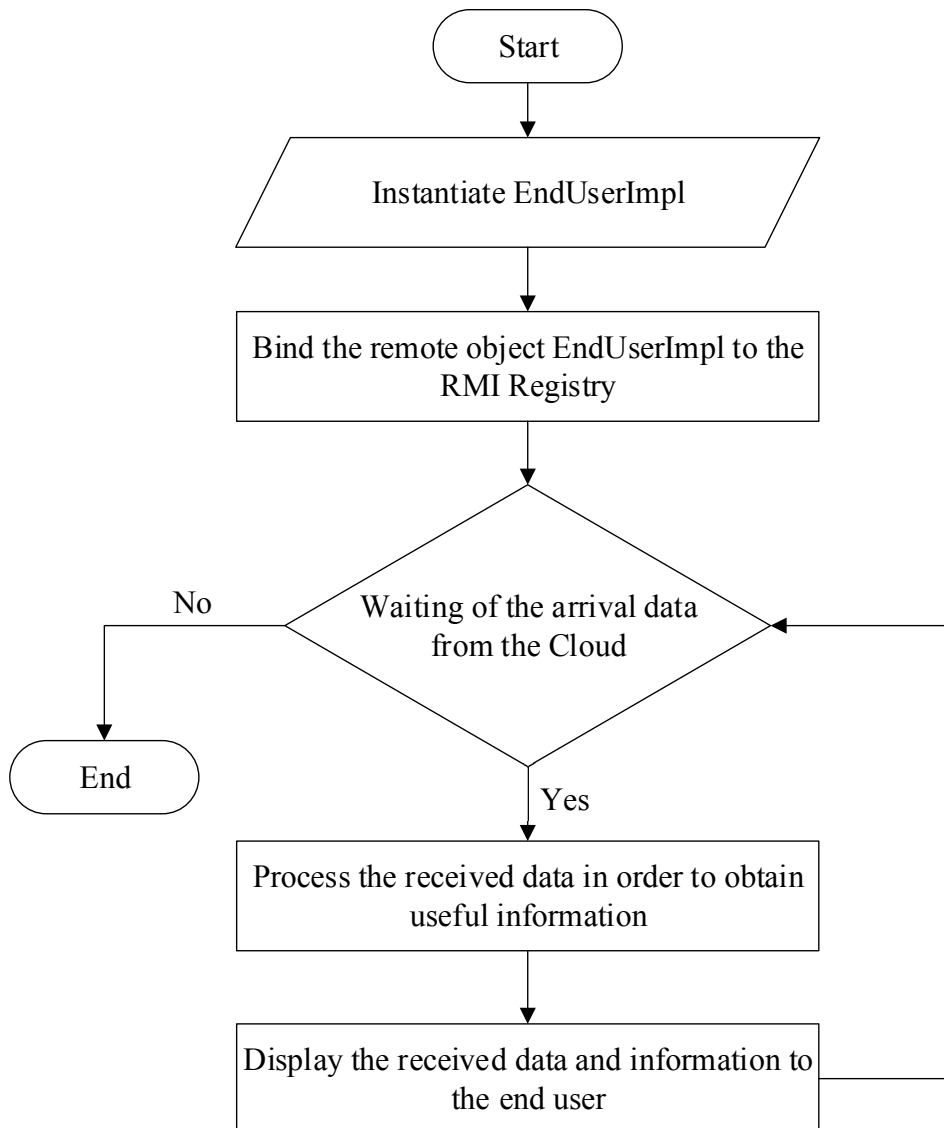


Figure 46 The flowchart of the End User's Algorithm

- **Cloud Infrastructure towards End-User:** as shown in the *Figure 46*, this application receives the data from the cloud with the concept of method invocation on distributed objects with Java RMI. By analogy with CloudImpl, the user application contains a class called EndUserImpl with some methods whose main is to receive data from the cloud. Unlike to CloudImpl, EndUserImpl is not bound directly into the RMI registry; it is sent as a parameter of the method that registers end users in the collection of CloudImpl. Indeed, at the launch of the user application, the object distributed by the cloud is retrieved and then the current user is registered in by passing as a parameter the instance of the EndUserImpl object. On arrival data, graphical user interfaces are updated to display the new row of data. At the closing of the user application, the object distributed by the user is removed from the collection in the cloud.

4.2.3.3.1 Data on-Demand for End-Users

The data on-demand in the proposed architecture refers to an information delivery and distribution model in which data from the wireless sensor network is made available to end users only in case of need. For example, when there is a lack of specific data, an end user can send the requests to the WSNs to collect the desired data. Indeed, there may be several scenarios in which an end user could use the specific data on demand. In this section we are going to deal with the case of sensors activation on demand.

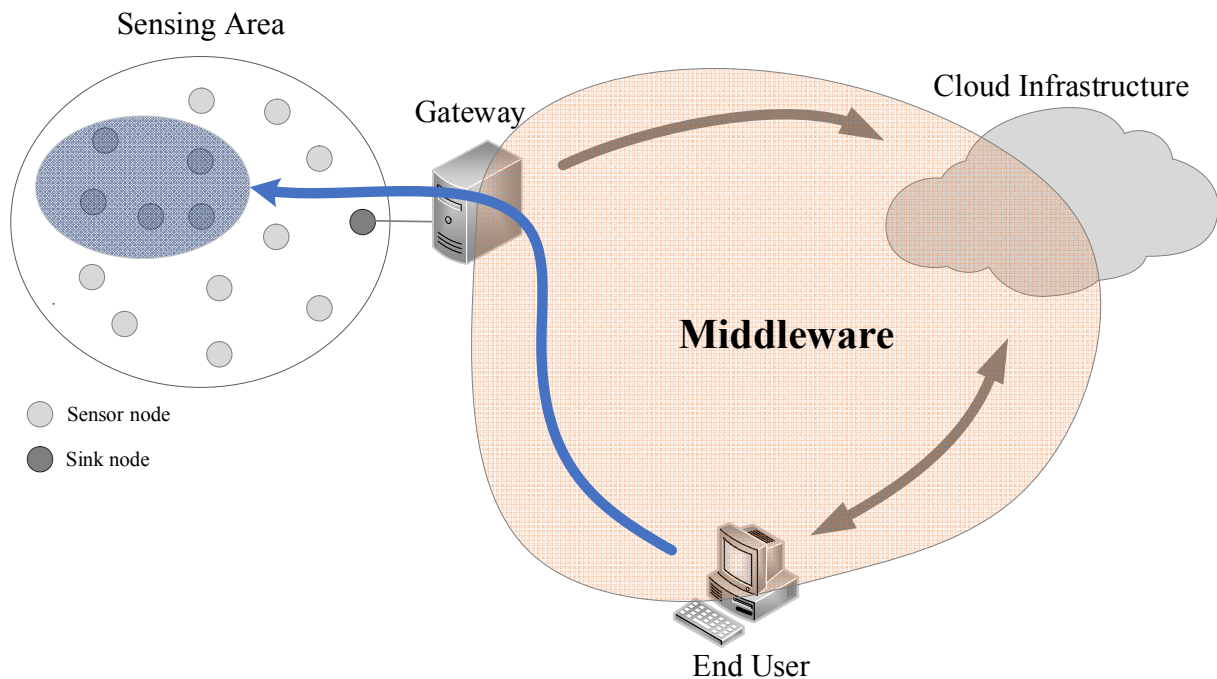


Figure 47 Logical architecture showing data on-demand service

In this context of sensors or data on demand, an end user would be able to target any part in the Sensing Area to request data that is not yet available, either because sensor nodes are disabled (sleeping mode) and do not transmit their information at the base station, either the sensor nodes are activated but do not collect all types of data they are supposed to provide. In each case, a user could send a request through the Gateway to the targeted nodes in the wireless sensor network to wake them up and start collecting the need data. In the case of the smart city, the target part of the wireless sensor network may be a neighbourhood, a road or a building ... whose list of the addresses of the sensor nodes that are deployed therein is mapped with a corresponding name of the site.

Let's imagine a case of using this service in an example of the smart city. Suppose that any city has decided to be smart by deploying different WSN platforms around its sites. These platforms would be used, for example, to collect ambient temperature, air humidity, air pressure, concentration of dust in the air, air pollution, sound detection, level of water in the sewer, etc. Some of these sensors are useful in everyday life such as the temperature to inform the citizens the weather it is at every moment of the day. However, some sensors do not seem to be useful at a certain time of the year, then they have to stay in hibernation in order to preserve their energy, this is the case for example sonar sensors fixed in the sewers to monitor the level of rainwater that flows there.

This service allows an end user to activate the sensors that were in hibernation so that they can start sensing the requested data. In the previous example of sewage water, let's imagine a scenario where it is raining unexpectedly, a citizens could take in charge the responsibility of asking the nearest sensors located in its neighbourhood to wake up in order to collect information on the quantity of the sewer water[110], when for example, the thresholds set by the city authorities are reached, it can react through this service on demand. Below on the *Figure 48* is shown an example of level sensing of sewage water by using sonar sensor.

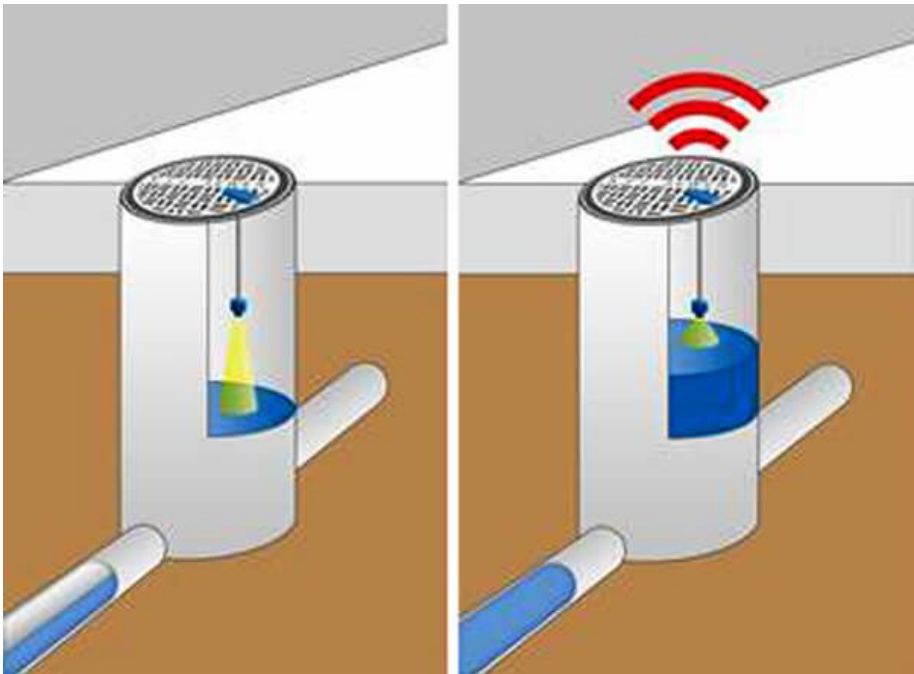


Figure 48 The concept of level sensing of the sewage water

Another similar and very useful example is to allow an end user to report the accumulation of garbage in the city where the risk of unpleasant odours is high. This garbage can be gathered around from many sources such as alluvium caused by rain, the corpses of putrefying animals, broken pipes of toilet sewers, etc. When we look at most of these sources of garbage are not permanent, garbage is following an event whose probability of realization is minimal. Therefore, it would be smarter to keep all sensors dedicated to detect odours in hibernation and only activated when a user requests to reactive them on demand.

4.3. Conclusion

Since wireless sensor networks are limited in computing power, battery life, and communication speed, cloud computing typically provides the storage capacity and computational power needed for long-term observation, analysis, and use in a variety of environments and projects, with the underlying infrastructure remaining the same. The

integration of WSN and cloud computing will benefit organizations and the research community. Businesses will benefit from the use of cloud storage, computing power, and WSN generation data recovery.

The concept presented in this chapter gives us a general idea of how users can benefit from wireless sensor networks across the cloud. The use of middleware allows users to not worry about the geographic location, configuration, and scale-up of wireless sensor network infrastructures as well as the cloud. The proposal of a logical architecture made it possible to consider an entire wireless sensor network as an object communicating on the network with a Gateway that represents it and through the Gateway, a user would be able to interact with the captors by demanding the information he/she needs. The work that followed this integration proposal (the following chapters) resulted in discussing the security and feasibility aspects of smart cities.

Chapter -5-

Data Privacy of the Integrated WSN-Cloud

Users on the external networks can exploit the collected data from the wireless sensor network (WSN) in real-time but they also need to be stored for later use, however it is not a simple matter to find enough processing power and storage for this multitude of data. The Cloud infrastructure provides several benefits such as mass storage, the demand for self-service access to the network wide resource consolidation, measurement service, mass scalability, consistency, virtualization, software low-cost, distribution, service orientation and advanced security [79]. All these qualities are essential to fill the gaps found in the WSN.

Consider a simple example of weather, a city where each neighbourhood has its own weather station, data on rainfall, temperature, humidity, wind, etc., will be stored in the cloud's database to serve statistics for example. Suppose each weather station has a multitude of redundant sensors, in fact we will not use a single temperature sensor for a single station but several and make an arithmetic average to be sent to the cloud. In this example, the sent values are plaintexts, an attacker can intrude into the network and tries to visualize the information that transit, but the big concern is the falsification or unauthorized modification that can occur in the cloud. An effective way of ensuring the confidentiality of data from the sensor network to the system administrator via the cloud is required. Such a system must be composed of four major actors, namely the sensor network, the cloud infrastructure, an administration subsystem and a web service for end users (citizens).

Imagine we decide to encrypt the data from the sensor network with the symmetric algorithms such as AES. Whenever the administration wants to perform the calculations (statistics) on the stored data, it must either send a secret key to the cloud for decryption (a potentially dangerous solution because the sending of the key on the network is likely to be intercepted but the administration can benefit the computing power of the cloud). The second option is about to download the encrypted data and performing the statistics locally (solution without risk but the administration machine will have a huge data capacity to process, which requires a powerful machine). In both cases, we do not have the best solution to remedy this problem.

Imagine a solution that does not require a decryption key to perform these operations (statistics, forecasting, etc.) on encrypted data. With that solution, the administration will not have to take the risk of sending that decryption key over the network and in addition, he/she will benefit the computing power of the cloud. The administrator's workstation only has to decrypt the results of the statistics using its key that it keeps locally; such solution can be the use of *Homomorphic Encryption* algorithm.

A homomorphic encryption scheme makes possible to perform the arithmetic operations on the encrypted data without decrypting them. Homomorphic encryption also reduces the processing time of initially encrypted data. Homomorphic encryption in the cloud computing ensures data confidentiality and minimizes the risk of attacks [111]. By applying a homomorphic encryption scheme in a system, it can avoid the risk that can occur when exchanging the secret keys. In short, this chapter focuses on the use of homomorphic encryption algorithm applied to the distributed system combining the sensor network, cloud infrastructure and end users.

The homomorphic encryption implemented in this chapter has been the subject of a theoretical study on the different symmetric and asymmetric algorithms. The choice of the implemented algorithm was based on the criteria about computational difficulties such as exponentiations, which are not adequate in the WSNs due to the congestion of the computational resources. The implementation was carried out with the Domingo-Ferrer [112] encryption scheme on the sensor nodes embedding 16-bit microprocessors, which are moderately sufficient for cryptography calculations. This encryption algorithm requires that the message must be split into several fragments before proceeding to encryption, so several experiments has been performed with the variable number of the fragments (2, 3 and 4) in order to understand the impact of this message splitting on the state of the battery of each sensor node. The results obtained showed that after sending 10,000 messages the battery levels decreased respectively by 6%, 7% and 8%, which shows that the more the message is fragmented the more the energy of the WSN is slightly diminished but the level of data security (confidentiality) is increasing.

This chapter reviews the wireless sensor network, the importance of integrating it with Cloud Computing as well as the security using homomorphic encryption. Then, there is a short summary about historic of the homomorphic encryption and related works to this chapter's topic. The next section focuses on the importance of homomorphic encryption on aggregated data in the WSNs. The next section is the security modelling required for aggregated data in the WSN. The next section is the implementation of the Domingo-Ferrer algorithm in the WSNs. It is in the sixth section where the results are presented and finally the discussion and c in the last section.

5.1. Homomorphic Encryption and Data Privacy

5.1.1. Brief History of Homomorphic Encryption

Asymmetric cryptography appeared in 1978 when Rivest, Shamir and Adleman published their RSA cryptosystem [113]. This cryptosystem is revolutionary in the sense that it allows everyone to send encrypted messages to a person, without anyone other than the recipient being able to decipher. This cryptosystem gives rise to a new era: that of public key encryption. Many cryptographers study this system, and it is soon realized that it has two peculiarities:

- It is homomorphic multiplicatively: $c_1 = Enc(m_1)$ and $c_2 = Enc(m_2) \Rightarrow c_1 \times c_2 = Enc(m_1 \times m_2)$. In other words, without knowing m_1 nor m_2 but only their ciphertexts, it is possible to calculate an encryption of $m_1 \times m_2$.
- Given a message $m \in P$ and an encrypted $c \in C$, it is possible to determine whether $c = Enc(m)$ or not.

The first property does not seem to affect system security, but generates many queries. To such an extent that in the same year Rivest, Adleman and Dertouzos [114] conjecture the existence of a completely homomorphic system of ciphering under the name of privacy homomorphism, and propose candidate schemes. However, these were all broken afterwards.

The second feature of the much less desirable RSA encryption system will be described in 1982 as non-indistinguishable against selected light attacks by Goldwasser and Micali [115]. Indeed, the fact that the encryption process is deterministic makes it possible to know a posteriori whether a cipher c encrypts a given message m . Since Goldwasser and Micali introduced the concept of probabilistic encryption, any encryption system must integrate randomness into the encryption process to be considered safe.

Three years later, El Gamal [116] published a system of encryption which Diffie-Hellman key exchange ideas [117], whose security hypothesis is different from previous cryptosystems.

Indeed, the previous cryptosystems were based on the RSA problem and the quadratic residuality respectively that of El Gamal is based on the Diffie-Hellman decisional hypothesis, which reduces to the discrete logarithm. On the other hand, as RSA, this cryptosystem is multiplicatively homomorphic.

In 1985, Miller [118] and Koblitz [119] independently suggest the use of elliptic curves in cryptography. Cryptography on elliptic curves has two major advantages over other cryptosystems: the size of the keys is much less (160 bits against 1024 for equivalent security), it evolves much more slowly (224 bits against 2048 for equivalent security). In the near future, this kind of cryptosystem will be more efficient than the asymmetric algorithms currently used. We will discuss this with the cryptosystem of Boneh, Goh, and Nissim [120].

In 1993, Fellows and Koblitz published "Polly Cracker" [121]: the first homomorphic encryption system capable of evaluating arbitrary functions on encrypted data. However, this cryptosystem is not "completely" homomorphic in the sense that the size of the cryptograms grows exponentially in the depth of the circuit to be evaluated.

In 1994, Benaloh [122] published a cryptosystem whose security is limited to the problem of higher order residuality, the peculiarity of which is the possibility of choosing the size of the blocks to be encrypted. However, this cryptosystem in its original version presented a defect on the choice of parameters as observed and corrected Fousse, Lafoucarde and Alnuaimi [123]. Certain parameters had to be chosen precisely so that the decryption is correct with a probability equal to 1. However, it will be noted that this is the first cryptosystem additively homomorphous.

Then in 1997, Naccache and Stern [124] published a cryptosystem based on the problem of the backpack. It allows like Goldwasser-Micali to perform XOR. The following year, Naccache and Stern propose another cryptosystem [125] based on the problem of higher order residuality this time. It happens that this cryptosystem generalizes that of Benaloh.

In the same year, Okamoto and Uchiyama published their cryptosystem [126]. They manage to find a new one-way function equivalent to factorization. However, the semantic security of the cryptosystem is assured only under the hypothesis of the subgroup of order p . However, their ideas will be resumed and improved in the years that follow. This cryptosystem is additively homomorphic.

In 1999, Coron, Naccache and Paillier [127] published an improved version of the Okamoto-Uchiyama cryptosystem, where decryption is optimized. In the same year, the same Paillier publishes a cryptosystem which takes up certain ideas of Okamoto-Uchiyama [128]. It also gives two other versions of its cryptosystem to speed up decryption. However, the second version is no longer IND-CPA, and modifies the homomorphic properties. On the other hand, Catalano, Gennaro, Howgrave-Graham, and Nguyen [129] have suggested using a low-binary exponent to accelerate the encryption process and a very special shape generator to replace a modular exponentiation by multiplication. However, the security of the system is close to the RSA problem.

Finally, in 1999, Sander, Young and Yung modify the cryptosystem of Goldwasser and Micali to replace the homomorphic XOR with a homomorphic AND (which corresponds to a modulo 2 multiplication) [130]. It will be noted that the technique used to modify this scheme is generic, and can therefore be used to transform other cryptosystems.

In 2001, Choi, Choi, and Won [131] modified the Okamoto-Uchiyama cryptosystem by selecting a generator of a particular shape, in order to speed up the decryption process. However, Sakurai and Takagi [132] emphasize that it is not guaranteed that the encryption function is not easy to reverse with a generator of this form. In parallel, Damgård and Jurik

publish a generalization of the cryptosystem of Paillier [133], as well as two modified versions, the length of which is flexible.

Motivated by the use of elliptic curves in cryptography, Paillier published a version of its cryptosystem in 2000 [134]. Galbraith shows in 2002 [135] that this version has a security flaw: the calculation of the discrete logarithm on the curves used is easy. It also suggests a safe version of Paillier on elliptic curves.

In 2005, Schmidt-Samoa and Takagi published a generalization of Paillier and Okamoto-Uchiyama [136]. At the same time, Boneh, Goh and Nissim publish a cryptosystem [120] on elliptic curves whose homomorphic properties are interesting. For the first time, a cryptosystem allows for both additions and multiplications. More precisely, their cryptosystem makes it possible to carry out an arbitrary number of additions, a multiplication, and then an arbitrary number of additions. The limited number of multiplications results from the fact that this multiplication is realized by means of a Weil coupling, which changes the underlying group.

Craig Gentry's thesis [137], which was the first completely homomorphic cryptosystem, in a few words, Gentry's scheme encrypts messages by adding noise. Homomorphic operations are performed also affecting this noise. When the noise becomes too great, Gentry applies a so-called "bootstrap" procedure, which reduces this noise to an acceptable level. Concretely, this procedure consists in evaluating the decryption of homomorphic function. The idea of Gentry is to start from a so-called "*somewhat homomorphic encryption scheme*" which can evaluate additions and multiplications as long as the noise is not too big, and apply the bootstrap procedure.

5.1.2. Homomorphic Encryption and Wireless Sensor Networks

A faster variant of Paillier's additive homomorphic encryption protocol was derived in 2011 focusing on the security of aggregated data in the WSNs [138]. The idea was to speed up exponentiation in the decryption process. Performance evaluation of the obtained results indicates that the protocol they proposed can accelerate about 49% in encryption and 50% in decryption.

An optimized implementation of the elliptic curve EL Gamal based on additive homomorphic encryption has been presented in order to offer a fast multiplication point by creating a small code beneficial to the memory of the sensor nodes [139]. The results they obtained show that their implementation is 44% faster compared to the previous best results.

Another homomorphic encryption scheme based on elliptic curves has been proposed to avoid eavesdropping to wireless channels and to ensure energy savings in cluster-based WSNs [140]. Based on the obtained experimental results, the authors promise that their method improve network performance compared to other methods in terms of energy consumption, memory requirement, network overhead and network lifetime.

A homomorphic encryption scheme with a low computation and communication overhead has been proposed to secure compressive data gathering in the WSNs [141]. The authors had the main goal of protecting against traffic analysis and flow tracing in WSNs. By using homomorphic encryption, data can be aggregated to reduce network traffic, but homomorphic encryption functions also increase the size of packets to be sent over the network while increasing power consumption because the consumed power is directly proportional to the amount of transmitted data.

A study was conducted in 2012 [142] to investigate the effect of increasing packet size for Domingo-Ferrer homomorphic encryption scheme in comparison to a symmetric cryptosystem.

The results approved that symmetric encryption outperforms homomorphic encryption for small WSNs, but as the network grows the homomorphic encryption outperforms symmetric encryption.

Another implementation of the Domingo-Ferrer's homomorphic cryptosystem has been simulated on the Mica2 nodes in OMNet ++ [143]. During the implementation, the authors considered three different scenarios (different key sizes and fixed message size, different message sizes and fixed key size, different message splits and fixed key size). The obtained results allowed them to conclude that splitting the message into several small messages makes it possible to increase the level of security but by penalizing the long-term network, which must be split at least the message.

Recently, a secure data collection scheme based on compression sensing has been proposed firstly in order to improve the data privacy by the use of the asymmetric semi-homomorphic encryption and secondly to reduce the computational cost by using the sparse compression matrix [144]. The asymmetric encryption mechanism reduces the difficulties of encryption secret keys management and distribution, while homomorphic encryption allows the aggregation of encrypted data in the network as well as improving security and load balancing on the network. The sparse compressive matrix reduces the computation and the communication cost by compensating the increasing cost caused by homomorphic encryption. The results that the authors obtained are satisfactory or even better compared with the most related research works.

Almost the same authors [145] as in previous research work have proposed what they called MODA (Multi-functiOnal secure Data Aggregation), which encodes raw data into well-defined vectors to provide value preservation, order preservation, and context preservation and thus by building blocks for multifunctional aggregation. The main purpose of this project was to compute efficiently in distributed mode even without worrying about security issues. To do this, they also used homomorphic encryption to enable in-ciphertext aggregation and end-to-end security.

All these related works to this subject concerns only the security inside the WSN, that means once the encrypted data arrived at the base station are decrypted to be processed or visualized but this chapter want this data to continue to the cloud by being encrypted for storage. This chapter shows how it is possible for an end user to perform through the cloud the arithmetic operations on this kind of data (homomorphically-encrypted data) without being decrypted in order to increase the privacy level.

5.1.3. Homomorphic Encryption applied to the Cloud Computing

Cloud Computing permits companies to increase capacity quickly without the need for new infrastructure investment and similarly companies can decrease capacity quickly and efficiently. Cloud Computing is principally designed and promoted to be data centre centric and efficient interaction with the outside world is an area where improved solutions are being sought [146]. In fact, Cloud Computing offers several advantages such as mass storage, the demand for self-service access to the wide network, resource consolidation, the measurement service, massive scalability, consistency, virtualization, low cost software, distribution, orientation of service and advanced security. All these qualities are essential to fill the gaps found in the WSN [147].

As we know, the WSN is to collect data in the real world, but the question is what to do with this data when the organisations no longer need it. There are many reasons to store the sensed data for historical, future research, and re-analysis at some future point in time. There is a

possible linkage between WSN and Cloud Computing and the eventual shift of data into the cloud and over time into the public domain [146].

By definition [111], we have an homomorphic encryption if from $Enc(m_1)$ and $Enc(m_2)$ it is possible to compute $Enc(f(m_1, m_2))$, where f can be $+$, \times or \oplus and without using the secret key. According to the operations that allow to access on raw data, we distinguish three types of homomorphic encryption:

- A homomorphic encryption is called *additive*, if: $Enc(m_1 + m_2) = Enc(m_1) \cdot Enc(m_2)$
e.g.: Pailler's cryptosystem

$$Enc(m_1) = g^{m_1} \cdot r_1^N \text{ mod } N^2 = c_1$$

$$Enc(m_2) = g^{m_2} \cdot r_2^N \text{ mod } N^2 = c_2$$

$$\Rightarrow Enc(m_1 + m_2) = g^{m_1 + m_2} \cdot (r_1 \cdot r_2)^N \text{ mod } N^2$$

$$= Enc(m_1) \cdot Enc(m_2) = c_1 \cdot c_2$$
- A homomorphic encryption is called *multiplicative*, if $Enc(m_1 \cdot m_2) = Enc(m_1) \cdot Enc(m_2)$
e.g.: RSA cryptosystem

$$Enc(m_1) = m_1^e \text{ mod } n = c_1$$

$$Enc(m_2) = m_2^e \text{ mod } n = c_2$$

$$\Rightarrow Enc(m_1 \cdot m_2) = (m_1 \cdot m_2)^e \text{ mod } n = c_1 \cdot c_2$$
- A homomorphic encryption is called *fully*, if it is at time additively and multiplicatively homomorphic.

e.g.: In 2010, a completely homomorphic encryption scheme [148] named DGHV was presented which is an application of Gentry encryption [137] on integers and whose security is based on the problem of the approximate common divisor.

Keys generations: r , p and q . Where $r \sim 2^n$, $p \sim 2^{n^2}$, $q \sim 2^{n^5}$, p and q are the prime numbers.

$$Enc(m) = pq + 2r + m = c$$

$$Dec(c) = (pq + 2r + m \text{ mod } p) \text{ mod } 2 = m$$

For two messages m_1 and m_2 , let c_1 and c_2 be their ciphertexts respectively:

$$\blacksquare c_1 + c_2 = (q_1 + q_2)p + 2(r_1 + r_2) + m_1 + m_2$$

So if $2(r_1 + r_2) + m_1 + m_2 \ll p$, then $((c_1 + c_2) \text{ mod } p) \text{ mod } 2 = [2(r_1 + r_2) + m_1 + m_2] \text{ mod } 2 = m_1 + m_2$. Thus, DGHV realizes the property of additive homomorphic encryption.

$$\blacksquare c_1 \times c_2 = [q_1 q_2 p + (2r_1 + m_1) + (2r_2 + m_2)] p + 2(2r_1 r_2 + r_1 m_1 + r_2 m_1) + m_1 m_2$$

So if $2(2r_1 r_2 + r_1 m_1 + r_2 m_1) + m_1 m_2 \ll p$, then $((c_1 \times c_2) \text{ mod } p) \text{ mod } 2 = [2(2r_1 r_2 + r_1 m_1 + r_2 m_1) + m_1 m_2] \text{ mod } 2 = m_1 m_2$. As a result, DGHV also performs the multiplicative homomorphic encryption.

5.1.4. Homomorphic Encryption on Aggregated data

A sensor network generally consists of a large number of sensor nodes strongly deployed in a sensing region and one or more base stations located within the collection zone. The base stations send requests or commands to the sensor nodes in the sensing zone while the sensor nodes collaborate to accomplish the sensing task and send the collected data to the base station(s). Meanwhile, base stations also serve as gateways to external networks, for example the Internet (cloud infrastructure). The sink node gathers all data from sensor nodes, performs simple processing, and sends the relevant information (or processed data) via the Internet to users who have requested it or who use the information [1].

To send data to the base station, each sensor node can use a single-hop long distance transmission, which leads to the single-hop network architecture. However, long-distance

transmission is costly in terms of energy consumption. In sensor networks, the energy consumed for communication is much higher than the energy required for data collection and computation. Therefore, it is desirable to reduce the amount of traffic and the transmission distance in order to increase energy savings and extend the life of the network.

In this case, short distance communication is highly preferred. In most sensor networks, the sensor nodes are highly deployed and the neighbouring nodes are close to one another, which makes it possible to use short-distance communication. In a multi-site communication, a sensor node transmits its sensed data to the base station via one or more intermediate nodes, which can reduce the power consumption for communication.

In a multi-site network, sensor nodes can be clustered, where cluster members send their data to cluster heads, while cluster heads serve as relays to transmit data to the base station. A low energy node can be used to perform the sensing task, send the sensed data to its cluster header at a short distance while a higher energy node can be selected as a cluster head to process data from the cluster members, and transmit them to the base station. This process can not only reduce the energy consumption for communication but also balance the traffic load and improve scalability as the network size increases.

In addition, aggregation of data can be performed on cluster heads to reduce the amount of data transmitted to the base station and improve the energy efficiency of the network. For example, if a weather station picks up temperature values with various temperature sensors, it is obvious that the temperature is the same for a given locality. Instead of sending each value to the base, the cluster head could perform small calculations on the data as the average in order to reduce network traffic while saving energy and increasing network longevity.

The objective of data aggregation is to combine and generalize the data coming from several sensor nodes in order to reduce the data to be transmitted. The majority of applications using the WSN require a certain level of security, the encryption of the data collected before transmission is preferable.

In the previous example of meteorology, the sensor nodes sense the ambient temperature; they encrypt and transmitting it to the cluster head. As the temperature's data come from the same neighbourhood it is possible to prepare an aggregation by summing them at the cluster head and the result is sent to the base station as the same time as the number of the nodes sent these encrypted values. In fact, the average of the temperature of the environment is calculated by the base station by summing aggregated data from cluster headers and then it can decrypts this sum in order to be divided by the addition of numbers of all nodes that took these measurements.

Suppose we have three clusters A , B and C (showed on the *Figure 49*). Cluster A comprises five sensor nodes, three sensor nodes for cluster B and four sensors for cluster C . In this example, the cluster headers do not collect the temperature values; they serve only to aggregate data from children nodes. Suppose we use an additive homomorphic encryption algorithm, for example the Pailler's cryptosystem. The cluster head A receives encrypted values A_1, A_2, A_3 and A_4 from children nodes and calculate the output A_s as the result of aggregation on inputs. A_s is sent to the base station with the number 4 corresponding to the number of nodes took the corresponding values. For Pailler's cryptosystem, in order to achieve the aggregation on cluster header, it necessary to perform multiplication on inputs.

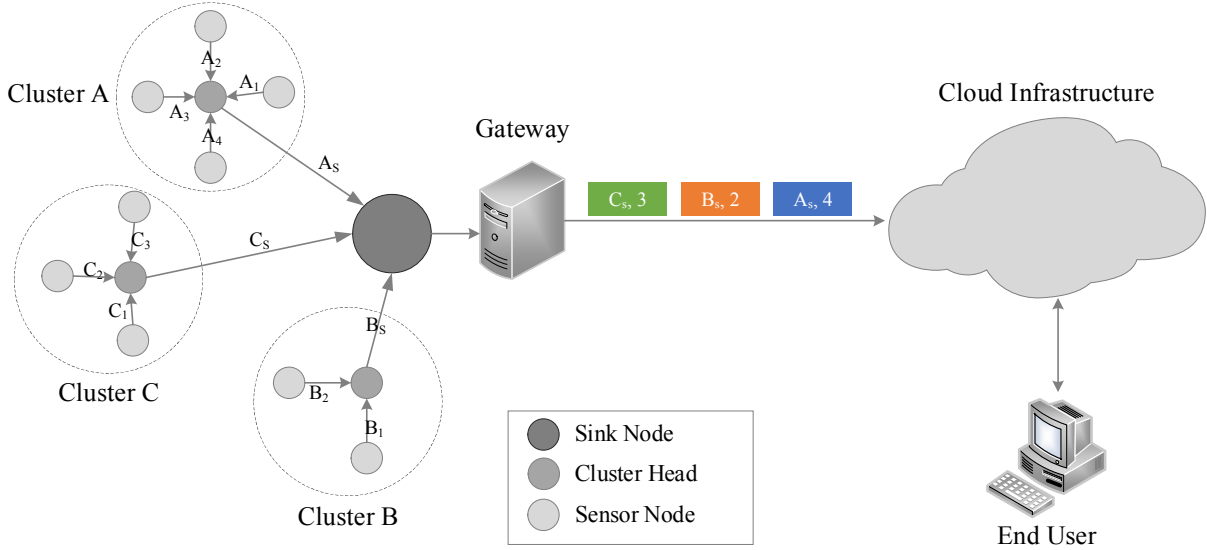


Figure 49 Homomorphic encryption on aggregated data in the integrated Cloud-WSN

$$A_s = A_1 \cdot A_2 \cdot A_3 \cdot A_4,$$

$$A_s = g^{(v_{A1} + v_{A2} + v_{A3} + v_{A4})} \cdot (r_{A1} \cdot r_{A2} \cdot r_{A3} \cdot r_{A4})^N \bmod N^2, \text{ where: } A_1 = \text{Enc}(v_{A1}) = g^{v_{A1}} \cdot r_{A1}^N \bmod N^2,$$

$$A_2 = \text{Enc}(v_{A2}) = g^{v_{A2}} \cdot r_{A2}^N \bmod N^2,$$

$$A_3 = \text{Enc}(v_{A3}) = g^{v_{A3}} \cdot r_{A3}^N \bmod N^2,$$

$$A_4 = \text{Enc}(v_{A4}) = g^{v_{A4}} \cdot r_{A4}^N \bmod N^2$$

By using the same algorithm, the cluster header B and C send to the base station respectively $(B_s, 2)$ and $(C_s, 3)$, where $B_s = g^{(v_{B1} + v_{B2})} \cdot (r_{B1} \cdot r_{B2})^N \bmod N^2$ and $C_s = g^{(v_{C1} + v_{C2} + v_{C3})} \cdot (r_{C1} \cdot r_{C2} \cdot r_{C3})^N \bmod N^2$. Then, the base station calculates the average by the following formula:

$$\text{Average} = \frac{\text{Dec}(A_s \cdot B_s \cdot C_s)}{4+2+3}$$

$$\text{Average} = \frac{\text{Dec}(g^{(v_{A1} + v_{A2} + v_{A3} + v_{A4} + v_{B1} + v_{B2} + v_{C1} + v_{C2} + v_{C3})} \cdot (r_{A1} \cdot r_{A2} \cdot r_{A3} \cdot r_{A4} \cdot r_{B1} \cdot r_{B2} \cdot r_{C1} \cdot r_{C2} \cdot r_{C3}) \bmod N^2)}{9}$$

$$\text{Average} = \frac{v_{A1} + v_{A2} + v_{A3} + v_{A4} + v_{B1} + v_{B2} + v_{C1} + v_{C2} + v_{C3}}{9}$$

Assume that the clusters A , B and C represent the neighbourhoods of a city, the calculated average will be the temperature of a city that can be sent over the Internet, for example on a weather website. Since the base station has already decrypted the aggregated values from each cluster, it seems obviously that the average sent over the Internet will be in plaintext and that the confidentiality of the data is not ensured on the Internet. Afterwards we will see how we can transmit these encrypted data on the Internet while ensuring its integrity and confidentiality.

Back to our example of weather, we want to keep the data archive in the cloud for future use. From the previous paragraph we can extend the logical architecture by sending immediately encrypted data $(A_s, 4)$, $(B_s, 2)$ and $(C_s, 3)$ in the cloud instead of calculating the average at the base station. The base station will behave as a gateway without perform any processing on the data.

It is assumed that there is a database in the cloud to accommodate data from WSN. This data is stored according to the date on which it was collected. When a user wants to perform arithmetic operations on the encrypted data, for example calculating the average temperature of the day, he/she just needs to send a request to the cloud, the cloud in turn retrieve the stored data on the indicated date in the query to apply the homomorphism property to the encrypted data. In our

previous example, the data is encrypted by the Pailler's algorithm, the cloud will only have to multiply all the entries of the day and the result is an encrypted representing the sum of these entries. In order to calculate the average, the result and the number n of the nodes that participated in the sensing are sent to the user. The workstation of the user decrypts this result and divides it by this number n to find the daily average.

5.2. Modelling Context and General Hypotheses

Previously, we discussed about aggregation of data in WSN and integration of WSN with cloud infrastructure using Pailler as homomorphic encryption algorithm but we can ask ourselves if this algorithm is the best for wireless sensor networks in terms of the resources needed to encrypt, like the sensor nodes' CPUs and the memories containing the programs and the encrypted data.

The majority of WSN platforms are not advanced like traditional computers that we are using in everyday life, the actual wireless sensor nodes embed few kilobits microprocessors ranging from 8 bits up to 32 bits [149]. So, let us take an example of an 8-bit microprocessor whose temperature value of 22 Celsius is to be encrypted with the Pailler algorithm using the following parameters:

Private key: $(p, q, r) = (5, 7, 12)$

Public key: $(N, g) = (35, 144)$

Encryption: $c = g^m \cdot r^N \text{ mod } N^2$

$$c = 14422 \cdot 1235 \text{ mod } 352$$

$$c = 14422 \cdot 1235 \text{ mod } 1225$$

$$c = 348$$

At first glance, an 8-bit microprocessor cannot perform this kind of calculations because already the result of encryption is on 9 bits (348 in decimal is equivalent to 101011100 in binary), so there is an overflow, which will lead to the bad results. Another remark is related to the power calculations, in spite of which the operands remain within the limit of 8 bits but this microprocessor cannot calculate 144 exponents 22 or 12 exponents 35 because there would also be an overflow. As a result, algorithms using exponential operations to encrypt the data are not well suited for this kind of device with lower computing power.

Apart from the computing power, another problem is related to energy consumption. To increase the level of security, we have obviously to increase the size of the encryption key. By increasing the size, the ciphertexts become huge and the sensor nodes will be forced to spend more energy for coding, modulation and radio transmissions. A long chain of data leads to more prolonged transmissions, while consuming a lot of energy. As a result, the algorithms to be used for encryption must use the minimum possible key size for an acceptable level of security.

Another challenge is the choice of a type of algorithm to implement, if we use a symmetric encryption algorithm we will be pleased to share the secret key with all the sensor nodes of the network. This presents a huge risk, as we know that the network's nodes are deployed in a hostile environment where the risk of being intercepted by malicious people is inevitable. If an attacker reaches physically a sensor node, he may search until he gets the secret key that this node shares with the other nodes in the network as well as the base station. The best solution is the use of an asymmetric encryption algorithm that has two keys, one public and the other private.

In this proposed system, only the end user can access the data in plaintext, i.e. the end user must generate asymmetric encryption keys. The private key must be shared with the sensor nodes either when programming the nodes or using public key infrastructures [150]. When a node

needs to send its collected data, it encrypts them using the public key that it has previously obtained and sends them to the next node or cluster head which in turn applies the aggregation and sends them to the base station. The base station will send the data to the cloud where they will be stored or sent in real-time to the end users. As long as the data are stored in the cloud and encrypted, only the end user can manipulate them without having to decrypt them since they are homomorphically encrypted. The end user retrieves the encrypted results found from the encrypted data and decrypts them using his/her private key.

Previously, we discussed some criteria necessary to pick out the suitable homomorphic encryption algorithm for connecting the WSN with the cloud, and then we will later use a comparative *Table 3* that will help us to identify a suitable homomorphic encryption algorithm to be implemented in the proposed architecture.

This comparative table makes it easy to determine the algorithm to be implemented under some criteria before mentioned. Elliptic curve cryptosystem seems to be the best suited for low power applications (such as sensor nodes). The EC-ElGamal offers the same level of security with a smaller bit size by reducing processing overhead as compared to RSA or other homomorphic algorithms.

Smaller key sizes result in less power, bandwidth, and computational requirements. This makes EC-ElGamal a good choice for low power environments. EC-ElGamal has applications as a public key sharing scheme and as digital signature authentication scheme.

Due to these factors, ECC is better suited for low bandwidth, computational power and memory situations especially in mobile and wireless environment [151]. So undoubtedly, we can make an ascertainment that EC-ElGamal is the strongest and the fastest (efficient) among the present techniques [151].

Table 3 Comparison of Homomorphic Cryptosystems

Homomorphic cryptosystems	Additive homomorphic operation	Exponential based encryption	Asymmetric encryption
RSA [113]	No	Yes	Yes
Pailler [128]	Yes	Yes	Yes
CMT [152]	Yes	No	No
EC-ElGamal [119]	Yes	No	Yes
Naccache-Stern [125]	Yes	Yes	Yes
Domingo-Ferrer [112]	Yes	Limited	No
Goldwasser-Micali [153]	No	Yes	Yes
Okamoto-Uchiyama [126]	Yes	Yes	Yes
Benaloh [122]	Yes	Yes	Yes
DGHV [148]	Yes	No	No

5.3. EC-ElGamal Cryptosystem

Based on the criteria mentioned above, EC-ElGamal seems to be efficient compared to other cryptosystems but its implementation is not obvious by considering it as homomorphic cryptosystem. Take for example an elliptic curve E over prime integers defined on a finite field F_p is represented by a following equation:

$$E_p(a, b): y^2 = x^3 + ax + b \pmod{p} \quad (1)$$

Since the coefficients a and b are integers chosen from the field F_p and the cubic $x^3 + ax + b$ must not have the repeated roots in F_p which is equivalent to the condition $\Delta = 4a^3 + 27b^2 \neq 0 \pmod{p}$. In addition to the points of the curve, we must define a point \mathbf{O} that we affectionately name "point at infinity".

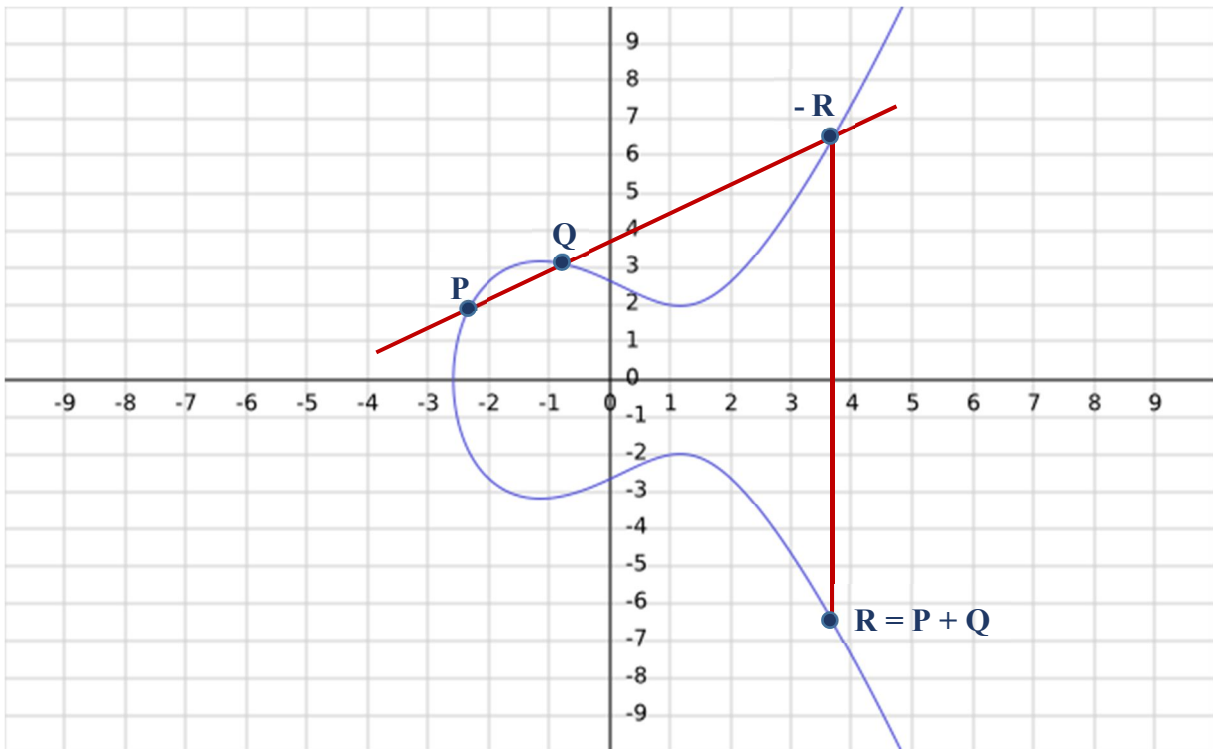


Figure 50 Example of a graph representing the elliptic curve $y^2 = x^3 - 4x + 7$

If $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ are points on $E_p(a, b)$ with $P_1, P_2 \neq \mathbf{O}$, let define $P_3 = (x_3, y_3) = P_1 + P_2$ by:

1. if $x_1 \neq x_2$, then $x_3 = \lambda^2 - x_1 - x_2$ and $y_3 = \lambda(x_1 - x_3) - y_1$, where $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$.
2. if $x_1 = x_2$, but $y_1 \neq y_2$, then $P_1 + P_2 = \mathbf{O}$.
3. if $P_1 = P_2$ and $y_1 \neq 0$ then $x_3 = \lambda^2 - 2x_1$ and $y_3 = \lambda(x_1 - x_3) - y_1$, where $\lambda = \frac{3x_1^2 - a}{2y_1}$.
4. if $P_1 = P_2$ and $y_1 = 0$ then $P_1 + P_2 = \mathbf{O}$.

Elliptic curve cryptography can be used to encrypt the plaintext message m into ciphertext. The plaintext message m is encoded at a point Pm from the finite set of points in the elliptic group, $E_p(a, b)$. The first step consists in choosing a generating point $G \in E_p(a, b)$ such that the smallest value of n for which $nG = \mathbf{O}$ is a very large prime number. The number n (called order of G) must verify the condition $1 \leq n < N$, where N is the number of all points of $E_p(a, b)$ [154].

Assume that Bob and Alice intend to communicate. Each user selects a private key and uses it to compute their public-key. For example, Alice (A) selects a private-key $n_A < n$ and computes

the public-key $P_A = n_A G$. To encrypt the message P_m for Bob (B), Alice chooses a random integer k and computes the ciphertext pair of points P_C using Bob's public-key $P_B = n_B G$:

$$P_C = [(kG), (P_m + k P_B)]$$

After receiving the ciphertext pair of points, P_C , Bob multiplies the first point, (kG) with his private-key, n_B and then subtracts the result to the second point in the ciphertext pair of points, $(P_m + k P_B)$: $(P_m + k P_B) - [n_B(kG)] = (P_m + k n_B G) - [n_B(kG)] = P_m$ which is the plaintext point, corresponding to the plaintext message m . Only Bob, knowing the private-key n_B , can remove $n_B(kG)$ from the second point of the ciphertext pair of point, i.e., $(P_m + k P_B)$ and hence retrieve the plaintext information P_m .

The problem with elliptic curves as an additive homomorphic cryptosystem is that the addition on an elliptic curve must involve only the points on that curve and the result must be a point on that curve (check the previous *Figure 50*). So if we have to encrypt the messages (data from WSN such as temperature for example) as integers we must first map them to the corresponding points on the elliptic curve $E_p(a, b)$. There are several ways of mapping [139], [155] to convert the message m to the point P_m of the curve. Using a method where P_m is a multiple of the generator point G i.e. $P_m = mG$ requires the reverse function to extract the original message m from the given encoded message on the point mG . The mapping function obeys the property of homomorphism because:

$$\begin{aligned} P_{m1} + P_{m2} + P_{m3} + \dots &= \text{map}(m_1 + m_2 + m_3 + \dots) \\ &= (m_1 + m_2 + m_3 + \dots) G \\ &= m_1 G + m_2 G + m_3 G + \dots \end{aligned}$$

Where $m_1, m_2, m_3 \dots$ are integer messages $\in F_p$. To perform homomorphic decryption the reverse mapping function is needed, *such function is required to resolve the discrete logarithm problem over an elliptic curve. For this reason, the reading device (sensor node, cluster header of the WSN or the end user) may not execute the reverse mapping function because of lack of computation power [139] unless we find a particular elliptic curve that leads easily to the reverse mapping (this case will be the topic of the further research work)*. Since it seems impossible for us to implement elliptic curves as homomorphic ciphers in WSNs, we must think of the other algorithms that best meet the criteria. The DGHV and the CMT cryptosystem are better candidates approaching the criteria but these two algorithms also have weaknesses against the proposed architecture.

First, DGHV is applicable to the binary number, which implies that each time it is necessary to send the messages (the data in the case of WSN) that it is also necessary first to convert them in binary then to apply the encryption what includes a costly step in energy consumption and time. The second disadvantage is the long size of its encryption key, the DGHV generates a large cryptogram that requires more bandwidth; transmit power and sufficient energy to perform encryption. For these reasons aforementioned, DGHV is not the proper cryptosystem for WSNs. Without entering in details of the CMT cryptosystem, it is not adequate for a network using a thousand wireless sensors because the base station shares a unique secret key with each node in the network which makes the aggregation process of ciphertexts more complicated to implement.

5.4. Domingo-Ferrer's Cryptosystem

In Domingo-Ferrer's cryptosystem, the size of parameter d affects the size of ciphertext [156]. Domingo-Ferrer's encryption algorithm is a symmetric-key based cryptosystem uses two secret parameters r_p and r_q for encryption and computes r_p^{-1} and r_q^{-1} for corresponding decryption.

Drastically reducing the size of the parameter d could attenuate the effect of exponentiation for not having very wide cryptogram. Although the cryptosystem is symmetrical, we could use an asymmetric algorithm (EC-ElGamal for example) to exchange the secret key globally throughout the WSN.

- This encryption algorithm begins with the choice of two prime numbers p and q and from them we could easily calculate $n = p * q$. These parameters (p , q , and n) are only available on sensor nodes and end users. The parameter d must be greater or equal to 2.
- Then choose randomly r_p and r_q belonging to the multiplicative subgroup Z_p^* and Z_q^* respectively.
- Before the encryption process, it is necessary to split randomly the message m into secret integers m_1, m_2, \dots, m_d such that $\sum_{i=1}^d m_i = m \pmod n$ and $m_i \in Z_n$. The ciphertext is obtained by computing: $E_k(m) = ([m_1 * r_p^1 \pmod p, m_1 * r_q^1 \pmod q], [m_2 * r_p^2 \pmod p, m_2 * r_q^2 \pmod q], \dots, [m_d * r_p^d \pmod p, m_d * r_q^d \pmod q])$.
- To decrypt, compute the scalar product of the i^{th} $[m \pmod p, m \pmod q]$ pair by $[r_p^{-i} \pmod p, r_q^{-i} \pmod q]$ to retrieve the $[m_i \pmod p, m_i \pmod q]$. Add up to get $[m \pmod p, m \pmod q]$. Use the Chinese remainder theorem [157] to obtain the unique $m \pmod n$.

The Figure 51 shows an unrealistically small example of a Domingo-Ferrer cryptosystem over an integer. This cryptosystem supports homomorphic processing when the ciphertext is obtained using the same key. The size of the parameter d directly affects the size of the cryptogram and each plaintext is divided into d sub-plaintexts so that each of them is encrypted using the secret parameters p , q , r_p and r_q .

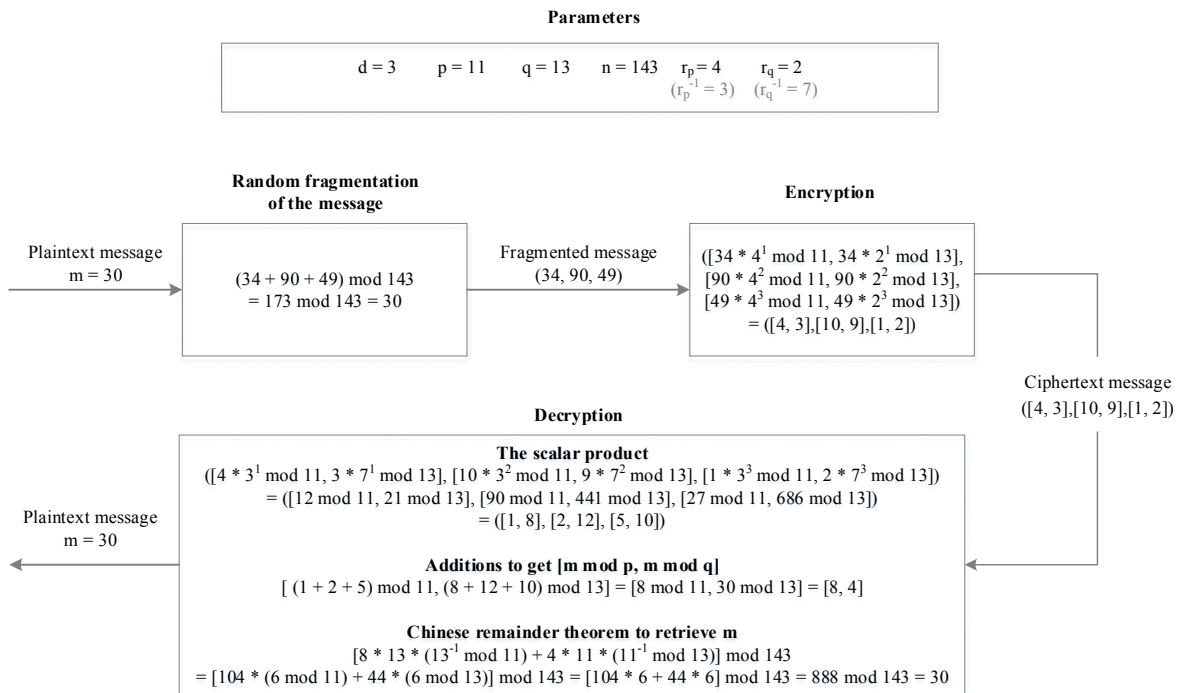


Figure 51 Encryption and decryption using Domingo-Ferrer's cryptosystem

As shown in the tree-based data aggregation topology described in following Figure 52, the aggregator node performs computation over encrypted data. The decryption is carried out at the end user, and it requires an inverse of the secret parameters r_p and r_q . In addition, the decryption operation requires the scalar product of r_p^{-1} , r_q^{-1} and coordinates of the aggregated ciphertext.

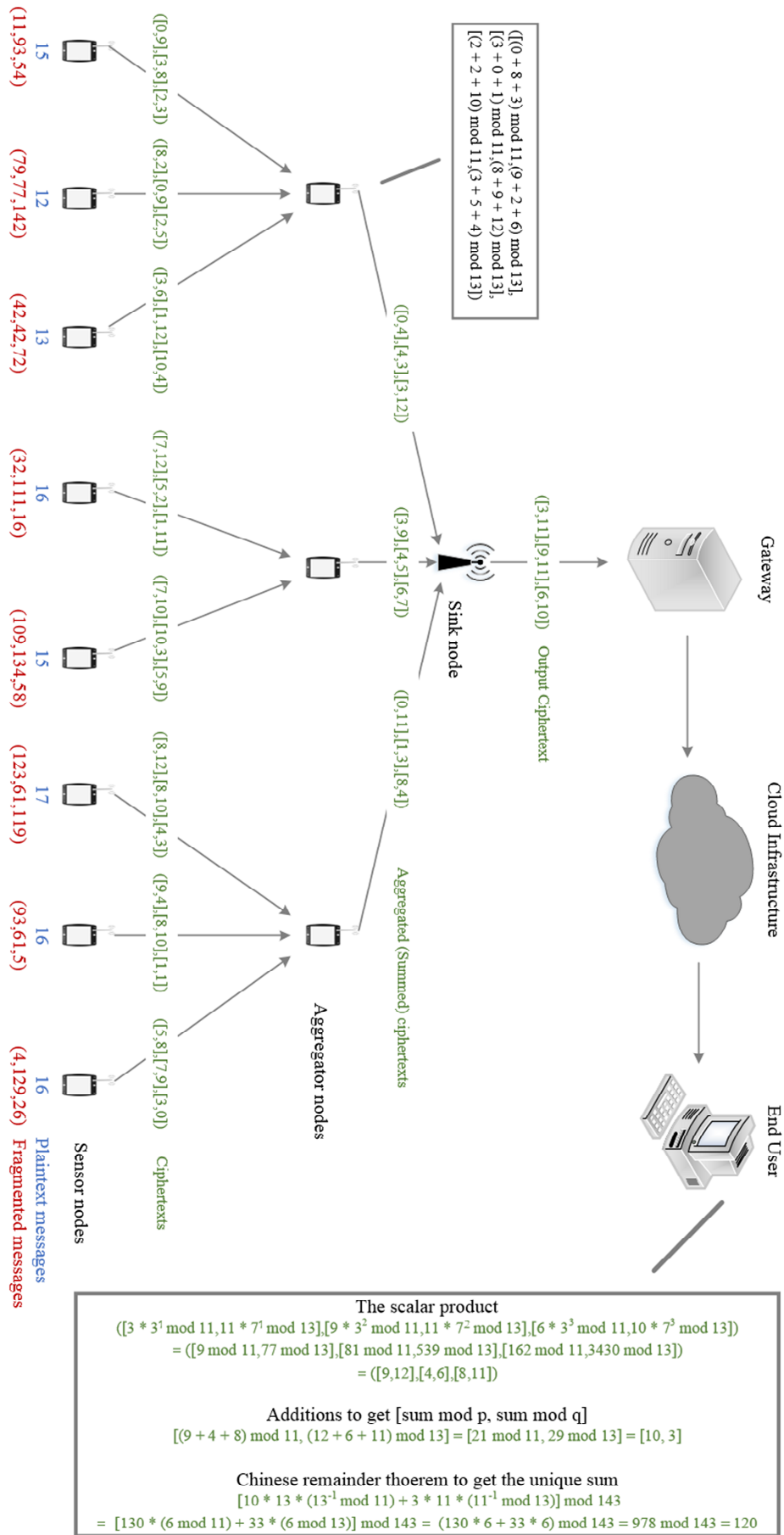


Figure 52 Tree-based data aggregation using Domingo-Ferrer's cryptosystem

5.4.1. Implementation of Domingo-Ferrer Encryption Scheme

The implementation is performed on a 16-bit microprocessor-based sensor node (i.e. the highest digit can handle is $2^{16} - 1 = 65535$) called Wasp mote and manufactured by Libellium (see the section 0 for details). For this implementation we take the values of the ambient temperature (in Celsius degrees) assuming they are positive numbers in the range $[0, 55]$. So the choice of encryption parameters will be based on this interval so that n is greater than 55. Every 15 minutes, each sensor node wakes up, collects the temperature as a float number, and converts it to an integer to be encrypted using the previously programmed Domingo-Ferrer algorithm. This cryptosystem is applicable to the integer numbers but in order to produce accurate results, instead of converting the floats values of temperature to the integers, we could multiply each value by 100 for example but this would lead to excessive calculations. The programming of the Wasp mote sensor nodes uses C++ language and code is loaded on the sensor node via the USB cable (check Appendix A to see the full source code). After encryption, the sensor node creates a ZigBee frame and sends the temperature as ciphertext to the MAC address of the base station. For this implementation, we do not include clustering due to lack of sufficient sensor nodes and the data are sensed every 10 seconds in order to accelerate the experiments. Through the sink node (connected to a laptop acting as a gateway toward the cloud), the received ciphertexts are parsed and then routed to the cloud for storage. The complete configuration is shown in *Figure 53*.

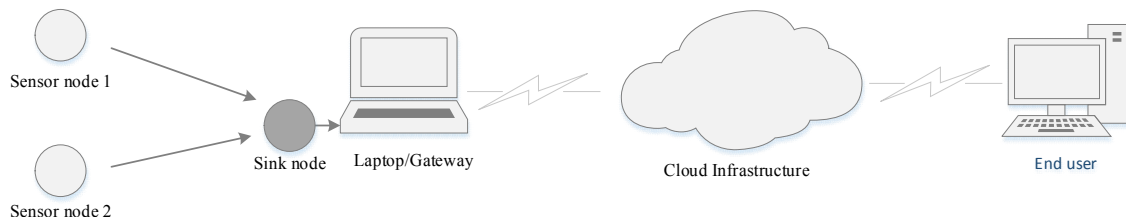


Figure 53 Configuration of the Domingo-Ferrer's Implementation

End users can access data stored in the cloud at any time, data operations must be performed in the cloud, and the user only gets results. Imagine we want to calculate the average annual temperature, as sensor nodes send data every 15 minutes, which is 4 values in one hour and 96 values each day. In a year, each sensor node collects 35040 temperature values. Suppose that 100 sensor nodes compose the WSN, we would have 3504000 values and it would be difficult to manipulate them with end-user computers having not enough power to compute. In the next *Table 4* are the sample of sensed data and we will prove the homomorphism on them (the parameters used for this sampling are: $d = 2$, $p = 17$, $q = 13$, $n = 221$, $r_p = 6$ and $r_q = 9$).

Table 4 Sample of encrypted data received after Domingo-Ferrer's Implementation

Encrypted temperature	Decrypted temperature
([12,12],[9,1])	32
([9,6],[12,6])	33
([1,9],[11,8])	34
([9,5],[14,5])	34
([2,12],[3,4])	33
([12,1],[9,9])	32
Total	198

To demonstrate the homomorphism, just decrypt the sum as ciphertext $([11, 6], [7, 7])$ to verify whether we get the value 198 or not. So, let us start finding r_p^{-1} and r_q^{-1} in Z_p and Z_q respectively.

$$\begin{aligned} r_p * r_p^{-1} \bmod p &= 1 \text{ and } r_q * r_q^{-1} \bmod q = 1 \\ 6 * r_p^{-1} \bmod 17 &= 1 \text{ and } 9 * r_q^{-1} \bmod 13 = 1 \\ \rightarrow r_p^{-1} &= 3 \text{ and } r_q^{-1} = 3 \end{aligned}$$

- Let us compute the scalar product of the i^{th} $[mod\ p, mod\ q]$ pair by $[r_p^{-i} \bmod\ p, r_q^{-i} \bmod\ q]$ to retrieve $[m_i \bmod\ p, m_i \bmod\ q]$:
 $([11 * r_p^{-1} \bmod\ p, 6 * r_q^{-1} \bmod\ q], [7 * r_p^{-2} \bmod\ p, 7 * r_q^{-2} \bmod\ q])$
 $= ([11 * 3 \bmod\ 17, 6 * 3 \bmod\ 13], [7 * 3^2 \bmod\ 17, 7 * 3^2 \bmod\ 13])$
 $= ([16 \bmod\ 17, 5 \bmod\ 13], [12 \bmod\ 17, 11 \bmod\ 13])$
- Let's add $[m_i \bmod\ p, m_i \bmod\ q]$ to obtain:
 $[m \bmod\ p, m \bmod\ q] = [11 \bmod\ 17, 3 \bmod\ 13]$
- Finally, we have to apply the Chinese Remainder Theorem to obtain the unique $m \bmod\ n$. Assume that:

$$\begin{aligned} z_1 &= n/p = q \\ z_2 &= n/q = p \\ y_1 &= z_1^{-1} \bmod\ p = 4 \bmod\ 17 \\ y_2 &= z_2^{-2} \bmod\ q = 10 \bmod\ 13 \end{aligned}$$

Thus, $m = (m_1 y_1 z_1 + m_2 y_2 z_2) \bmod\ n = (11 * 4 * 13 + 3 * 10 * 17) \bmod\ 221 = 198$.

Hence, we proved the homomorphism on temperature values using the Domingo-Ferrer algorithm.

5.4.2. Performance Analysis of the Domingo-Ferrer's Implementation

With a 16-bit processor, we have carefully to choose the parameters to encrypt the data from the WSN. First of all, the choice of p and q must be such that the maximum value to be encrypted must be strictly less than n because by splitting the message m into m_1, m_2, \dots, m_d , all these short messages must be less than n also. During the implementation we used $n = 221$ with the temperature values included in the interval $[0, 55]$, this assures us that this condition is sufficiently verified. Second condition is related to the randomly chosen parameters r_p and r_q , these two belong respectively in Z_p and Z_q , and consequently they can take any value but realistically small so that the r_p^d and r_q^d (also r_p^{-d} and r_q^{-d}) are not very big to saturate the maximum bits capacity of the processor. The third criterion is the choice of the parameter d , the more it is bigger the more we have several splits of message. Therefore, the r_p^d (r_p^{-d}) and r_q^d (r_q^{-d}) become relatively large which can lead to the bad results if during the encryption the calculations include the intermediate results exceeding the 16 bits (the maximum bits for the sensor node's microprocessor).

Table 5 Experiment configuration and range of varying parameter values

Implementation	Sent messages	p	q	n	r_p	r_q	d
(a) No cryptography	10,000	-	-	-	-	-	-
(b) Encryption with 2 fragments	10,000	17	13	221	3	2	2
(c) Encryption with 3 fragments	10,000	17	13	221	3	2	3
(d) Encryption with 4 fragments	10,000	17	13	221	3	2	4

To evaluate the effects of Domingo-Ferrer encryption on the life state of the sensor node's battery, we carried out four experiments. The first is about to send 10,000 messages (temperature and battery level) in plaintext (Figure 54). The second experiment (Figure 55) is about to send the same number of messages but being encrypted (whose encryption parameters are $p = 17$, $q = 13$, $n = 221$, $r_p = 3$ and $r_q = 2$) with the parameter d equal to 2. The third experiment (Figure 56) repeats experiment 2 but changing $d = 3$ and the fourth (Figure 57) is identical to the last three but with 4 fragments of plaintext ($d = 4$). The experiment configurations and parameters are summarized in the previous Table 5.

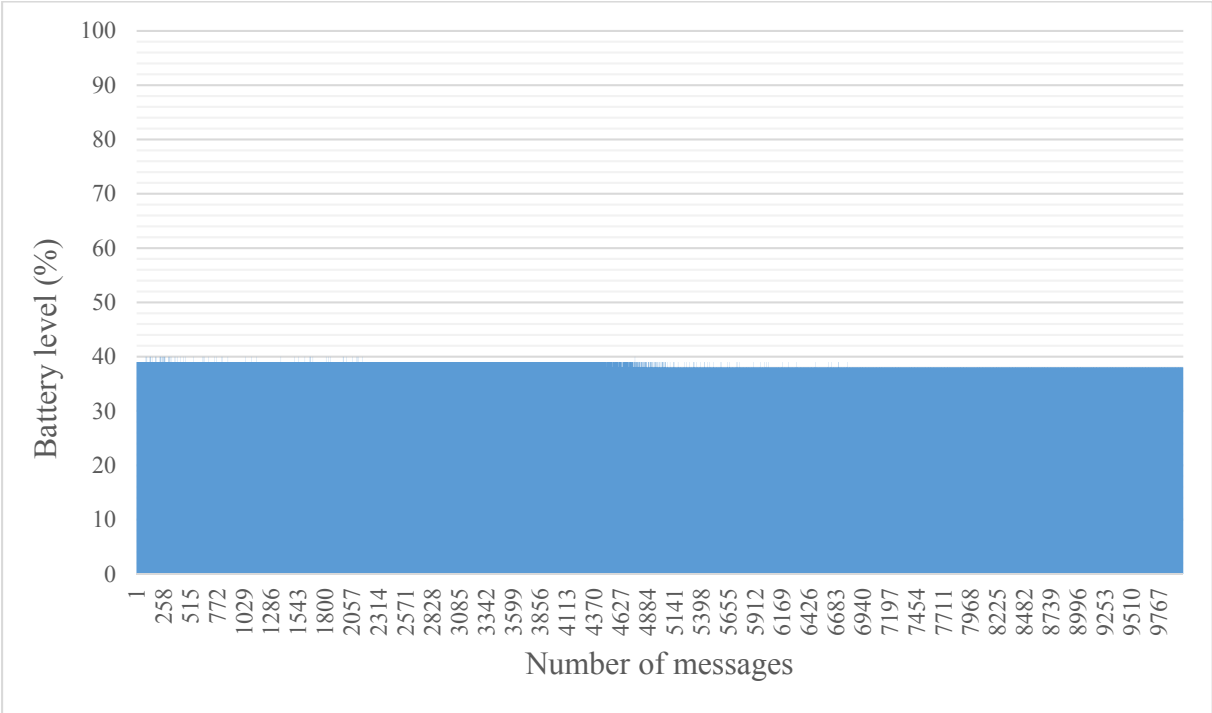


Figure 54 Battery status without Domingo-Ferrer homomorphic encryption

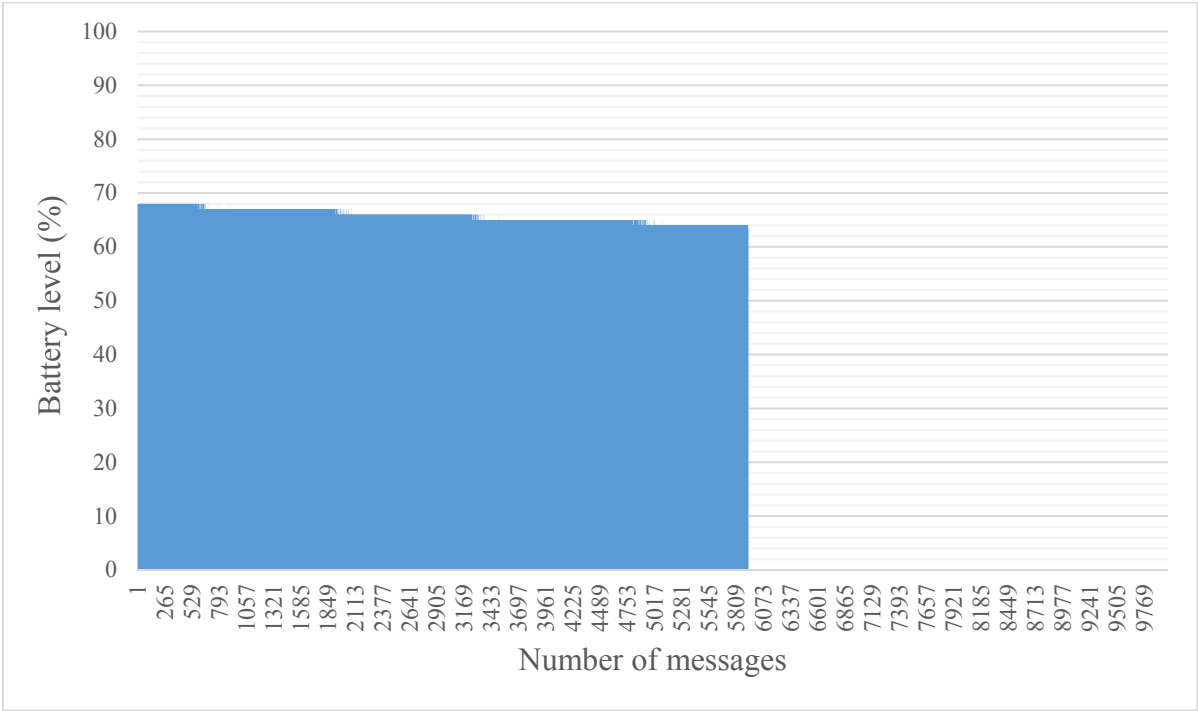


Figure 55 Battery state with Domingo-Ferrer homomorphic encryption using 2 message fragments

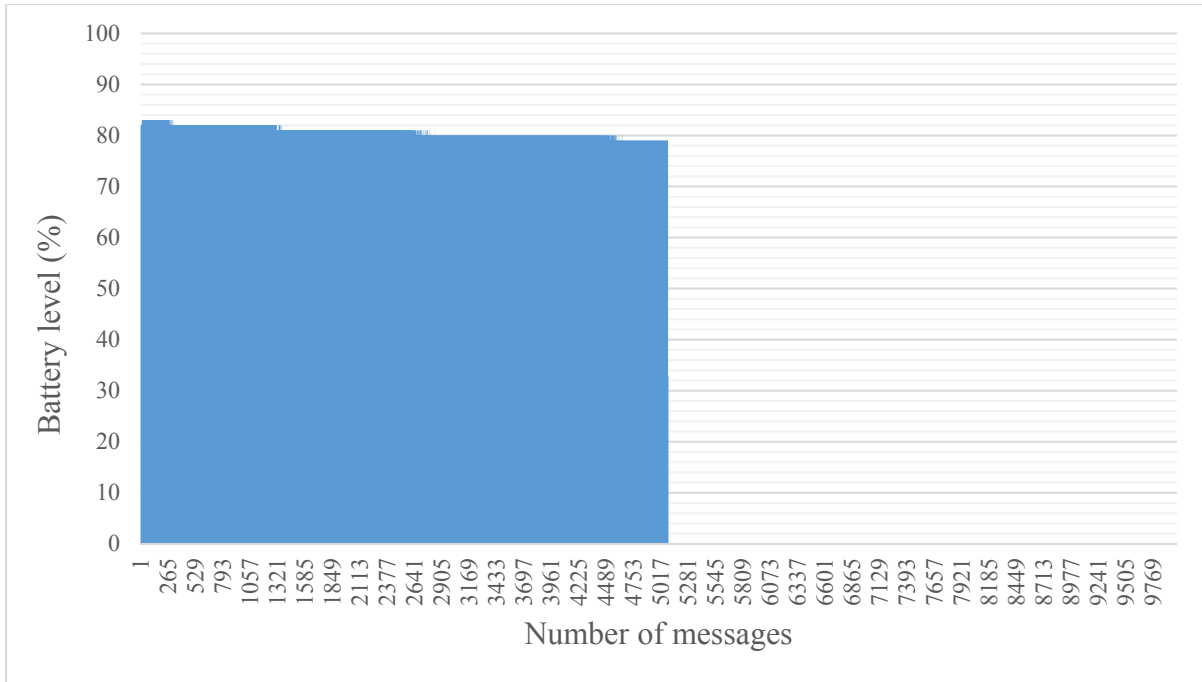


Figure 56 Battery state with Domingo-Ferrer homomorphc encryption using 3 message fragments

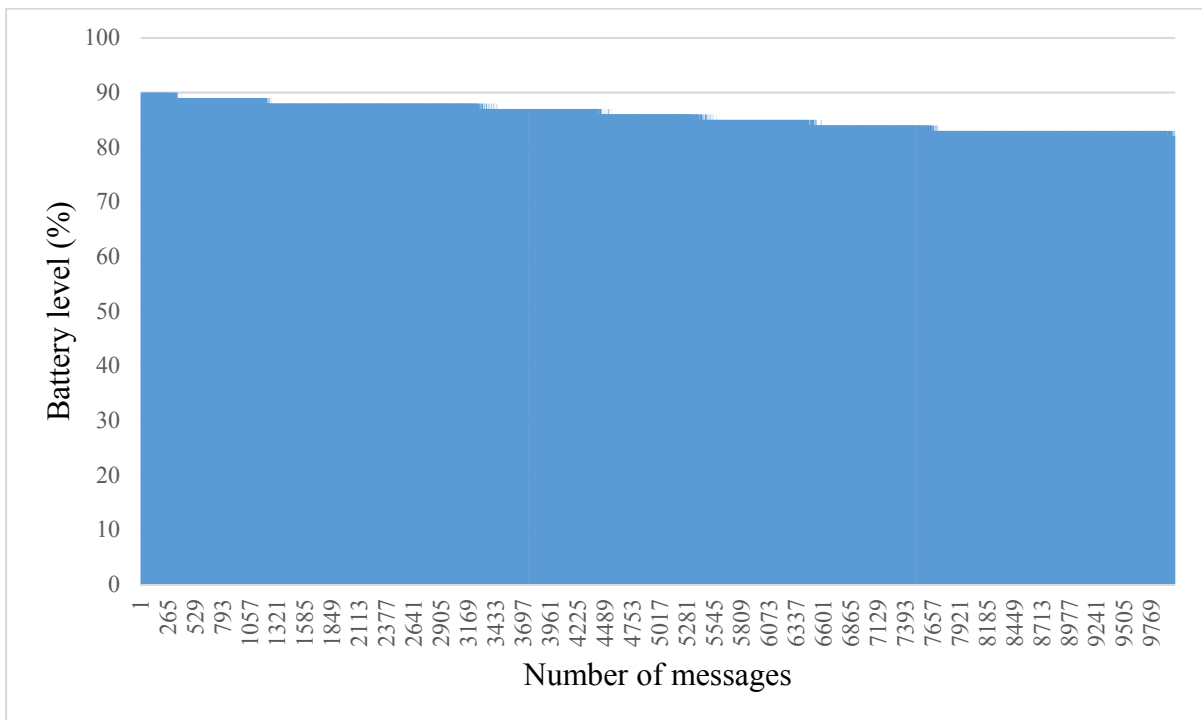


Figure 57 Battery state with Domingo-Ferrer homomorphc encryption using 4 message fragments

Note in *Figure 54* when the 10,000 messages are sent as plaintext the battery level of 39% drops to 38%, so the sensor node loses one percent, which is relatively convincing for autonomous sensor nodes. When applying the encryption to the data collected with the parameter $d = 2$ we notice that at the end of the sending of 10,000 messages the level of the battery has dropped 6% (from 68% to 62% by referring to the *Figure 55*). By modifying the parameter $d = 3$, we notice a slight difference with the result obtained with $d = 2$ because in *Figure 56* we had the battery level which varied from 82% to 75%, i.e. 7% in this case and 1% more using $d = 2$. On the *Figure 57*, the battery has dropped from 90% to 82% with $d = 4$, so undoubtedly we can

comment that when increasing the number of fragments of the plaintext is slightly decreasing the whole energy of the wireless sensor network (Find the summary of the obtained results in the *Table 6*).

Comparing with the results obtained in other related works [143], we can validate our methodology because we come with the same ascertainment that even if several message splits offer better security they have a detrimental impact on the whole network's energy, thus the number of fragments of the message should be kept at minimum.

Table 6 Summary of the experiment results.

Implementation	Sent messages	Battery level before experiment	Battery level after experiment	Dropped %
(a) No cryptography	10,000	39%	38%	1%
(b) Encryption with 2 fragments	10,000	68%	62%	6%
(c) Encryption with 3 fragments	10,000	82%	75%	7%
(d) Encryption with 4 fragments	10,000	90%	82%	8%

5.5. Discussion and Conclusion

Homomorphic encryption is useful in some applications of wireless sensor networks; its use has a positive impact firstly by promoting the confidentiality of the aggregated data and then allowing it to be stored safely into the cloud while performing arithmetic operations on the encrypted data. The use of a symmetric encryption algorithm such as the Domingo-Ferrer cryptographic system is implementable in the WSNs. Therefore, it is necessary to know the specifications of the sensor nodes in order to choose the best encryption parameters in order to avoid depletion of the network energy or exceeding the value limits that the sensor nodes' processors can support. Mutual encryption for WSN and cloud computing seems to be an essential solution for two types of networks, which one is WSN and the other is the Internet. Despite the advantages of these homomorphic encryption algorithms, they also have disadvantages to wireless sensor networks; the encryption process requires power calculations that consume a lot of energy, which makes the network vulnerable to longevity. The use of these algorithms lies between the importance of the project to be protected and the profitability of the network.

This chapter proposes a solution to secure the aggregated data by using Domingo-Ferrer's homomorphic encryption scheme with the objective of understanding its impact on the life of the WSN. From the obtained results, we find that although more message splits offer a higher level of security, generally in terms of performance the energy of the wireless sensor network is penalized. Therefore, message splitting should be kept to minimum. To maintain the balance between the expected services and the security level, it would be better to choose the encryption parameters from the basis of the desired performances and the energy state of the sensor nodes. Thereafter, we will work on the arithmetic operations that Domingo-Ferrer homomorphic encryption is able to ensure.

Chapter -6-

Implementation of the WSN-Cloud Architecture

(Case Study: Smart Cities)

According to the survey carried out in the city of Fez/Morocco, the company distributing the electricity in the city called RADEEF (in French, Regie Autonome intercommunale de Distribution d'Eau et d'Electricite de Fès) has the ways to recognize the breakage of an electrical cable but there is no way to know if the electric or even light poles are inclined. Accordingly, this chapter is mainly dedicated to implement this tilt sensing system in the electric transmission network. The implemented system can transform from accelerometer's data into useful information such as tilt angle for end-users. The tilt may be temporary when an external force acts on the electric tower as wind, earthquake, etc. or permanently for example due to the beam fatigue, the depression in the ground, accident, etc.

In fact, to maintain an electric or lighting tower in a vertical position in order to face gravity, the angle of inclination between the pole and its attachment point (floor, ground, etc.) must be very close to 90° . The tilt recognition can prevent the future status of the beams, if the tilt angle becomes lower than threshold, the beam may fall therefore damage all that it raised and all that were around. So, by fixing the accelerometer sensor on the electric tower in a best-known position, it follows the movement of the tower and at the same time the sensed data relating to the position of this one are sent to the processing and monitoring center.

Basically, the linear tri-axial accelerometer sensor principle is to detect acceleration experienced on each one of the three axes, but by transforming the values from these three axes with a mathematical model, it is possible to calculate with precision the angles created by its axes with the direction of gravity. This tilt recognition method uses the calculation and mapping of the sensed values by the tri-axial accelerometer to the corresponding angle. The z-component corresponding to the gravity allows recognizing the real-time tilt of the horizontal plane while the x and y-components indicate in which side the plan is bent. An end user application processes the angle results of the mapping and plots the 3D scene on the monitoring screen reproducing the behaviour of electric pole remotely located. The experimental results of this approach are satisfactory because the average error that can be committed by comparing the calculated and measured angle is about 1.0° .

6.1. Smart Cities and the Wireless Sensor Networks

Town planners and mayors are increasingly using technologies such as sensors, data management systems and analytics tools to monitor and analyze traffic flows, energy consumption and the use of public transport. Thus, they can better synchronize fires to smooth traffic, or deploy more efficiently a public transport network to cope with demand. The adoption of the smart city concept is partly based on the emergence of technologies and IT trends, such as the deployment of the Internet of Things architectures, capable of collecting data from remote devices and mobile devices connected to the wireless networks (Wi-Fi, ZigBee, Ad-Hoc, etc.) and the proliferation of GPS devices that constantly capture and transmit location data. The success of the smart city comes from the use of wireless sensor networks that collect various information about dust, street lighting, cracks in bridges and buildings, and so on.

Few years ago, the sensor nodes were confined to the simple role of detector: temperature, smoke, intrusion, etc. They are now asked to collect several pieces of data, to communicate with each other, and even to analyse the sensed data. A wireless sensor network (WSN) is composed of a set of on-board processing units, called "motes", communicating via wireless links. The general purpose of a WSN is the collection of a set of environmental parameters surrounding the motes, such as the temperature or pressure of the atmosphere, in order to route them to processing center. Industrial processes, military tracking applications, habitat monitoring, and precision farming are just a few examples of a wide and varied range of possible applications for continuous monitoring offered by WSN. Unfortunately, the WSN are not perfect, because of their low cost and their deployment sometimes in hostile environment, the motes are quite fragile and vulnerable to various forms of failures: breakage, low energy, etc. These problems make the WSN systems innately fragile, which must be considered a normal property of the network.

Information and Communications Technology (ICT) will be at the heart of the Smart City of the future. ICT development will allow for better urban management through obtaining, analysing and managing a variety of information. This information helps administrators of urban territories in decision-making and enables firstly, improving existing services and, secondly, to provide new services to the community (smart street lighting, management of urban tolls, smart parking, etc.). Smart City is a concept which urban development is directly based on multiple information and communication technologies. One of the main topics of the smart city is the use of wireless sensors networks where the different assets of the cities (Lightning poles, Bridges, etc.) are managed by a variety of wireless sensors grouped in network for collecting various data. For example, the detection of dust, traffic light, cracks on bridges or buildings, etc.

6.1.1. Understanding Data and Information in the Smart Cities

Data is defined as a collection of facts and details such as text, figures, observations, symbols, or simply a description of objects, events, or entities gathered together to draw conclusions. This is the raw fact that must be treated to obtain information. These are unprocessed data, which contain numbers, statements and characters before the researcher refines them. The term data is derived from the Latin term "datum" which refers to "something given". The concept of data is related to scientific research, collected by various organizations, government departments, institutions and non-governmental organizations for various reasons.

Information is described as the form of data that is processed, organized, specific and structured, presented in the given context. It gives meaning and improves the reliability of data, ensuring comprehensibility and reducing uncertainty. When the data is transformed into information, it is devoid of useless details or immaterial things, which have some value for the researcher. The term information was discovered from the Latin word "informare", which means "to give shape to". The raw data is not at all meaningful and useful as information. It is refined and cleaned thanks to a deliberate intelligence to become an information. As a result, the data is manipulated by tabulation, analysis, and other similar operations that improve the explanation and interpretation.

To understand the difference between data and information, let take an example in agriculture, a farmer does not always need to know the collected data related to the temperature, pressure, humidity, etc. Because sometimes he/she does not understand how those data can be useful for him/her but if the weather service carries out a forecast from these data, the farmer will be able to have the information about rain, drought or thunderstorms. Therefore, a farmer needs to know if it is going to rain rather than know the data used to predict the rain. In the Smart City, users

need also information about city activities such as less traveled roads, disaster preparedness, etc.

6.1.2. Sensors at the basis of the Smart Cities

In smart cities, a network of sensors, cameras, wireless devices, data centers forms the key infrastructure that enables civilian authorities to provide the necessary services faster and more efficiently. Smart cities are also much more environmentally friendly, as they use environmentally friendly materials for building projects and reduce energy consumption. Efficient use of technology helps to create an efficient transport management system, improve medical facilities and create a reliable communications network for all enterprises, people and beyond, between central and subnational government levels.

There will be an urban environment that constantly communicates with citizens and is able to manage public services in real time in order to improve their quality of life through traffic management, garbage collection, waste disposal, irrigation systems, parking through, local authorities being alerted about the incident and allows the government to stay in touch with people.

A smart city that can become both environmentally sustainable and attractive to citizens and enterprises requires a new kind of intellectual infrastructure. An innovative and open platform based on intelligent sensor networks that can help projected cities to more predictably integrate a complex set of services that requires costs efficiently, at a pace and scale. While many smart city technologies, including smart grids, smart meters and real-time transportation information, are already included in the pilot programs. Here are some of the main components of the Smart City:

- **The use of sensory technology:** A smart city can create an effective and intelligent platform for providing services to state and municipal employees by installing sensors in the city and creating platforms that allow sharing information and providing it for the proper use of the population, city managers, business and professionals. The platform may have a common data repository, where various sensor systems store their information.
- **Remote control network:** The integrated control network with a common data transmission infrastructure controls all municipal and supply networks of service companies participating in the project. The goal is to manage and learn about normal consumption, incidents and opportunities in these networks, all independently of municipal services. All networks have warning devices and monitor consumption, flows, intrusions, etc., which allows to act in case of leaks.

Today's city councils face many challenges that require effective monitoring. Noise pollution is a common environmental problem that affects both the quality of life and health. Atmospheric pollution, whether in the form of gases such as CO₂ and NO₂ or dust, poses a threat to the health of urban residents, causing respiratory diseases. Since many large structures, including buildings, bridges and roads, are located in cities, ensuring the reliability of such structures is crucial for public safety. Last but not least, traffic management is a serious problem, since it is necessary to minimize emissions and avoid unnecessary travel. To date, some of these problems have been solved separately using separate vertical solutions; but city councils cannot afford to use one sensor network for each type of problem. The Smart Cities platform allows system integrators to deploy a heterogeneous wireless sensor network with a combination of sensor boards for city councils. Afterwards, we are going to discuss some possibilities of using wireless sensor networks in the smart cities.

6.1.2.1 Noise Pollution Sensing

The natural migration of populations to cities and their exponential growth makes the maintenance of a sustainable environment in urban areas a major challenge. The increase in the urban population and the resulting frenetic activity result in a proportional increase in the noise generated. The main source of noise in cities today is road traffic. However, there are other more difficult sources to control that cause citizens more nuisance, such as outdoor activities and nightlife.

Government authorities are concerned about the environmental and human risks associated with pollution, especially gas emissions. But another type of pollution also greatly affects the health of cities and residents. Noise pollution is increasing in urban areas, mainly due to the impact of transportation, construction work or industry. In fact, the World Health Organization (WHO) stated that noise pollution is the second cause of disease for environmental reasons after air pollution. The report explains how harmful the traffic noise is to health, which currently affects every fifth Westerner who is regularly exposed to noise at night.

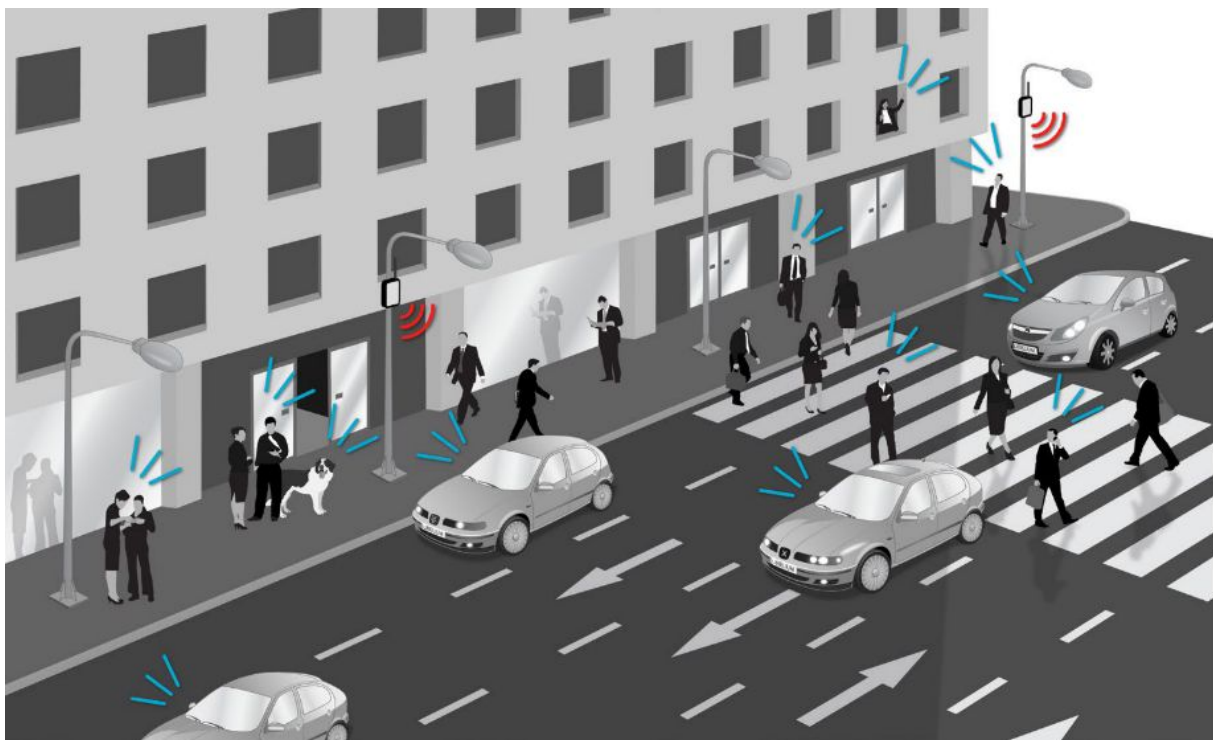


Figure 58 Wireless Sensor Network for Noise Sensing

Since the advent of new technologies of the Internet of Things and wireless sensor networks in the field of environmental acoustics, it is now possible to measure noise continuously [158] and know the sound decibels at any time. Based on the sound sensor measurements (by using an example, the *Figure 58* illustrates the way that the noise pollution is sensed), real-time information is provided on the equipped zones and alerts are generated when a limit is exceeded. The network of wireless sensors for noise pollution monitoring allows the development of the city and sustainable strategies adapted to the real sound situation of the environment.

6.1.2.2 Air Quality Sensing

Contamination by particles (also known as "particulate matter") in the air includes a mixture of solid particles and liquid droplets. Some particles are emitted directly; others form in the atmosphere when other pollutants react. Particles come in different sizes. A diameter of less

than 10 micrometers (PM10) is so small that they can enter the lungs, which can cause serious health problems. Ten micrometers is less than the width of a single human hair.

Particle size is directly related to their ability to cause health problems. Small particles with a diameter of less than 10 micrometers present the greatest problems as they can affect the lungs and the heart. Larger particles are less dangerous, although they can irritate the eyes, nose and throat.

- **Small particles (PM2.5):** Particles with a diameter of less than 2.5 microns are called "fine" particles. Sources of small particles include all types of combustion, automobiles as shown on the *Figure 59*, power plants, burning wood in residential buildings, forest fires, burning in agriculture and some industrial processes. PM2.5 contains more toxic heavy metals and hazardous organic pollutants and can get directly to the lungs. It is easier to join the bacteria and viruses in the environment, so the particles have a greater impact on the environment and human health.
- **Large dust particles:** Particles with a diameter of 2.5 to 10 micrometers are called "coarse". Sources of coarse particles include crushing or shredding operations, as well as dust lifted by vehicles driving on roads.



Figure 59 Wireless Sensor Network for Particle Pollution Sensing

Prolonged exposure is associated with problems such as decreased lung function and the development of chronic bronchitis and even premature death. Short-term exposure to particles (hours or days) can aggravate lung disease, cause asthma attacks and acute bronchitis, and can also increase susceptibility to respiratory infections. In people with heart disease, short-term exposure is associated with heart attacks and arrhythmias. Healthy children and adults may experience temporary minor irritation with increasing particle levels.

The new particle sensor uses light scattering theory and particle counting technology and can accurately determine the number of particles in its environment in order to provide useful reference data for improving the environment. With the new dust sensor, city planners can get information about the size and density of particles in the range from 1 to 10 microns (PM1 / PM2.5 / PM10).

6.1.2.3 Structural Health Monitoring

Structural Health Monitoring (SHM) is defined as a process whose goal is to obtain information on the state and behaviour of a structure over time[159]. This is an important method for monitoring the sustainability of critical infrastructures, such as electrical grids, oil and natural gas systems, nuclear power plants, or transportation networks. The monitoring process consists of a continuous compilation of the most representative parameters that show the state of the structure. The choice of these parameters depends on several factors, such as the type of construction, its purpose, building materials and environmental conditions.

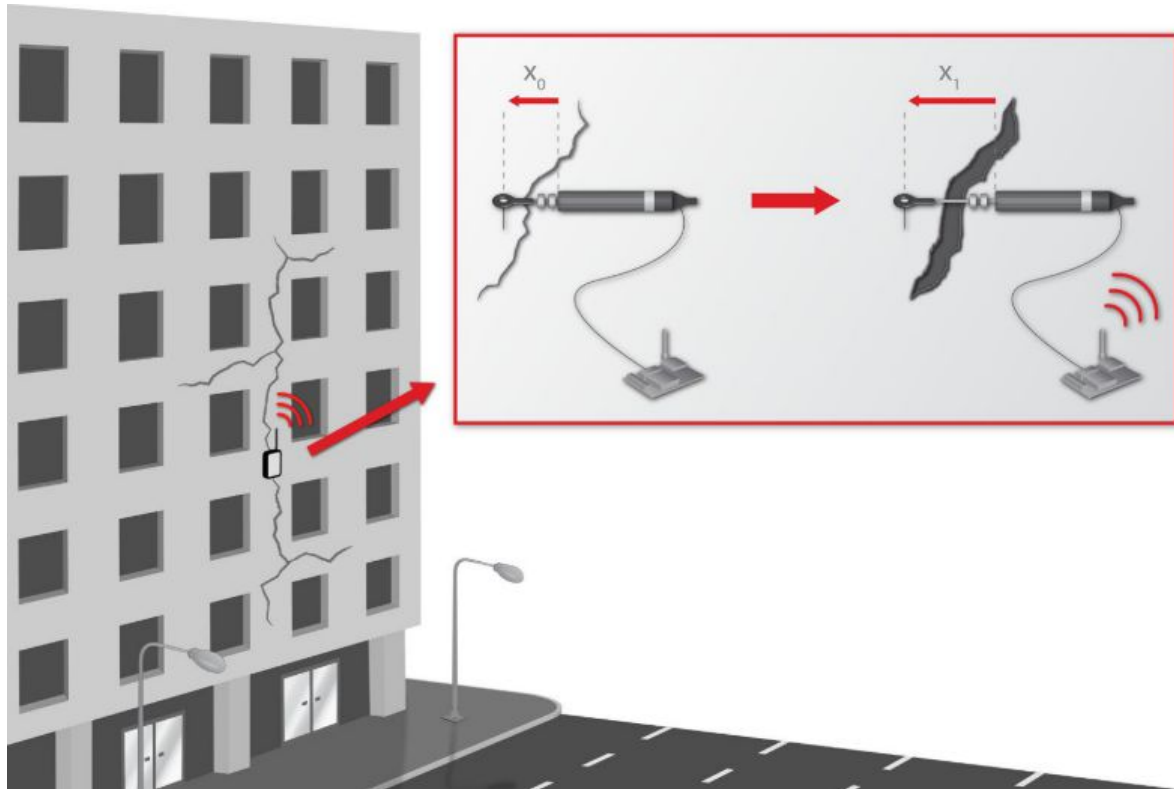


Figure 60 Wireless Sensor Network for Structural Health Monitoring

Crack sensor for measuring cracks in public structures such as buildings and bridges. It can detect displacements of up to 10 micrometers, as well as oscillations and dilatations. This sensor is applicable for monitoring the condition of structures. It is very important to design an adapted housing, in which the sensor nodes are installed, which, in turn, is inserted into the controlled infrastructure, for example, into the hole made in the critical infrastructure as the case of the *Figure 60*. Packaging must be sealed and protected from unauthorized attacks, as well as weatherproof. However, if due attention is paid to how the housing is arranged, this approach will allow operators to easily change a node if necessary (breakage, change monitoring type, etc.). Or remove it to replace a discharged battery. Thus, the monitoring system is integrated into the structure itself, either at its creation or after it, receiving what we might call the Intellectual Critical Infrastructure. This represents an important innovation with respect to traditional deployments of SHM systems, which are performed by placing various devices with sensor nodes at different points in the monitored structure[160].

6.1.2.4 Smart Garbage Collecting

The smart garbage management system allows cities and businesses (for example, retailers, airports, building managers, etc.) to overcome common problems associated with waste collection and overflow. Such Smart City platform is designed to optimize garbage collection

efficiency. It provides a comprehensive collection of historical data and analytical reports that allow cities and enterprises to make more efficient, data-based decisions. Based on the data received from the garbage container sensors, the proposed cloud system in chapter 3 can create intelligent gathering routes that can be sent to truck drivers every morning. Select the containers they want to pick up, or let the system automatically select them for the most efficient and optimized routes.

In Morocco, there is a mobile application that strongly illustrates the commitment of moroccans in the cleanliness of their cities. Indeed, this application is called "Clean City" and is designed around this action: take pictures of waste. The latter, seemingly innocuous, allows faster cleaning, control of cleaners, and the initiation of Moroccan citizens to the idea of recycling by rewarding their civic actions. "Clean City" is actually available on Android and Apple devices, as well as web version. With the geolocation function, these photos are hosted on the server and viewed by all until they are cleaned. Thus, the act of taking pictures of waste is a real lever of performance of the cleaning services because it increases their reactivity and serves as a real roadmap in their location of waste.

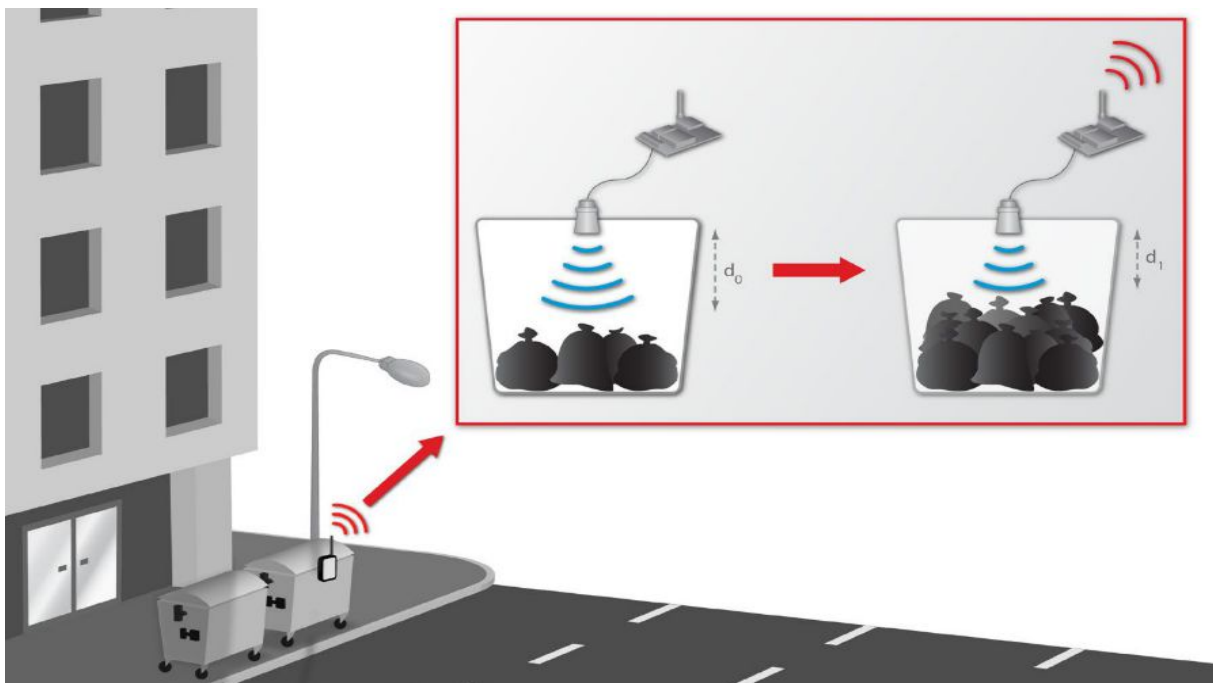


Figure 61 Wireless Sensor Network for Garbage Level Sensing

This application can be an easy and realistically example of the service on demand discussed in chapter 3 but it does not use the wireless sensors, it is just based on reporting with photos towards the competent authorities so that they have to deal with garbage. Ultrasonic sensor is capable of measuring the level of garbage in bins. According to the *Figure 61* above, with WSN-based ultrasonic sensors, the garbage management service will be able to see the actual position of garbage containers in any area and plan routes accordingly in order to remove garbage only from sufficiently full bins, which saves time and fuel. In fact, only about half of the containers really require collection, while the other half is only half full and can expect the next collection cycle.

6.1.2.5 Smart Street Lightning

The street lamp is an integral part of the urban infrastructure, the main function of which is to illuminate the city streets at night. It also has a function for decorating streets and even has some effects for reducing traffic accidents and street crime [161]–[163]. Therefore, it is

important for the public and the governor of the city to ensure the normal switching on and off of street lighting. Initially, street lighting was turned on at dusk and turned off at dawn using manual control, and then the intelligent controller was used to automatically turn on and off street lighting depending on the intensity of lighting at sunrise/sunset, surrounded by the controller, each transformer station. Although an intelligent controller can automatically turn on and off street lighting, cases of street lighting during the day and off during the night often occur due to the cloudy time of the intelligent controller or because the light sensor is covered with dust or something else.

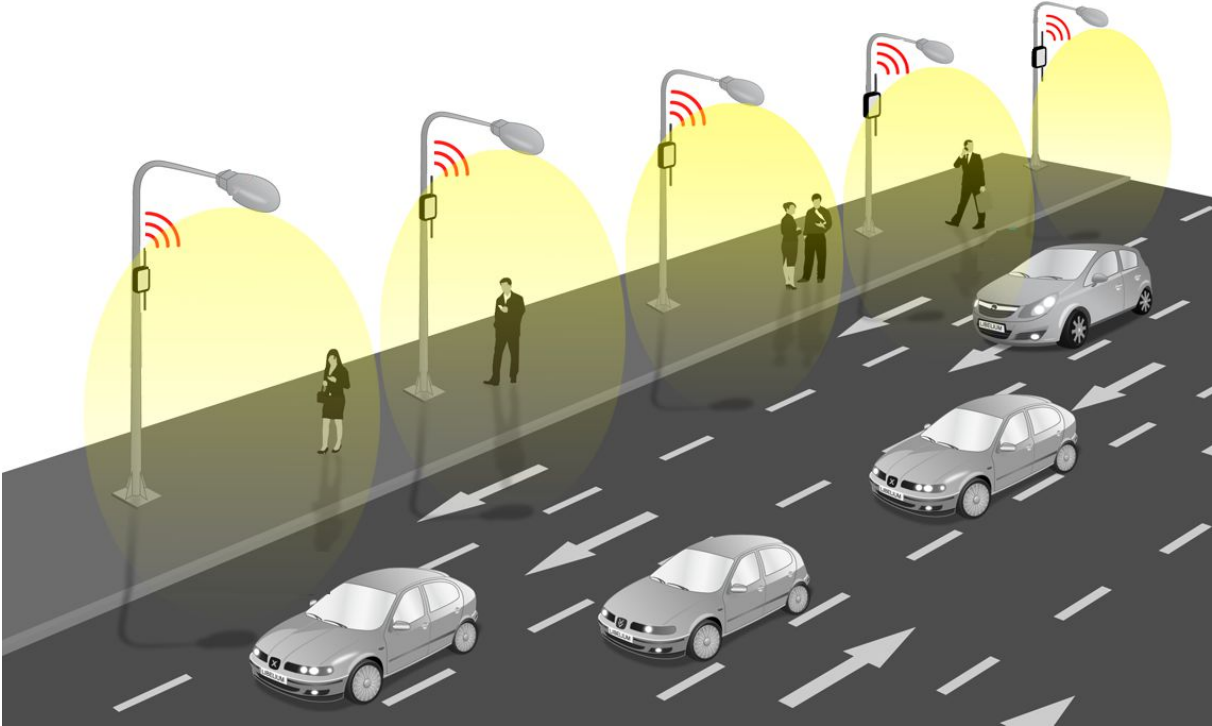


Figure 62 Wireless Sensor Network for Street Lighting

This increases public discontent with unsuitable street lamps and will lead to some potential danger to street vehicles and pedestrians. In order to detect a street lighting malfunction and reduce energy loss, service personnel must patrol the street at night and during the day. This increases the cost of maintenance and management. Since the above reasons and the diversification of street lighting functions, street lighting control using an intelligent controller cannot meet the requirements of street lighting control.

Smart Lighting is one of the key applications of Smart Cities, it allows municipalities to save money and energy by reducing the brightness of lighting during low-traffic hours and increasing safety by illuminating dark areas while people are passing. The brightness sensor can measure accurate information about the intensity of light in the range from 0.1 to 40,000 lux. That allows us to detect sunrise, sunset or small changes in the appearance of clouds and launch actions for intelligent lighting control. Since the emergence of directionable sensor probe, it seems easy to tune these sensors in the desired direction, even pointing straight at the sky, to get the best accurate results.

6.2. Accelerometer-based Wireless Sensor Network for Tilt Sensing

One of the simplest, smallest and cheapest sensors is called “*accelerometer*”, which is widely used in research projects as recognition of army movement [164] where the built system can recognise arm movements with accuracies ranging 85%-96% for healthy subjects and 63%-

75% for stroke survivors involved in 'making- a-cup-of-tea'. Another similar project relate to the human-activity recognition [165] via wearable sensors that provide valuable information regarding an individual's degree of functional ability and lifestyle. Also, eleven movement classes were constructed in order to recognize specific arm movements in stationary positions and also during the movement of the body [166], which can be used for the detection of emergency situations. Free-fall detection is one of the best-known applications of accelerometer sensors, by calculating signal vector magnitude, angle, angular velocity, power, and spectral energy using a Fast Fourier Transform (FFT) technique, elderly people can provide information about an individual's physical movement such as stumbling, forward fall, backward fall, and side fall [167]. Accelerometer data of an Android phone was also used to identify the activities of daily living (sitting, standing, walking, and jogging) with a good accuracy of results that is about 96% on average is achieved in all activities [168].

In addition, Structural Health Monitoring (SHM) is the concept whose main objective is to guarantee the health of the structures (bridges, buildings, ship, aircraft, etc.) to prolong their life, to anticipate their failures and to reinforce their performances. Most of the implemented methods used to monitor the health of these structures rely on the use of sensors including the accelerometer [68], [169]–[171]. Another interesting application of the accelerometer in the WSN domain is the protection of forest trees against poaching [172]; indeed, smuggling of most important trees such as sandalwood in forests, poses a serious threat to forest resources, causes significant economic damage and ultimately has a devastating effect on the environment all over the world.

6.2.1. Designing of Accelerometer-based Wireless Sensor Network

To build a wireless sensor network based on the accelerometer sensor, we consider that each electric transmission or light tower can have a wireless sensor node hanged to its upper top in order to protect it against any physical damage (check the network design on the *Figure 63*). A sensor node must be protected against severe weather like the overwhelming sun, heavy rains, winds, humidity, etc. Accordingly, a protective box is needed in which a sensor node is comfortably fixed inside. Another parameter to consider during the installation of this acceleration sensor, the sensor node must be installed on the pole horizontally so that one of the axes of the accelerometer is oriented vertically in the same direction as the pole.

When the sensor nodes are attached to the light or electrical poles, they start collecting the data regarding the acceleration. Knowing the data collection interval is essential because it affects the battery life of the sensor nodes. An interval of six hours is rational in relation to the importance of this application. For example, an air pollution application would require a smaller collection interval of about 15 minutes because its data is very valuable for improving the air quality which can affect citizens. Thus, the sensed data is sent from one sensor node (pole) to another until the sink node. Often, light or electrical poles are aligned along a road, it is noted that the sensor nodes tend to build a flat architecture (see section 1.4.1) where each sensor node communicates with the sink node through multi-hop communication and its neighbour sensor nodes as relays.

6.2.1.1 Network Kit and Coverage

The success of implementing such an application depends on several factors, mainly the wireless sensor network kit to use. It is important to understand how the WSN platform works especially for the radio modules, which is needed for transmissions and receptions of wireless signals. For example, for the implementation performed as an experiment in this thesis, we used a WSN platform based on sensor nodes called "*Waspnote*". This WSN platform is

manufactured by “*Libellium*”, which its headquarters is located in Zaragoza, Spain. The Wasmote (instead of sensor node elsewhere in this document to refer to the node used during implementation) can be deployed in *Mesh*, *Tree* or *Peer to Peer* topology, knowing the installation of electrical or light poles often as a straight line, undoubtedly we can confirm that the Wasmotes are well suited for such application.

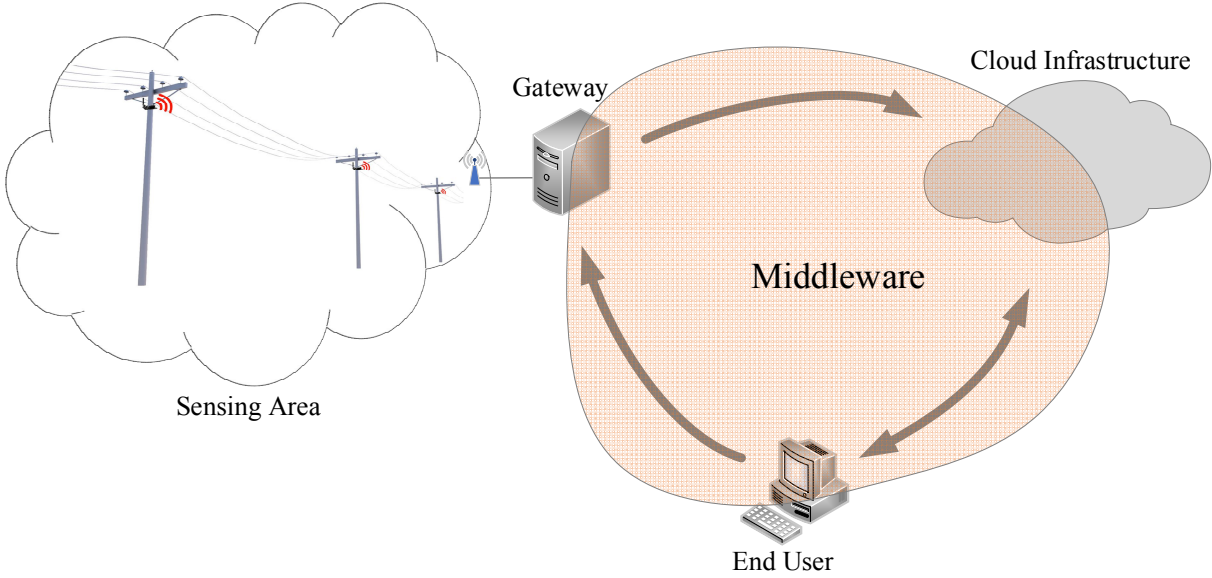


Figure 63 Wireless Sensor Network for Tilt Sensing

As will be discussed later in this document, a Wasmote has a modular architecture, for example a communication module, a sensor module, and so on. As we are currently interested in communication modules, there are several communication modules that integrate seamlessly with a Wasmote. These modules include GSM/GPRS module, WIFI PRO module, XBee module, Bluetooth module, LoRaWAN module, Sigfox module, LoRa module, GPRS + GPS module, 3G module, 4G module, RFID/NFC module, etc. In this thesis, we use the XBee module running ZigBee technology/protocol or the IEEE 802.15.4 standard. The theoretical range of Wasmote’s XBee module varies from 750 to 15500 m depending on radio version[173]. Therefore, knowing that the distance between two poles does not exceed 100 m that to suggest that the XBee module range can cover at least seven times more than required distance. The theoretical range of Wasmote is provided for maximum transmission power, however, when implemented it would be necessary to adjust on the transmission power while reducing the range and saving energy in order to cover a short distance of about 200 to 300 m.

6.2.1.2 Power Optimization and Latency

Note that for this implementation the sensor nodes are fixed on the poles, therefore we do not need to consider the mobility of the sensor nodes. On the other hand, it is essential to optimize the transmission energy by orienting the sensor nodes so that the directivity of its antenna is directed towards a neighbour sensor node. Another solution would be to mount solar panels with the sensor nodes to charge the batteries during the day in the presence of sunlight. Wasmote has a socket for solar panel of 5V and 10mA, in absence of this solar panel it is charged by USB. As we often see with solar power super bright LED on traffic signs, a solar panel charges their batteries during the day and its batteries feed them when the sky gets black. In the same way, the Wasmotes could make itself autonomous as regards energy. Another solution would be to use the electric current already available in the poles. This approach would probably be more expensive than the use of the battery and solar panels because the cables running through the electrical and light poles for domestic use are about 220V and the use of

this electric current on a sensor node would require a transformer under voltage to adapt to Waspmotes.

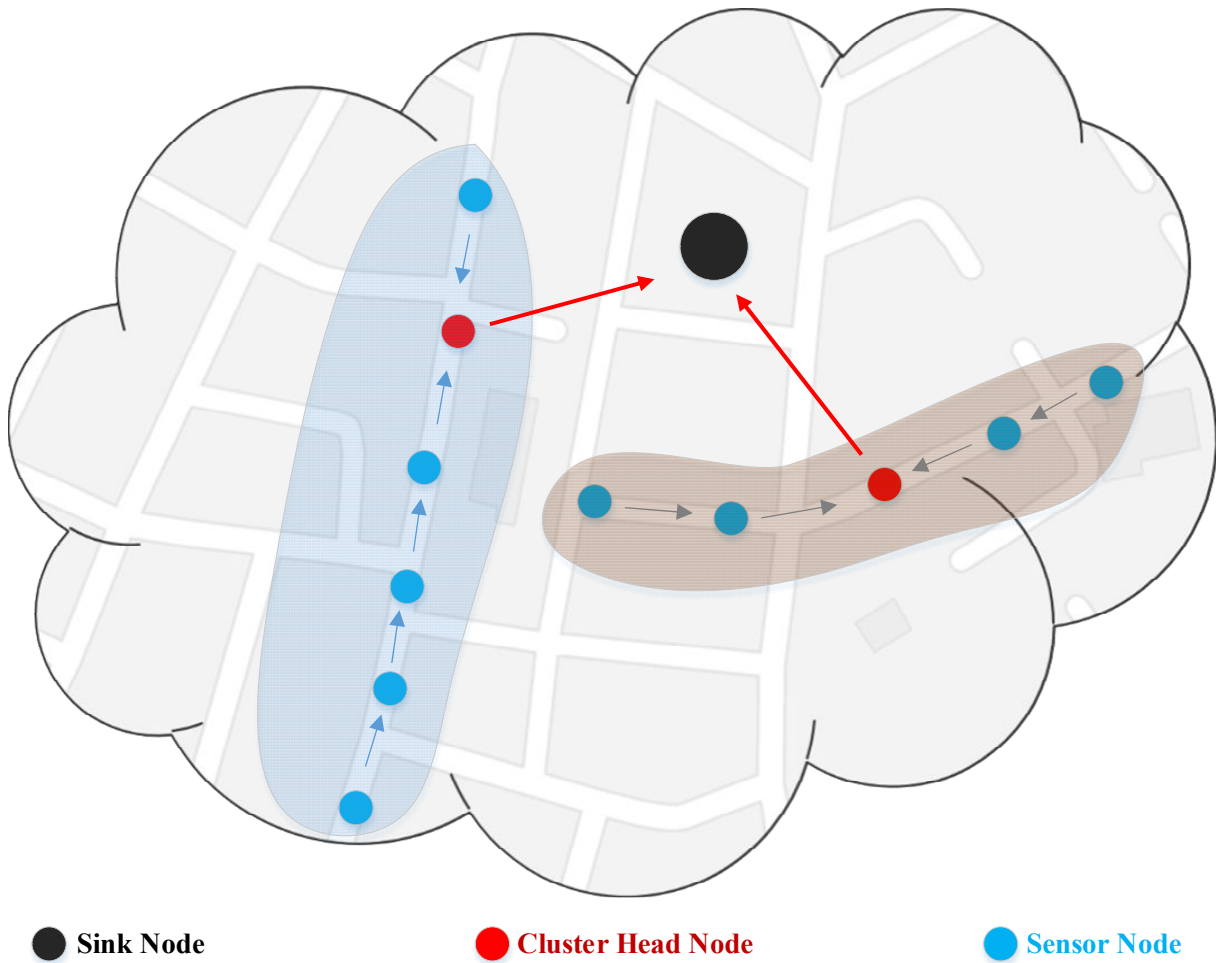


Figure 64 Introducing clustering into accelerometer-based wireless sensor network

As previously mentioned, sensor nodes communicate in multi-hop mode to reach the sink node, but as the number of electrical or light poles increases, the time needed for a packet to reach the sink node becomes larger. This latency does not seem too annoying for such an application but it remains a challenge for this architecture based on electrical and light poles. In fact, the challenge does not come directly from latency, but excessive transmissions from the intermediate sensor nodes. Let's take an example of an illuminated road over a length of 1 km, if we implement this network of wireless sensors on its light poles whose average distance between two poles is about 30 m, it will take at least 33 sensor nodes. The nodes at the end of the road must use the intermediate nodes as relays in order to reach the sink nodes. These intermediate nodes use energy to collect and send their own data, and in addition to forward data from remote nodes, these intermediate nodes may exhaust their energy early, so the network may be cut off. Therefore, it would be inadvisable to build a very long chain network, but small networks in the network. Here in this document, we propose to cut the network into clusters as shown on the *Figure 64* above. Each portion of the road constitutes a cluster and for each cluster a sensor node is designated as a cluster head whose role is to gather the sensed data coming from the sensor nodes of the same portion of the road in order to transmit them to the sink node. This cluster head does not perform aggregation, because it does not make sense to aggregate the acceleration data from the different light poles. On the other hand, if two sensors are fixed on the same light pole, then these data can be aggregated.

6.2.2. Hardware and Calculation of the Tilt Angle

Waspnote PRO v1.2 is the Libelium's advanced open source wireless sensor platform specially focused on the implementation of low consumption modes to allow the sensor nodes ("motes") to be completely autonomous and battery powered, offering a variable lifetime between 1 and 5 years depending on the duty cycle and the radio used. *Waspnote* hardware architecture has been specially designed to be extremely low consumption. Digital switches allow turning on and off any of the sensor interfaces as well as the radio modules. Two different sleep modes make *Waspnote* the lowest consumption sensor platform in the market:

- The consumption in Deep Sleep mode is $55\mu\text{A}$. Sensors may generate an interruption to wake the main microcontroller up when the value read goes above or below a pre-programmed threshold. The device is completely slept and the sensors powered. Values of the sensor thresholds are controlled by software via digital potentiometers.
- In Hibernate, the consumption is only $0.7\mu\text{A}$. All systems are switched off to ensure minimum consumption. *Waspnote* will be woken up by an alarm from the internal clock. Using this mode, lifetime of each node may vary from 1 to 5 years depending on the duty cycle and the battery capacity. Although the lifetime can be extended indefinitely connecting a solar panel in the dedicated socket on the board.

Waspnote is based on a modular architecture. The modules available for integration in *Waspnote* are categorized in ZigBee/802.15.4 modules (2.4GHz, 868MHz, 900MHz), GSM/GPRS Module (Quad band: 850MHz/900MHz/1800MHz/ 1900MHz), GPS Module, Sensor Modules (Sensor boards) and Storage Module (SD Memory Card). *Waspnote* Core Board is device with connectors for all modules aforementioned. The idea is to integrate only the modules needed in each Core Board device; these modules can be changed and expanded according to needs.

Each *Waspnote* PRO v1.2 (different to *Waspnote* v1.1 and *Waspnote* v1.5) Core Board has a built in acceleration sensor LIS331DLH from STMicroelectronics which informs the mote of acceleration variations experienced on each one of the 3 axes X, Y and Z (shown on the *Figure 65* and *Figure 66*). The LIS331DLH sensor is defined as a 3-axis linear accelerometer, which allows measurements to be taken from the sensor through the I2C interface. The accelerometer has a 12 bit resolution (4096 possible values) and includes 3 measurement ranges $\pm 2g$, $\pm 4g$ and $\pm 8g$ ($g \approx 9.81 \text{ m/s}^2$).

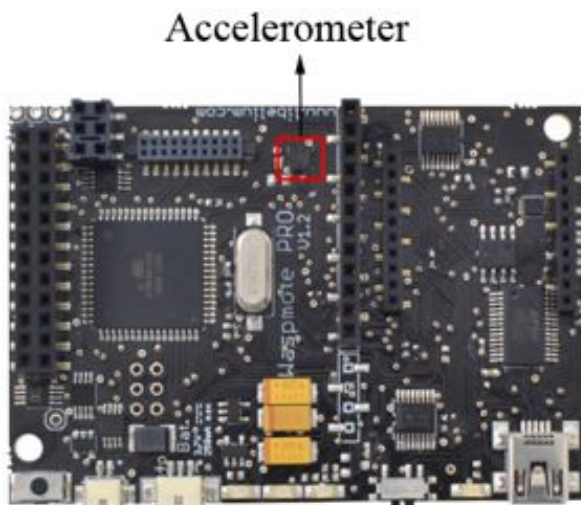


Figure 65 Accelerometer Sensor embedded in *Waspnote* (PRO 1.2) Core Board

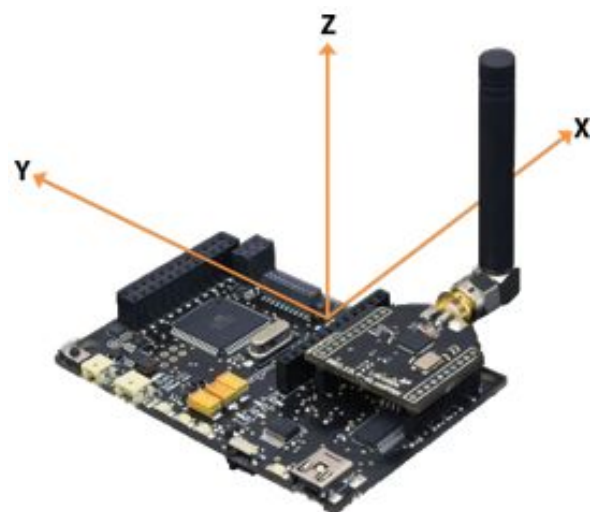


Figure 66 Accelerometer Axes' Orientation in *Waspnote* (PRO 1.2) Core Board, communication module plugged in)

Table 7 Calculations of the accuracy of the reading and the acceleration resolution depending on the range

Range	$\pm 2g$	$\pm 4g$	$\pm 8g$
Accuracy of the reading	$4096/4g = 1024 \text{ LSB/g}$	$4096/8g = 512 \text{ LSB/g}$	$4096/16g = 256 \text{ LSB/g}$
	$= \frac{4g}{2^{12}}$	$= \frac{8g}{2^{12}}$	$= \frac{16g}{2^{12}}$
Acceleration resolution	$= 0.0009765625 \times g$	$= 0.001953125 \times g$	$= 0.00390625 \times g$
	$= 0.0009765625 \times 9,81 \text{ m/s}^2$	$= 0.001953125 \times 9,81 \text{ m/s}^2$	$= 0.00390625 \times 9,81 \text{ m/s}^2$
	$= 0.009580078125 \text{ m/s}^2$	$= 0.01916015625 \text{ m/s}^2$	$= 0.0383203125 \text{ m/s}^2$

Assume that v is the value of any component sensed by the accelerometer; the acceleration of the relevant component can be calculated using the following formula:

$$\text{Acceleration} = v \times \text{Acceleration resolution} \quad \text{where } v \in [-2048, 2048]$$

– For the first range $\pm 2g$, we calculate the linear acceleration as following:

$$\text{Acceleration} = v \times 0.009580078125 \text{ m/s}^2 \quad (2)$$

– And for the first range $\pm 4g$, the linear acceleration is obtained by replacing the corresponding acceleration resolution as following:





$$\text{Acceleration} = v \times 0.01916015625 \text{ m/s}^2 \quad (3)$$

– In the case of using the third range $\pm 8g$, the linear acceleration would be:

$$\text{Acceleration} = v \times 0.0383203125 \text{ m/s}^2 \quad (4)$$

By using the $\pm 2g$ range, theoretical values collected by the Waspnote at the vertical position (shown in *Figure 69*) are near to (0, 0, 1024). By replacing these components in the formula (1), we note that the linear acceleration along the X and Y-axes are zero and the acceleration along the Z-axis is around $1g = 9.81 \text{ m/s}^2$ (Corresponds to the gravity). When tilting the Waspnote around its Y-axis from 0° to 90° so that the Y-axis is directed upward and perpendicular to the ground, the components of the accelerometer vary from (0, 0, 1024) to (-1024, 0, 0). By rotating again from 90° to 180° , the collected values change from (-1024, 0, 0) to (0, 0, -1024). Pivoting finally Waspnote with a total angle of 270° around the Y-axis, the values acquired by the sensor nodes are equal approximately equal to (1024, 0, 0). All results of this experiment are summarized in the next *Table 8*.

Table 8 Summary of the experiment about rotating Waspnote core board around the Y-axis

Observation	Angle (radian)	Rotation axis	X value	Y value	Z value
	$0, 2\pi$	Y-axis	0	0	1024
	$\frac{\pi}{2}$	Y-axis	-1024	0	0
	π	Y-axis	0	0	-1024
	$\frac{3\pi}{2}$	Y-axis	1024	0	0

If the Waspnote is rotated around another axis with the same angles shown in the *Table 8*, it supposed to get always the sensed values spaced by 1024. At each rotation, we note that if the

angle variation is about 90° , then the y and z -component absolute variation is 1024. As Waspnote's accelerometer is linear, we can deduce that the offset angle is proportional to the component of the accelerometer. Assume that $\Delta v = 1024$ and $\Delta \tau = \pi/2$, then the tilt angle τ between the Waspnote at vertical position and each axis when it is tilted can be calculated as below:

$$\tau = v \Delta \tau / \Delta v \quad (5)$$

Or v is any component of the accelerometer i.e. v can be x , y or z , thus we can also calculate α as following:

$$\alpha = z \Delta \tau / \Delta v, \alpha \in [-\pi/2, +\pi/2] \quad (6)$$

It is obvious that the formula (5) does not change when changing the range of the accelerometer. By cons, the values of v and Δv also change when changing range.

6.2.3. Software and Tilt Angle Algorithms

6.2.3.1 Data Sensing

The Waspnote sensor node's architecture is based on the Atmel ATmega1281 microcontroller and programmed in C++. The structure of the codes (the code is saved as ".pde" file) is divided into 2 basic functions: *setup()* and *loop()*. The *setup()* is the first part of the code, which is only run once when the code is initialized (or Waspnote is reset). In this function, it is recommendable to include the initialization of the modules, which are going to be used, as well as the part of the code, which is only important when Waspnote is started. On the other hand, the *loop()* function runs continuously, forming an infinite loop. The goal of the *loop()* function is to sense, send the collected data and save energy by entering a low consumption state. The Waspnote API is divided into two folders: "core" and "libraries". The libraries inside "core" (for example "WaspACC.h", library for accelerometer sensor) are invoked automatically so there is no need to add them to the ".pde". However, it is mandatory to manually include to the ".pde" a library which is inside the "libraries" folder (if we need to use it).

When Waspnote is connected and starts the bootloader, there is a waiting time (62.5 ms) before beginning the first instruction; this time is used to start loading new compiled programs updates. If a new program is received from the USB during this time, it will be loaded into the FLASH memory (128 kB) replacing already existing programs. Otherwise, if a new program is not received, the last program stored in the memory will start running. When Waspnote is reset or switched on, the code starts again from the setup function and then the loop function. By default, variable values declared in the code and modified in execution will be lost when a reset occurs or there is no battery. To store values permanently, it is necessary to use the microcontroller's EEPROM (4 kB) non-volatile memory. EEPROM addresses from 0 to 1023 are used by Waspnote to save important data, so they must not be over-written. Thus, the available storage addresses go from 1024 to 4095. Another option is to use of the high capacity 2 GB SD card.

For this tilt sensing approach, *setup()* function activates the accelerometer sensor, initializes global variables such as gateway MAC address and extracts the Waspnote MAC address as the node identifier to be sent with the collected data because it is essential to know the identity of the node where the data come from. Unlike the *setup()* function, as aforementioned, the *loop()* function is executed several times in an infinite loop until the battery is dead in order to collect the data every time the Waspnote is switched in awake mode. After sensing data, a ZigBee 802 frame is created, the acquired data is added to the frame, which is sent to the MAC address of the sink node. After submitting of the frame, a green LED is blinked if the frame has been

successfully sent to the next hop or a red LED in case of failure. After that, the Wasmote is switched in sleep mode and wakes up after the loop period (five seconds for the experiment carried out in this chapter, see the section 6.3) and restarts the process. The *Figure 67* is the flowchart of the source code written in C ++ to run on the Wasmote core board in order to collect and send acceleration data to the sink node.

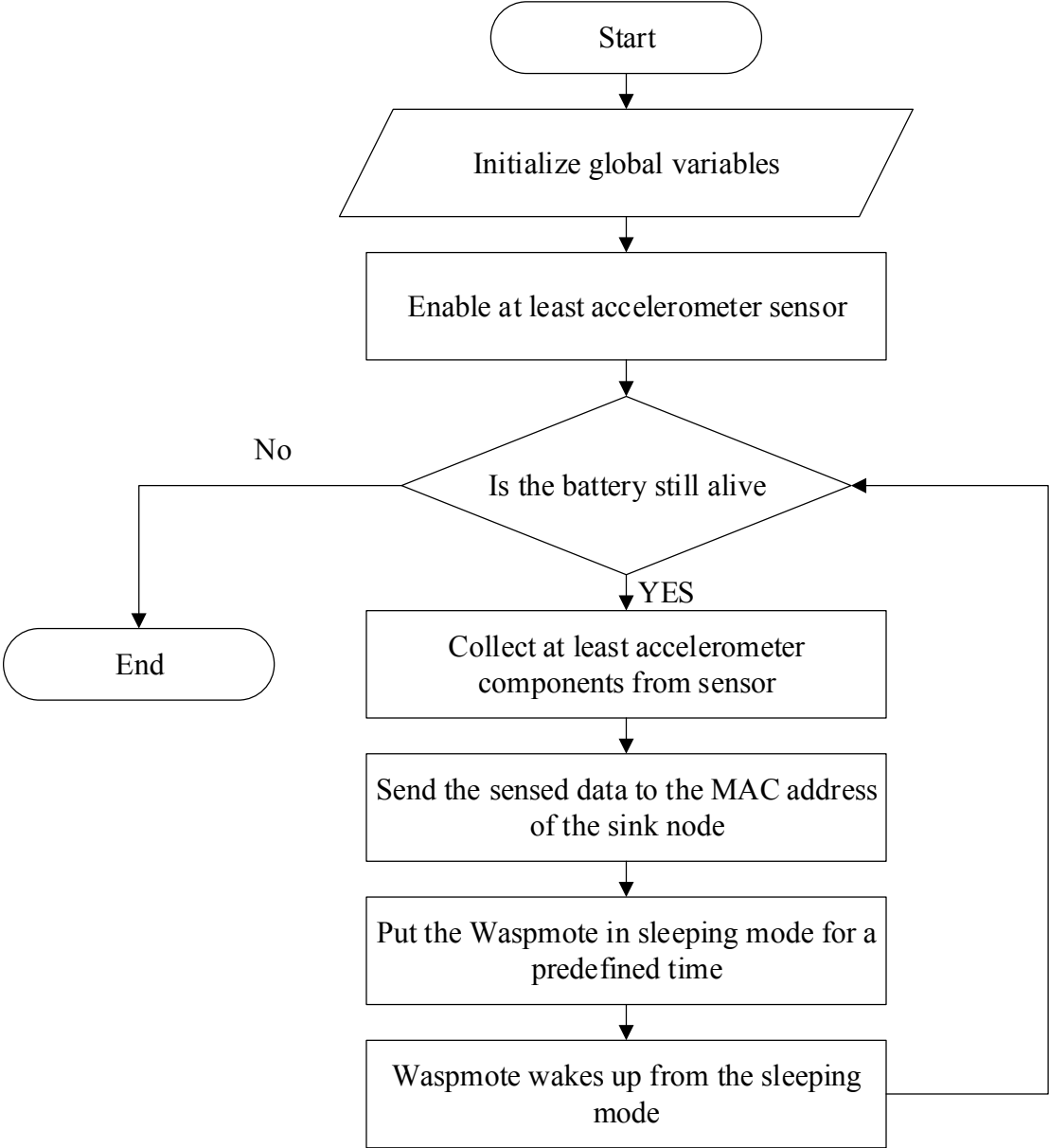


Figure 67 Flowchart of the accelerometer’s data sensing

6.2.3.2 Data Acquisition and Parsing

The gateway runs a distributed application with Java RMI, which connects the WSN to the End User monitoring application through the Cloud Infrastructure. This monitoring application is running on end user machine located on the external network and interconnected by a middleware as described in the section 0. The gateway application uses a Java extension library called RXTX and the native serial driver to be integrated with the operating system that allows accessing the serial ports (USB) of the computer where the WSN sink node is connected. When the application is launched, the first step is about to select the serial port on which the gateway is connected, then this port is initialized in the application and an event listener on this port is

registered to allow the observation of the data coming through this port. A frame sent by the Wasmote XBee module eventually arrives at the gateway where it is directly routed via the serial port of the computer, at this moment the event listener immediately reports the arrival of unencrypted data (no cryptography has been used for this implementation subject to accelerate the experiment) whose format is as follow:

#N1#I#MAC: 0013A200414E20E6#ACC: 53;-116; 1023#

Each row arrives at time and it has four splits delimited by “#”. From left to right, we first receive the name of the sender node (N1), then the sequence number of the frame (1, 2 and 3), the mac address (0013A200414E20E6) of the XBee module plugged in the sender node, and finally all components of the accelerometer in this order: X; Y; Z.

At that moment, these raw data are parsed as string token through the input of the serial port instance and they are filled to a new instance of object. Then the gateway application retrieves the instance of the cloud object distributed with Java RMI and executes the *saveData()* method for sending as parameter the object instantiated before. The gateway application waits again until the arrival of another frame and restarts the process (the flowchart of this algorithm has been showed early in this document, check the *Figure 44*).

6.2.3.3 Data Processing

The data processing consists of calculating the tilt angle based on the accelerometer components and displaying it on the GUI of the end user. To implement this process programmatically, when the object containing data arrives at the end user’s monitoring application, based to the formula (6) the acceleration’s components are used to calculate the corresponding angle as plotting the electric pole in 3D scene displayed as part of the graphical user interface (GUI). 3D scene is plotted by using the JAVA programming with OpenGL or simply JOGL API. This API is designed to provide hardware-supported 3D graphics to applications written in Java and integrates with the AWT, Swing and SWT widget sets, as well as with custom windowing toolkits using the Native Window API.

6.3. Experimental Results

The experiment was carried out in the laboratory, we had a small pole on which we had fixed a sensor node. The goal of this experiment is to tilt the pole to collect the accelerometer information needed to calculate the tilt angle. When the pole was tilted, the corresponding angle with the geometric tools was measured to make a comparison between the sensed angle and the measured angle. Experienced angles are not taken randomly, they vary from 15 degrees downward from 90 ° to 0 °, i.e. from the vertical position to the horizontal position as shown on the *Figure 69*. The obtained results are summarized in the following *Table 9*:

Table 9 Tilt sensing experiment results showing sensed and measured angles

N°	x	y	z	Sensed angle	Measured angle	Difference
1	655	-271	1005	88.3	90.0	1.7
2	648	-569	872	76.6	75.0	1.6
3	643	-790	685	60.2	60.0	0.2
4	639	-899	526	46.2	45.0	1.2
5	628	-959	361	31.7	30.0	1.7
6	619	-1016	164	14.4	15.0	0.6
7	607	-1021	0	0.0	0.0	0.0

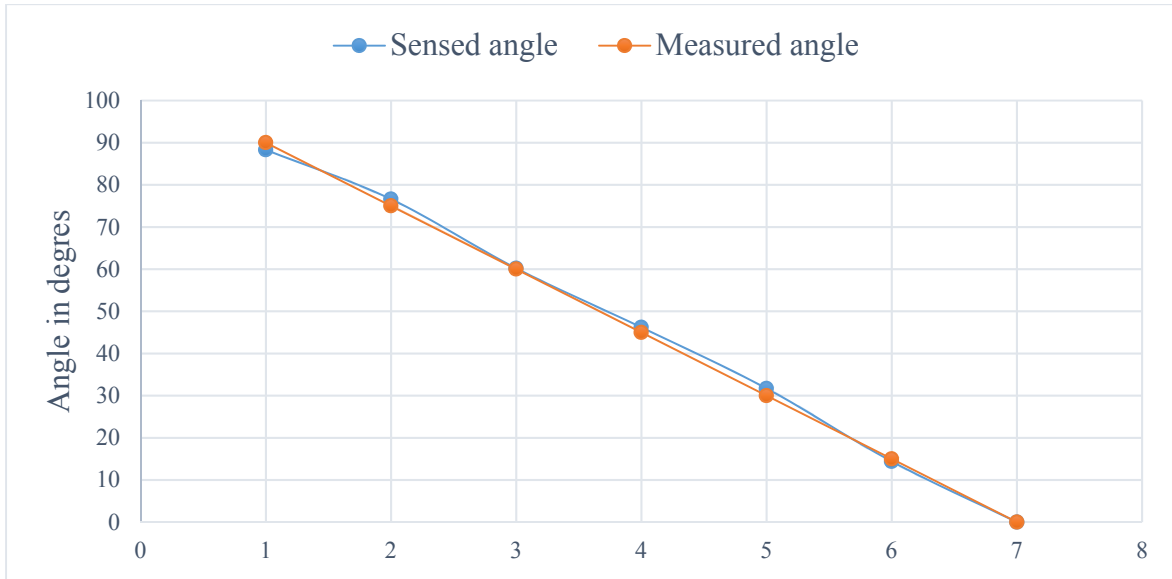


Figure 68 Graph comparing the sensed angle and the measured angle

From the *Table 9*, we notice that there is a discrepancy between the sensed angle and the measured angle, using this difference it possible to evaluate this tilt sensing system in terms of error that can be committed. This error can be calculated by averaging the absolute values of the difference between the sensed and measured angle, which is equal to 1.0° .

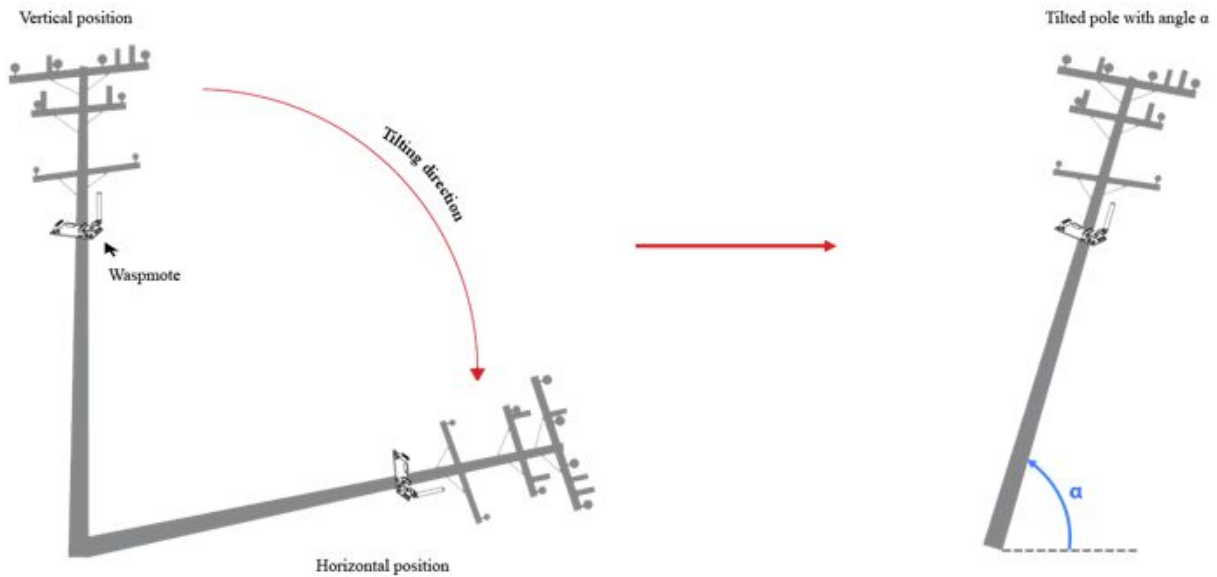


Figure 69 Tilt sensing experiment's configuration

For each angle, the end user application should display the sensed angle (obtained by applying the formula (6) to the collected data) as information as well as the components of the acceleration that allowed this angle to be calculated. The implementation also includes the 3D graphical interface to reproduce the tilted pole. During the 3D implementation, we noted that JOGL's axes are not oriented in the same way as the Waspnote accelerometer, the Y-axis and the Z-axis are interchanged; therefore, the z-component of the accelerometer was plotted at the Y-axis and the y-component at the Z-axis.

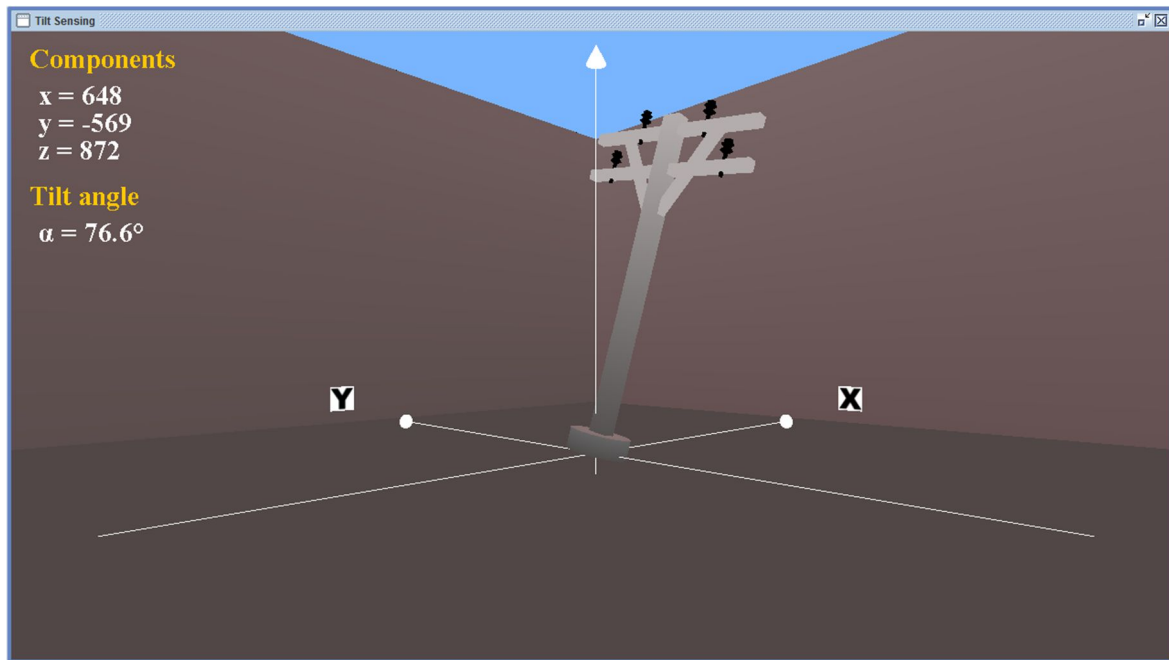


Figure 70 Screenshot of the end user application reproducing the 3D scene of the 75° tilted tower
(See all screenshots in Appendix A)

6.4. Discussion and Conclusion

The difference between the results obtained by calculation and by measurement can be at the origin of the various factors such as the sensitivity of the accelerometer, the range of “g” ($\pm 2g$, $\pm 4g$ or $\pm 8g$) used, the tools of measurements, etc. Despite this difference, this system is adequate for use that does not require the peak of precision as in Structure Health Monitoring. For the use in electrical poles monitoring, the error of a 1.0° is acceptable because from the graph in figure *Figure 68*, it appears that the two curves are arranged almost in the same way, to explain why the calculated angle is very close to the measured angle.

According to formula (6), it is possible to have negative results, hence a negative angle, this explains why the z-component of the accelerometer is negative. According to the fixing position of the accelerometer on the electric pole in the *Figure 69*, the z-component may be negative in the case where the pole falls on an inclined plane, for example if the pole is on a mountain so that the Z-axis of the accelerometer is directed towards the ground.

The 3D graphics on the monitoring screen reproducing the inclination of the pole do not show sufficient details to know in which direction the electrical pole is tilted (e.g. towards North or East). Under real conditions we could use the x and y-components of the accelerometer to determine with certainty the direction of inclination, for example, by installing the Waspnote on the poles so that the X-axis is directed to the North (Obviously, the Y-axis is fixed to the West).

In this implementation, the proposed approach is applicable to the electric poles monitoring by using a network of wireless sensors based on the accelerometer. The results obtained in the laboratory correspond to better conditions for this tilt sensing system. An elaborated and advanced study may be necessary if the system is to be deployed in the real smart cities where environmental conditions are increasingly harsh such as wind, vibrations, the force exerted by the cables connecting the pole to its neighbours and anything else that could alter the results. Another advantage of this application is that it is easier to integrate a sensor board into the Waspnote core board with sensors dedicated to sampling the electricity voltage values, current, magnetic field, etc

Conclusion and Future Work

I. Summary of Contributions

In this thesis, we studied the problem of integration of wireless sensor networks with the cloud computing. Such a network has a certain interest in environmental monitoring applications. To achieve this goal, we rely on the use of a middleware, whose wireless sensor network communicates with the cloud through a gateway, which is considered as one of the components of the same distributed system as the cloud and as end users.

We started by proposing a physical architecture with four layers namely Sensing Area (SA), the Gateway (GW), the Cloud Infrastructure (CI) and the End Users (EU). These four layers allowed us to divide the system into small components that can be handled more easily. Despite these four layers, the middleware communication sees only three logical layers namely the Gateway (GW), the Cloud Infrastructure (CI) and the End Users (EU). It is through the gateway of logical architecture that we reach the fourth layer (SA) of the physical architecture. In order to get WSN data to end users through the cloud, we first thought of doing a study to find a powerful middleware for WSN applications that often require speed for real-time monitoring. This study allowed us to distinguish Java RMI as the ideal middleware among others (CORBA and web services). Note that this comparative study between middleware was performed on the basis of invocation time (time required to invoke a remote method or function on distributed object) of each middleware and the middleware that consumes less network resources. The results obtained in this study are similar to those of other studies made by other authors, which allowed us to validate the choice of Java RMI as middleware for the interconnection of system components.

It is obviously important to study in detail such a network in order to design a configuration that can above all satisfy the requirements of the application but also make its operation efficient and optimize its financial cost. Given the heterogeneity of the networks (the WSN, the internet and the cloud) involved in this thesis, it is not possible to simulate the entire architecture on WSN traditional simulators such as the NS2 or NS3 because these simulators do not include all the components of the proposed architecture. For this, we used a few wireless sensor nodes in the rest of the work.

In order to reduce the heavy traffic of the data collected in the WSN, one of the techniques used is to aggregate the data. Aggregation reduces the number of packets to the sink node while saving network energy. However, aggregating the data can have a negative effect on the encrypted data. When the data to be aggregated are encrypted, they must first be decrypted, then aggregated and finally encrypt the result of the aggregation. Another contribution of this thesis is about to use a homomorphic encryption algorithm that allows the aggregation process to perform the aggregation functions on the data directly encrypted without decrypting them, and to return the result encrypted. However, it should be noted that not all homomorphic encryption algorithms are suitable for WSN because some use calculations that require so much computational power that the sensor nodes cannot support. So, in this thesis we have posed assumptions in order to move in the right way to choose a practical algorithm for sensor networks. We have found that homomorphic encryption by using elliptic curves is a perfect solution for WSNs because they do not use exponentiation calculations and they perform additive homomorphism that is useful for aggregation functions.

Unfortunately, elliptic curves are not practical while mapping the data (values) on the elliptic curve, we found that it is possible to encrypt data with elliptic curves but it is not easy to find the original data because it will be necessary to perform discrete logarithms. As a result, we descended into the algorithm preference hierarchy to choose an implementable algorithm on the sensor nodes. We noticed that Domingo-Ferrer homomorphic encryption is implementable on the sensor nodes but that we need to find the perfect encryption parameters.

Homonymous Domingo-Ferrer encryption requires that the message to be encrypted as an integer must be split into different pieces. We have found that the number of pieces has a not significant impact on the condition of the battery. For 10,000 messages sent using 2, 3 and 4 pieces we noticed that the battery level dropped by 6%, 7% and 8% respectively. We must admit that the more we increase the number of pieces of the initial message, the more we increase the level of security higher. In this case, we only have to conclude that the encryption parameters for the Domingo-Ferrer algorithm are in compromise with the type of the sensor nodes (hardware, available resources) and the desired level of security.

In order to involve the application in our research, we have found a way to apply what has been proposed previously to monitor smart cities assets. Using our logical architecture, we were able to transmit data from the WSN to the end user across the cloud, but how the data can be useful to the end user? Therefore, the proposed application makes it possible to transform the data into useful information. The acceleration data has been exploited and processed to provide a tilt angle of an electrical tower on which is fixed a sensor node collecting acceleration in three dimensions. The proposed application makes it possible to display on the screen the angle of inclination and at the same time reproduces the 3D scene of the tilted electric tower. Thanks to this application, we were able to validate the feasibility of our research.

II. Future Work

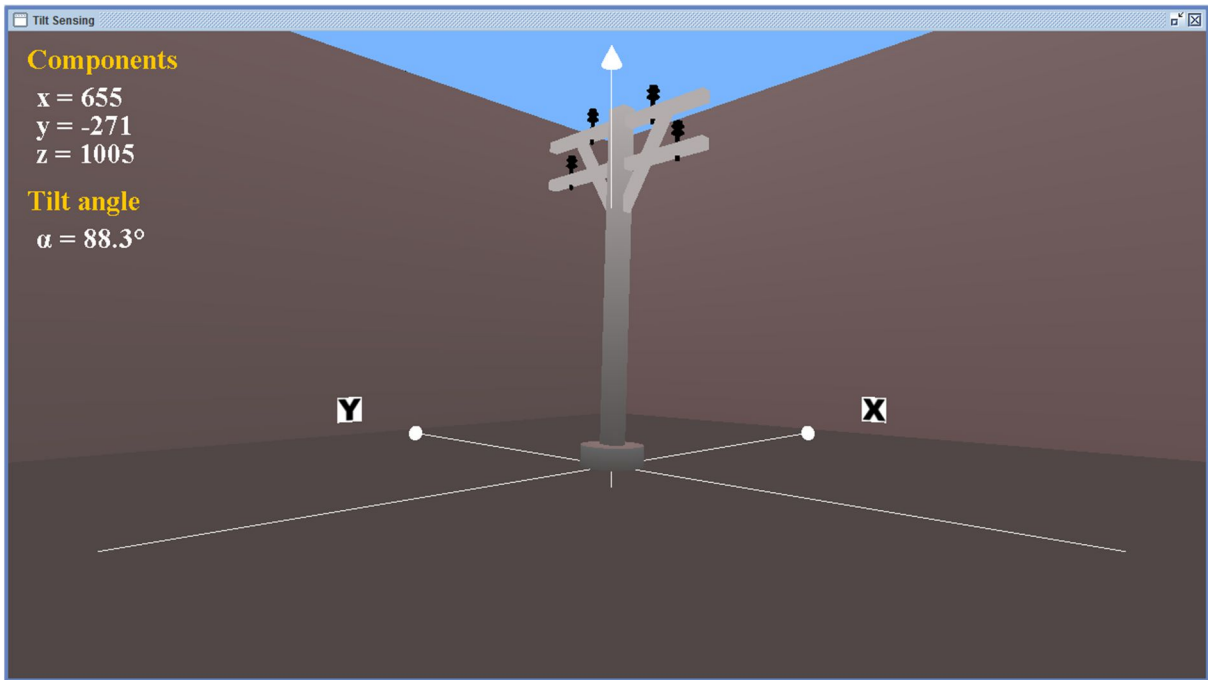
Since the work done in this thesis has been able to highlight the applicability of a quasi-optimal solution that can secure the data from the WSN using an additive homomorphic encryption algorithm, another research direction would be interesting to study in the future. It is to exploit the multiplicative homomorphism of Domingo-Ferrer's encryption scheme. By combining the additive and multiplicative properties of this homomorphic cryptosystem, we will be able to reach an advanced level of data processing statistically.

Despite the successful implementation of the Domingo-Ferrer homomorphic encryption, elliptic curves cryptography is so far the best solution for the data privacy of cloud integration with the wireless sensor network. In the future research, we will strive to find particular elliptic curves that do not lead to the problem of the reverse mapping function.

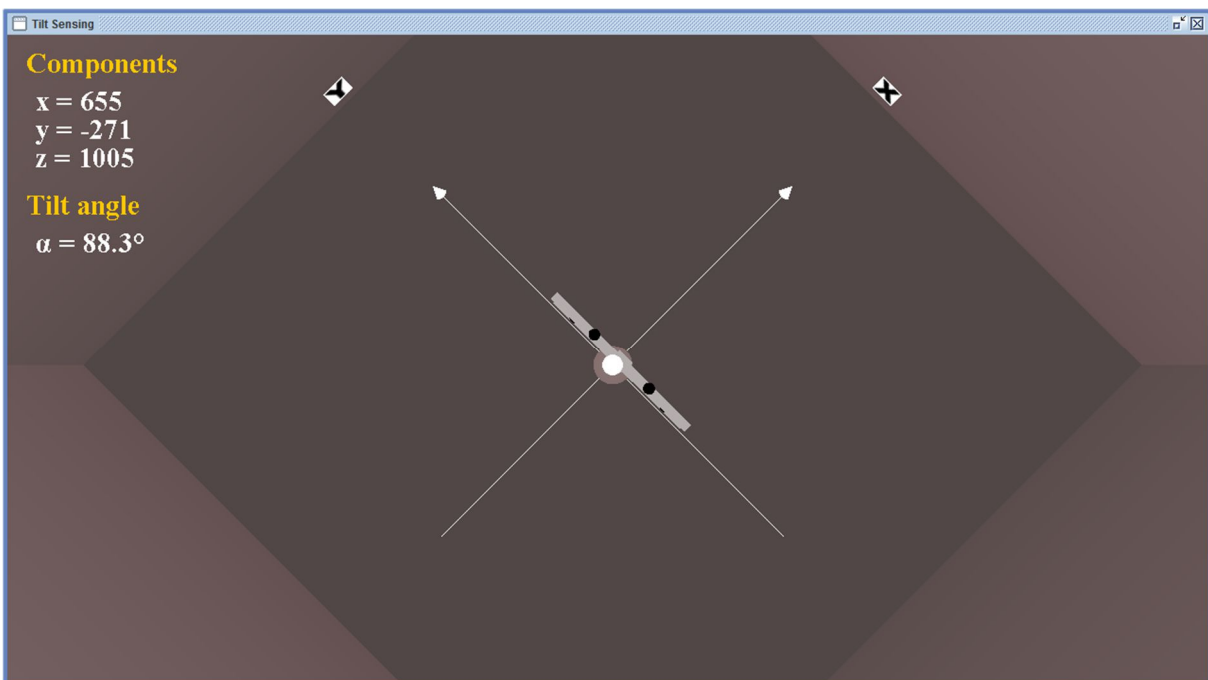
Appendix A

Results of the Tilt Sensing Experiment in Java 3D

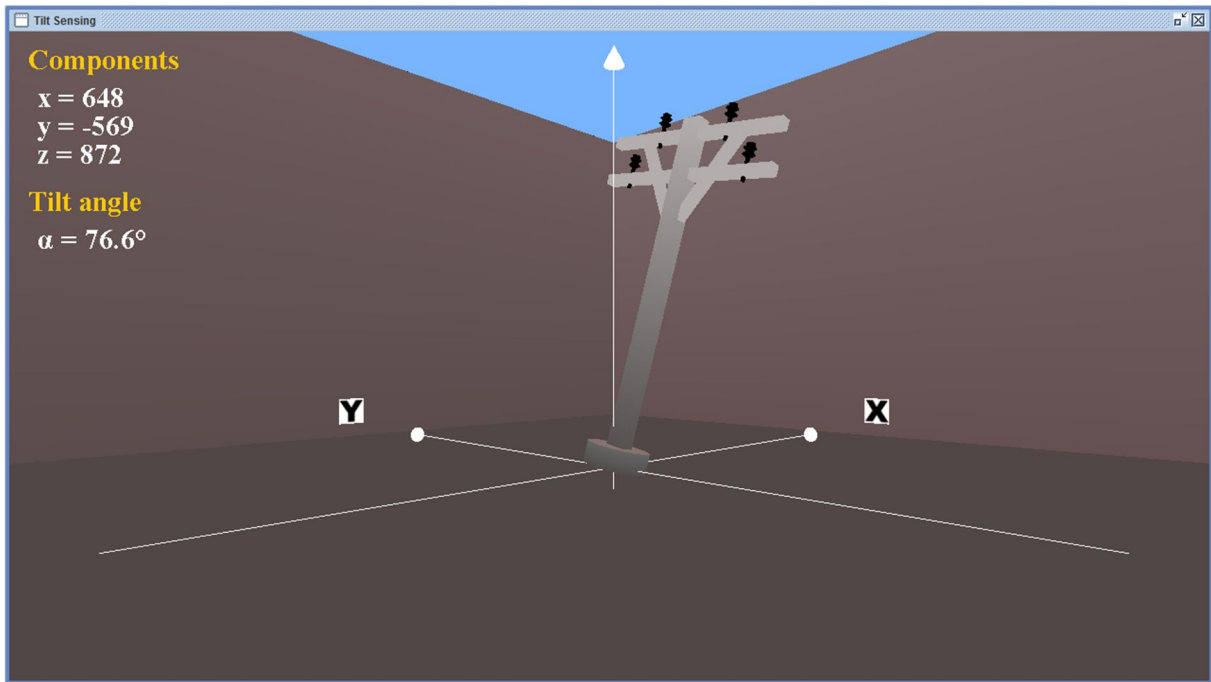
Front view, measured angle = 90°



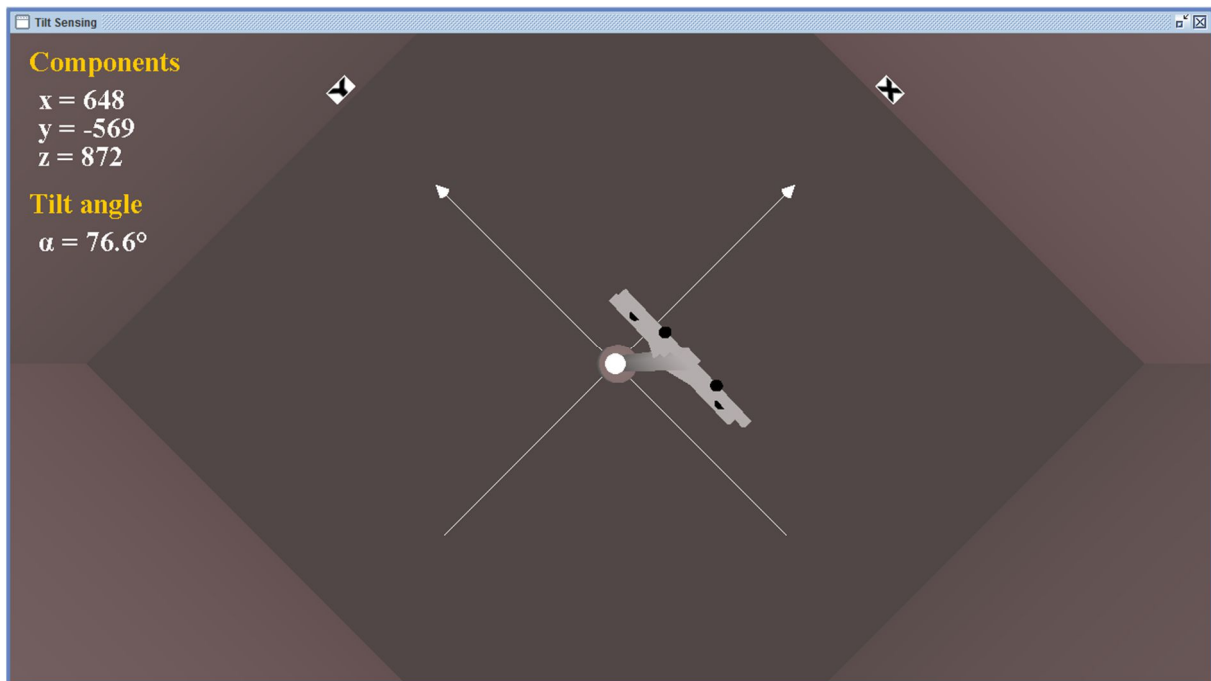
Air view, measured angle = 90°



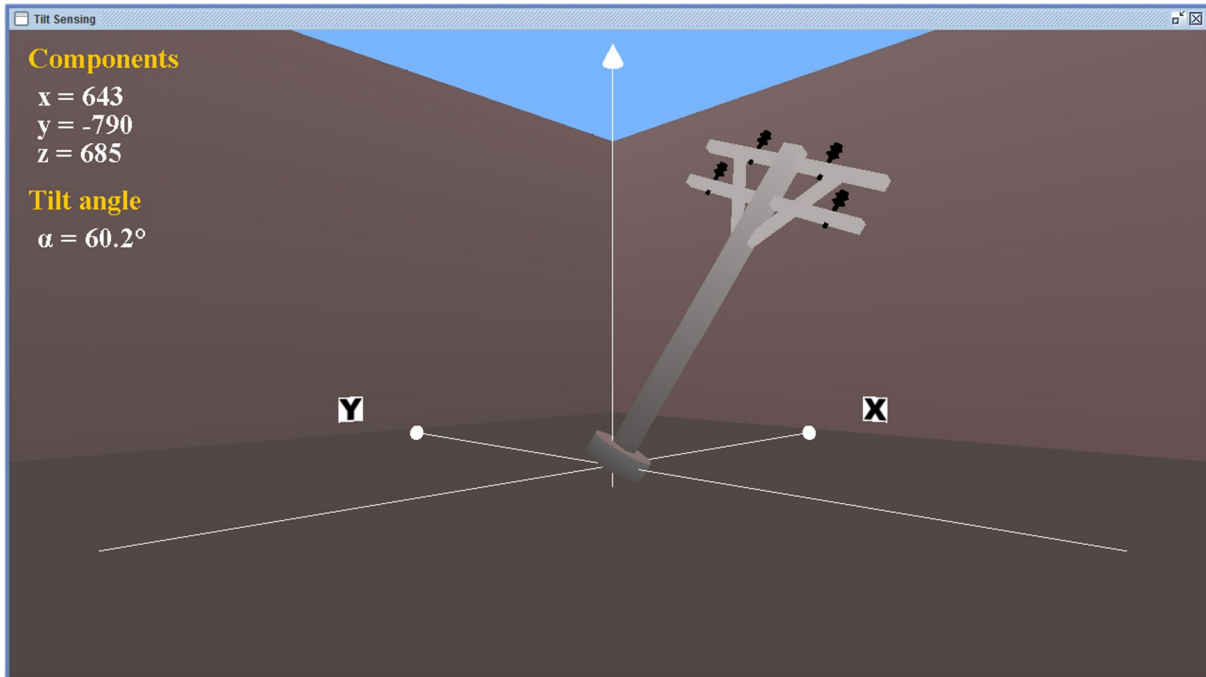
Front view, measured angle = 75°



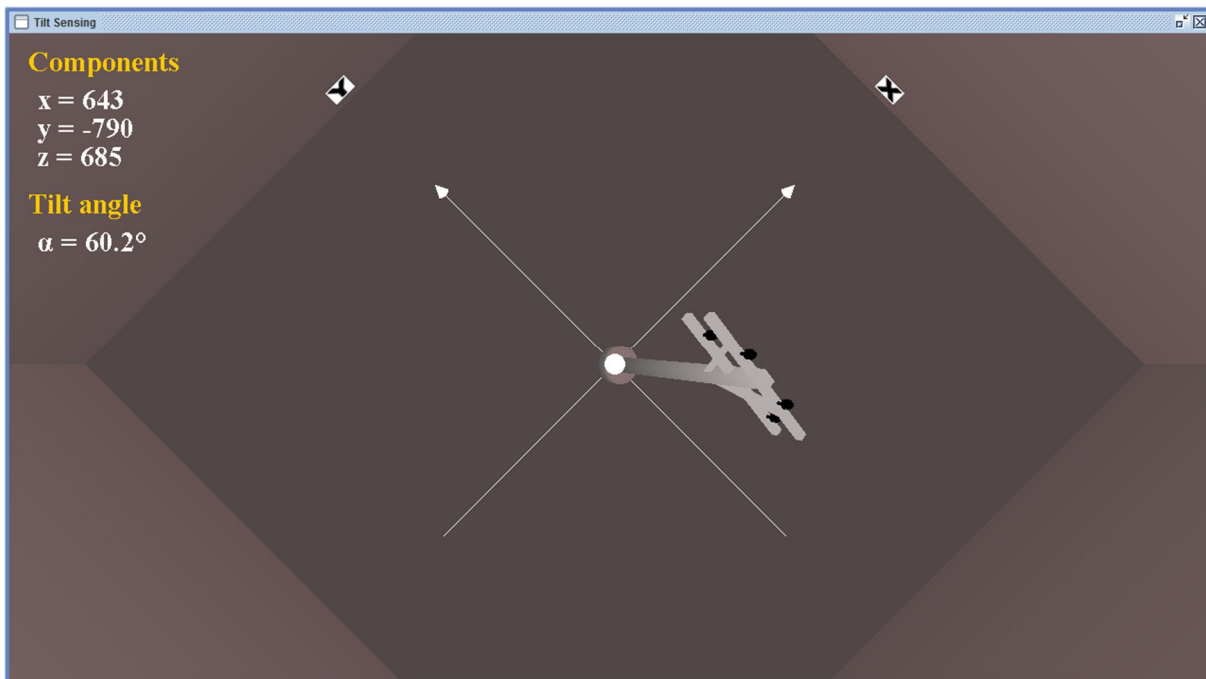
Air view, measured angle = 75°



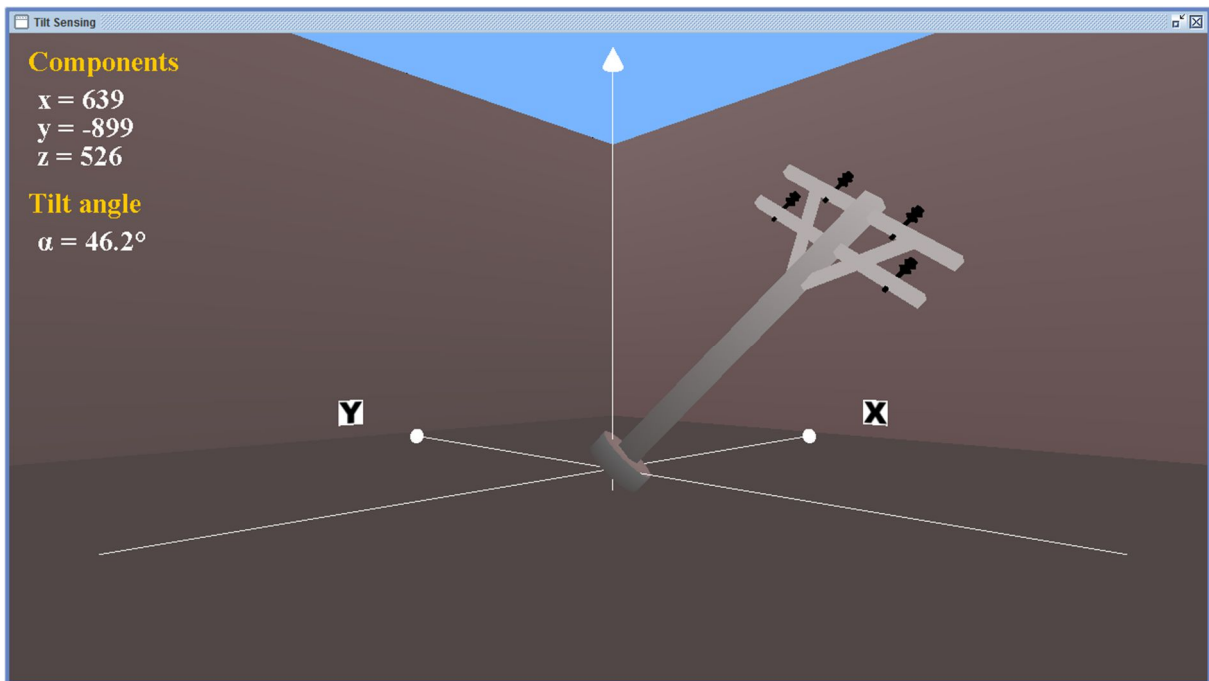
Front view, measured angle = 60°



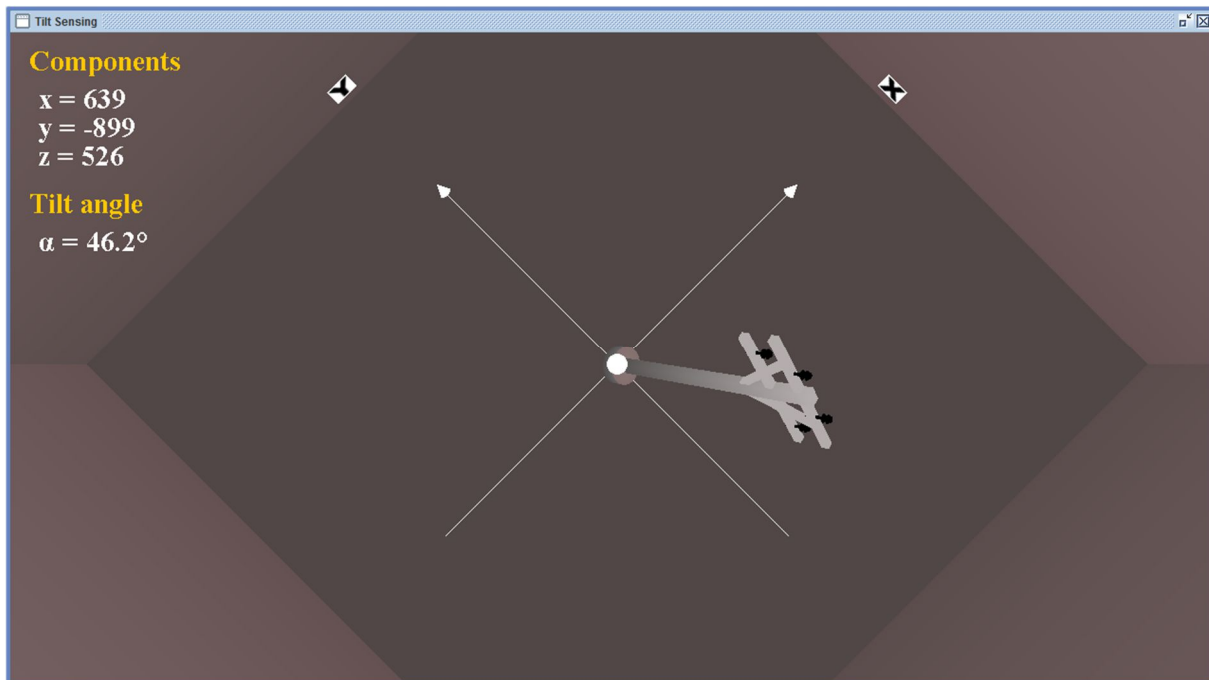
Air view, measured angle = 60°



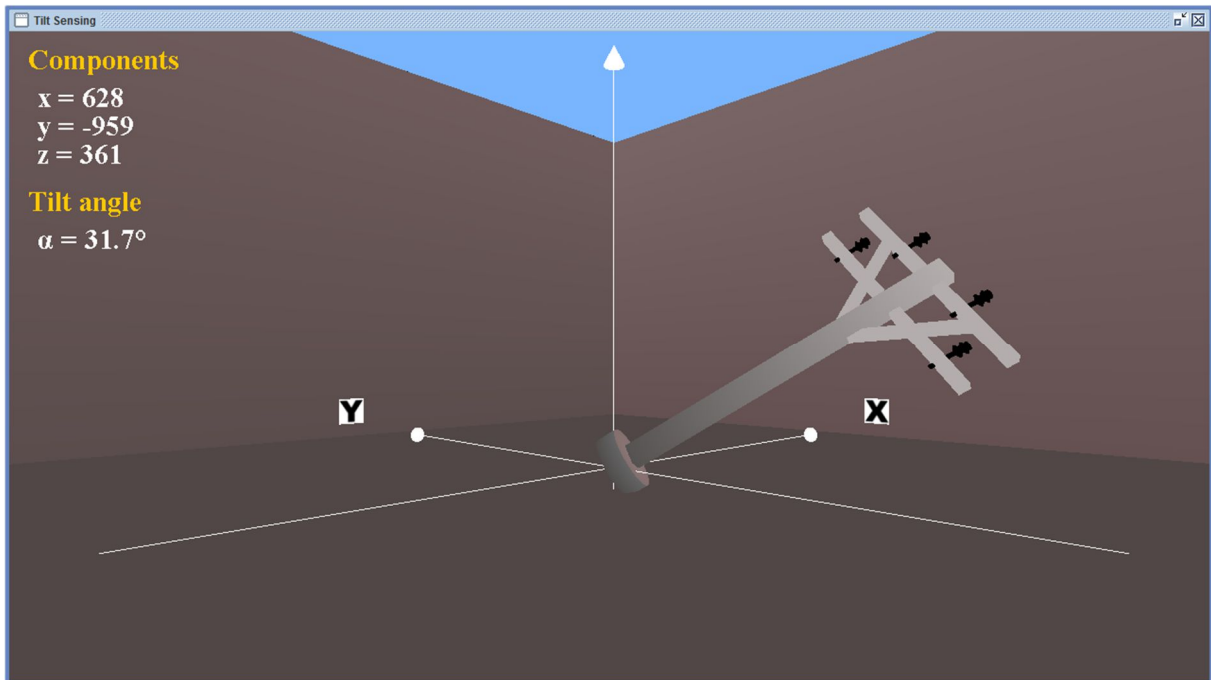
Front view, measured angle = 45°



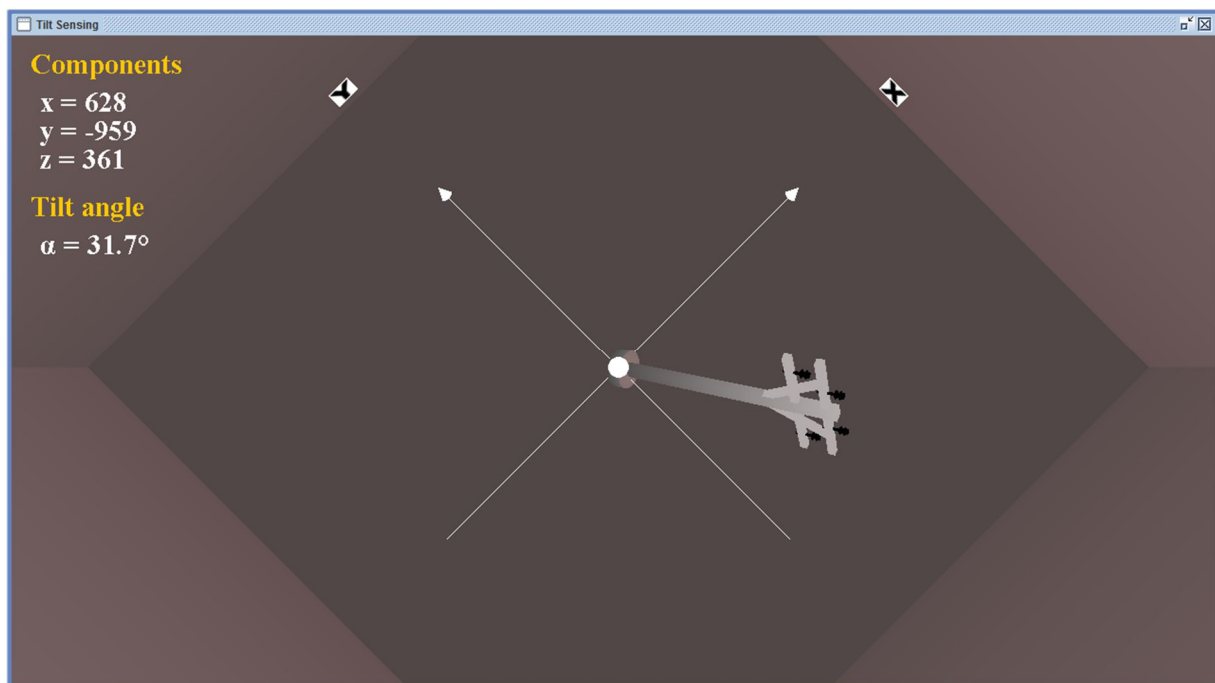
Air view, measured angle = 45°



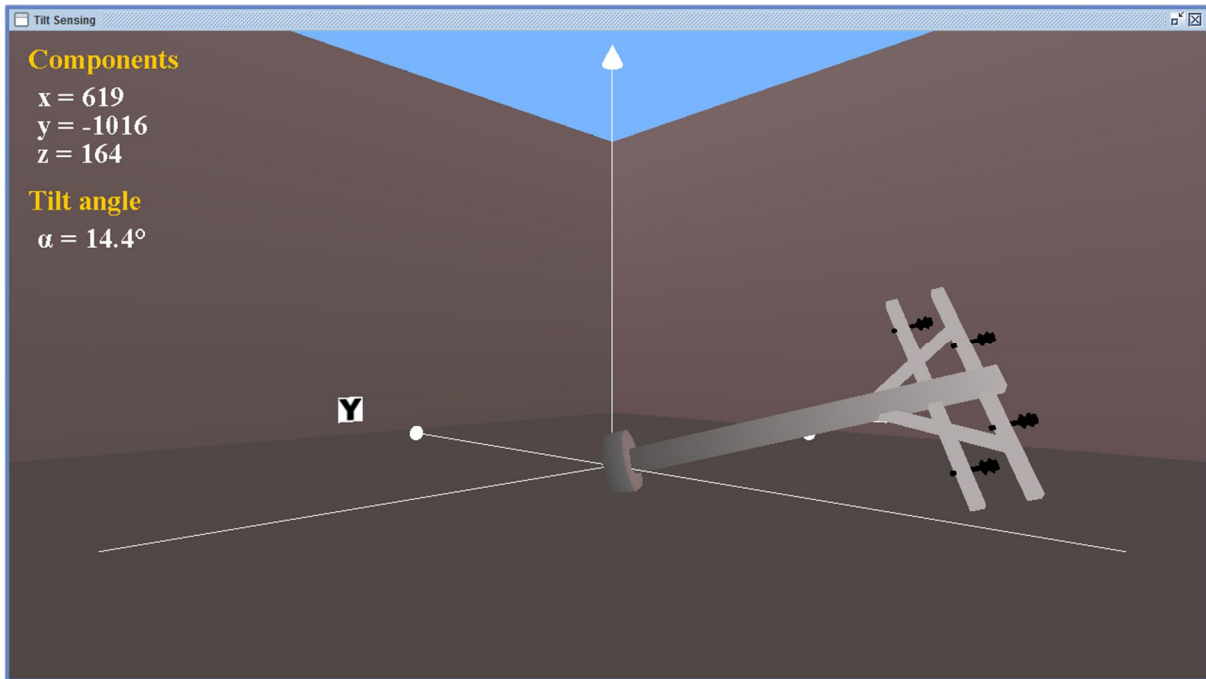
Front view, measured angle = 30°



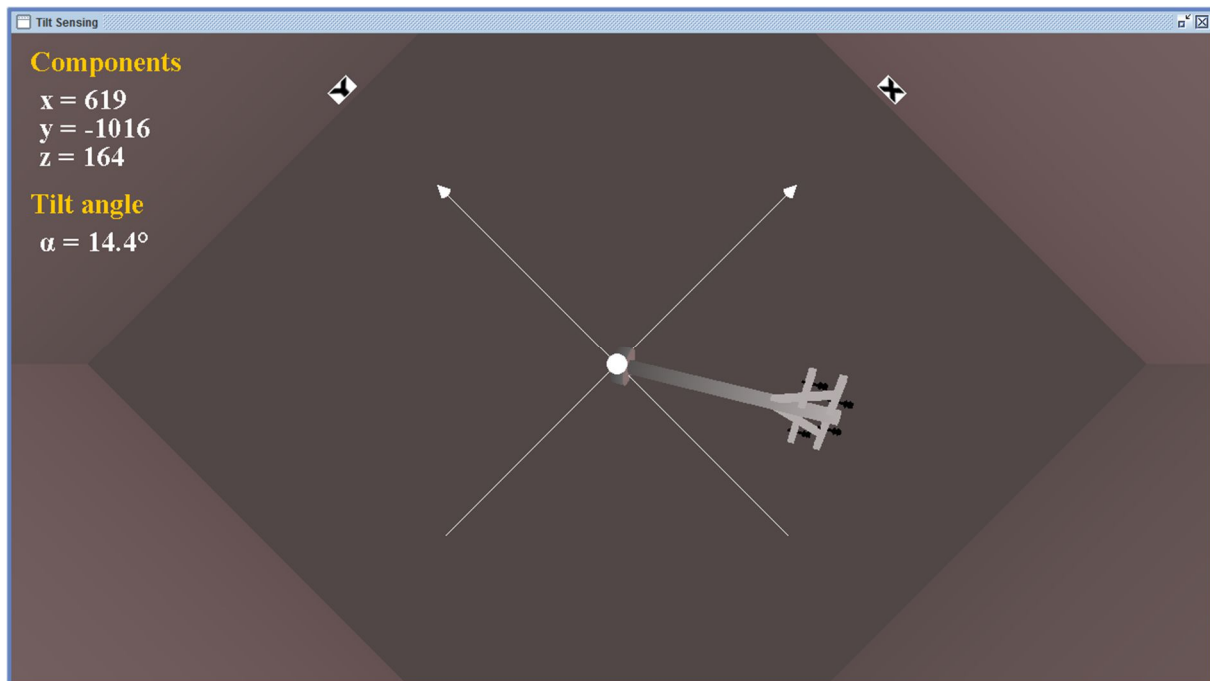
Air view, measured angle = 30°



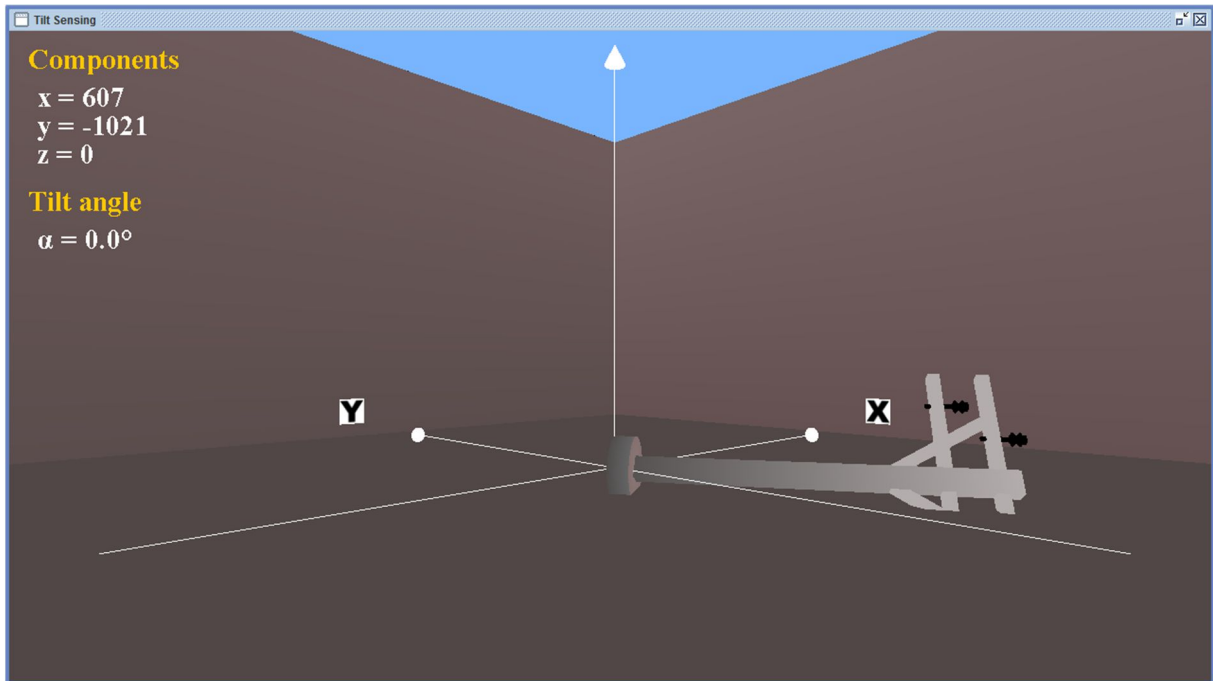
Front view, measured angle = 15°



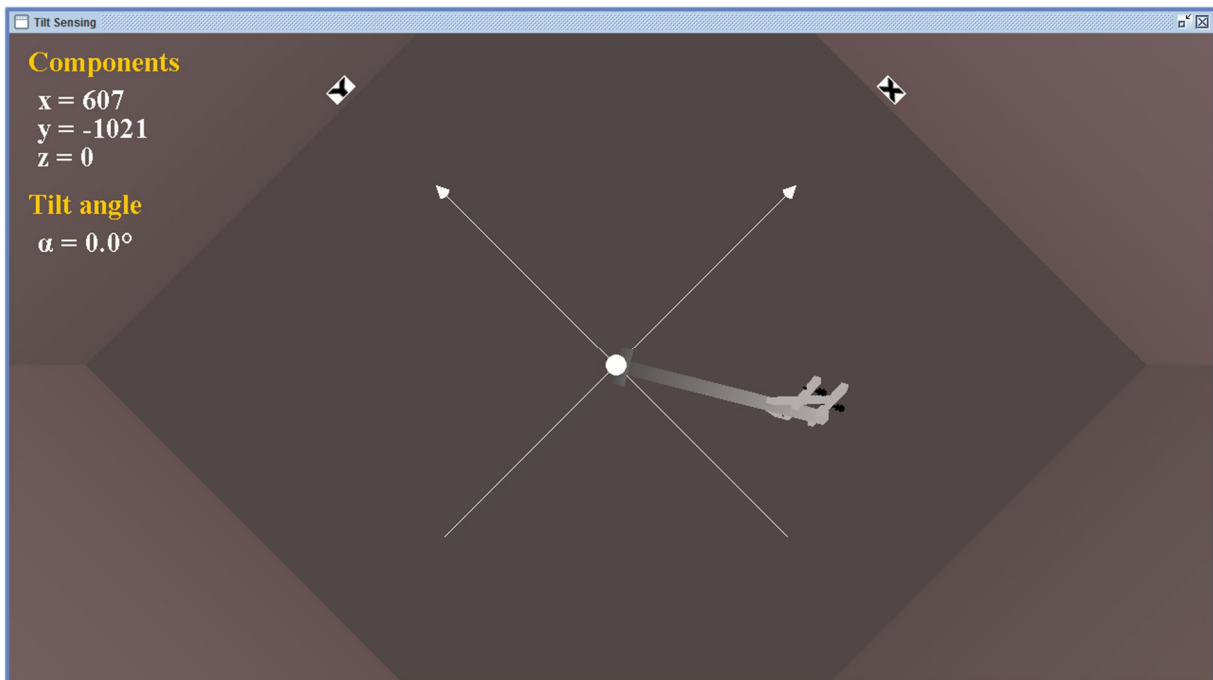
Air view, measured angle = 15°



Front view, measured angle = 0°



Front view, measured angle = 0°



Appendix B

Scientific Activities

I. Regional and International Publications

1. Ngabo, C. I., & Beqqali, O. El. (2019). **Implementation of Homomorphic Encryption for Wireless Sensor Networks Integrated with Cloud Infrastructure**. Journal of Computer Science, 15(2), 235–248. <https://doi.org/10.3844/jcssp.2019.249.257>.
2. Ngabo, C. I., & El Beqqali, O. (2018). **3D tilt sensing by using accelerometer-based wireless sensor networks: Real case study: Application in the smart cities**. In 2018 International Conference on Intelligent Systems and Computer Vision (ISCV) (pp. 1–8). Fez, Morocco: IEEE. <https://doi.org/10.1109/ISACV.2018.8354013>.
3. Ngabo, C. I., & El Beqqali, O. (2016). **Real-time Lighting Poles Monitoring by using Wireless Sensor Networks Applied to the Smart Cities**. In ACM International Conference Proceeding Series (Vol. Part F1263). New York, NY, USA: ACM. <https://doi.org/10.1145/3010089.3010097>.
4. Ngabo, C. I., & El Beqqali, O. (2015). **Distributed System based on Cloud Computing with Wireless Sensor Networks**. Mediterranean Telecommunication Journal, 5(June), 79–86. Retrieved from <http://revues.imist.ma/?journal=RMT&page=article&op=view&path%5B%5D=3096>.

II. International Conferences

1. Oral presentation entitled “**3D tilt sensing by using accelerometer-based wireless sensor networks: Real case study: Application in the smart cities**” at the International Conference on Intelligent Systems and Computer Vision (ISCV 2018). Date: 02-04th April 2018, Venue: Faculty of Medicine and Pharmacy, Sidi Mohamed Ben Abdellah University of Fez, Morocco.
2. Oral presentation entitled “**Real-time Lighting Poles Monitoring by using Wireless Sensor Networks Applied to the Smart Cities**” at the International Conference BDAW “Big Data and Advanced Wireless technologies”. Date: 10-11th November 2016, Venue: American University in Bulgaria. Blagoevgrad, Bulgaria.
3. Oral presentation entitled “**Distributed System based on Cloud Computing with Wireless Sensor Networks**” at the International Conference WITS “Wireless Technologies, embedded an intelligent Systems”. Date: 29-30th April 2015, Venue: Faculty of Science and Techniques, Sidi Mohamed Ben Abdellah University of Fez, Morocco.

REFERENCES

- [1] J. Zheng and A. Jamalipour, *WIRELESS SENSOR NETWORKS - A Networking Perspective*, vol. 2008. Hoboken, New Jersey, USA: John Wiley & Sons, Inc., 2009.
- [2] K. Sohraby, D. Minoli, and T. Znati, *Wireless Sensor Networks*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2007.
- [3] M. Armbrust *et al.*, “A view of cloud computing,” *Commun. ACM*, vol. 53, no. 4, p. 50, 2010.
- [4] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, “A break in the clouds,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, p. 50, 2008.
- [5] A. Khan, A. Zugenmaier, D. Jurca, and W. Kellerer, “Network virtualization: a hypervisor for the Internet?,” *IEEE Commun. Mag.*, vol. 50, no. 1, pp. 136–143, Jan. 2012.
- [6] M. Michal, “Base Station for Wireless sensor network,” MASARYKOVA UNIVERZITA, 2013.
- [7] R. Rajagopalan and P. K. Varshney, “DATA-AGGREGATION PROTOCOLS BASED ON NETWORK ARCHITECTURE,” *IEEE Commun. Surv. Tutorials*, vol. 8, no. 4, pp. 48–63, 2006.
- [8] M. C. Castro, P. Dely, J. Karlsson, and A. Kassler, “Capacity Increase for Voice over IP Traffic through Packet Aggregation in Wireless Multihop Mesh Networks,” in *FGCN '07 Proceedings of the Future Generation Communication and Networking*, 2007, pp. 350–355.
- [9] L. Krishnamachari, D. Estrin, and S. Wicker, “The impact of data aggregation in wireless sensor networks,” in *Proceedings 22nd International Conference on Distributed Computing Systems Workshops*, 2002, pp. 575–578.
- [10] M. Malboubi, L. Wang, C. N. Chuah, and P. Sharma, “Intelligent SDN based traffic (de)Aggregation and Measurement Paradigm (iSTAMP),” in *Proceedings - IEEE INFOCOM*, 2014, pp. 934–942.
- [11] M. Thangaraj and P. P. Ponmalar, “A Survey on data aggregation techniques in wireless sensor networks,” *Int. J. Mob. Netw. Des. Innov.*, vol. 1, no. 3, pp. 36–42, 2011.
- [12] Y. Emhemmad, M. Youssef, and R. Yadav, “Survey on Several Secure Data Aggregation Schemes in WSN,” *Int. J. Curr. Eng. Technol.*, vol. 6, no. 4, pp. 1154–1159, 2016.
- [13] V. Pandey and A. Kaur, “A Review on Data Aggregation Techniques in Wireless Sensor Network,” *J. Electron. Electr. Eng.*, vol. 1, no. 2, pp. 1–8, 2010.

- [14] I. S. Akila, S.V. Manisekaran, and R. Venkatesan, “Modern Clustering Techniques in Wireless Sensor Networks,” in *Wireless Sensor Networks - Insights and Innovations*, IntechOpen, 2017, pp. 141–156.
- [15] A. V. Sisal and S. Khiani, “Data Aggregation Techniques in Wireless Sensor Network : Survey,” *nternational J. Comput. Appl.*, pp. 12–14, 2014.
- [16] F. Dobsław, M. Gidlund, and T. Zhang, “Challenges for the Use of Data Aggregation in Industrial Wireless Sensor Networks,” in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, 2015, pp. 138–144.
- [17] N. . Labraoui, M. . Guerroui, M. . Aliouat, and T. . Zia, “Data aggregation security challenge in wireless sensor networks: A survey,” *Ad-Hoc Sens. Wirel. Networks*, vol. 12, no. 3–4, pp. 295–324, 2011.
- [18] M Healy, T Newe, and E Lewis, “Efficiently securing data on a wireless sensor network,” *J. Phys. Conf. Ser. 76 012063*, vol. 76, no. 1, 2007.
- [19] G. Guimaraes, E. Souto, D. Sadok, and J. Kelner, “Evaluation of Security Mechanisms in Wireless Sensor Networks,” in *2005 Systems Communications (ICW’05)*, 2005, pp. 428–433.
- [20] J. P. Walters and Z. Liang, “Chapter 17 Wireless Sensor Network Security : A Survey,” pp. 1–50, 2006.
- [21] M. A. . Khan, G. A. . Shah, and M. . Sher, “Challenges for security in Wireless sensor networks (WSNs),” *World Acad. Sci. Eng. Technol.*, vol. 80, no. 8, pp. 390–396, 2011.
- [22] R. Anderson and M. Kuhn, “Low cost attacks on tamper resistant devices,” in *Proceedings of the 5th International Workshop on Security Protocols*, 1997, pp. 125–136.
- [23] Anthony D. Wood and John A. Stankovic, “Denial of service in Sensor Networks,” *Computer (Long. Beach. Calif.)*, vol. 10, no. 35, pp. 54–62, 2002.
- [24] J. Newsome, E. Shi, D. Song, and A. Perrig, “The sybil attack in sensor networks,” in *Proceedings of the third international symposium on Information processing in sensor networks - IPSN’04*, 2004, pp. 259–268.
- [25] M. Bellare, R. Canetti, and H. Krawczyk, “Keying Hash Functions for Message Authentication,” in *Advances in Cryptology — CRYPTO ’96*, 1996, pp. 1–15.
- [26] W. X. W. D. Liu, “Detection of man-in-the-middle attacks using physical layer wireless security techniques,” *Wirel. Commun. Mob. Comput.*, vol. 16, no. 4, pp. 408–426, 2016.
- [27] Y.-C. Hu, A. Perrig, and D. B. Johnson, “Packet leashes: a defense against wormhole attacks in wireless networks,” in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, 2003, pp. 1976–1986.
- [28] C. Ioannou, V. Vassiliou, and C. Sergiou, “An Intrusion Detection System for Wireless Sensor Networks,” in *24th International Conference on Telecommunications (ICT)*,

2017, pp. 1–5.

- [29] B. Parno, A. Perrig, and V. Gligor, “Distributed Detection of Node Replication Attacks in Sensor Networks,” in *SP '05 Proceedings of the 2005 IEEE Symposium on Security and Privacy*, 2005, pp. 49–63.
- [30] M. Keshtgari and A. Deljoo, “A Wireless Sensor Network Solution for Precision Agriculture Based on Zigbee Technology,” *Wirel. Sens. Netw.*, 2012.
- [31] A. J. Garcia-Sanchez, F. Garcia-Sanchez, and J. Garcia-Haro, “Wireless sensor network deployment for integrating video-surveillance and data-monitoring in precision agriculture over distributed crops,” *Comput. Electron. Agric.*, 2011.
- [32] W. S. Lee, V. Alchanatis, C. Yang, M. Hirafuji, D. Moshou, and C. Li, “Sensing technologies for precision specialty crop production,” *Comput. Electron. Agric.*, vol. 74, no. 1, pp. 2–33, Oct. 2010.
- [33] M. Ruiz-Altisent *et al.*, “Sensors for product characterization and quality of specialty crops—A review,” *Comput. Electron. Agric.*, vol. 74, no. 2, pp. 176–194, Nov. 2010.
- [34] D. Moshou *et al.*, “Plant disease detection based on data fusion of hyper-spectral and multi-spectral fluorescence imaging using Kohonen maps,” *Real-Time Imaging*, vol. 11, no. 2, pp. 75–83, Apr. 2005.
- [35] N. Sakthipriya, “An effective method for crop monitoring using wireless sensor network,” *Middle - East J. Sci. Res.*, vol. 20, no. 9, pp. 1127–1132, 2014.
- [36] R. Hussain, J. L. Sahgal, P. Mishra, and B. Sharma, “Application of WSN in Rural Development , Agriculture Water Management,” *Int. J. Soft Comput. Eng.*, 2012.
- [37] E. Correa-Hernando *et al.*, “Development of model based sensors for the supervision of a solar dryer,” *Comput. Electron. Agric.*, 2011.
- [38] X. Dong, M. C. Vuran, and S. Irmak, “Autonomous precision agriculture through integration of wireless underground sensor networks with center pivot irrigation systems,” *Ad Hoc Networks*, vol. 11, no. 7, pp. 1975–1987, Sep. 2013.
- [39] Y. Zhu, J. Song, and F. Dong, “Applications of wireless sensor network in the agriculture environment monitoring,” *Procedia Eng.*, vol. 16, pp. 608–614, 2011.
- [40] D. D. Chaudhary, S. P. Nayse, and L. M. Waghmare, “Application of Wireless Sensor Networks for Greenhouse Parameter Control in Precision Agriculture,” *Int. J. Wirel. Mob. Networks*, vol. 3, no. 1, pp. 140–149, Feb. 2011.
- [41] A. Cano, J. L. Añón, C. Reig, C. Millán-Scheiding, and E. López-Baeza, “Automated Soil Moisture Monitoring Wireless Sensor Network for Long-Term Cal/Val Applications,” *Wirel. Sens. Netw.*, vol. 04, no. 08, pp. 202–209, 2012.
- [42] V. Hejlová and V. Voženílek, “Wireless Sensor Network Components for Air Pollution Monitoring in the Urban Environment: Criteria and Analysis for Their Selection,” *Wirel. Sens. Netw.*, vol. 05, no. 12, pp. 229–240, 2013.

- [43] K. K. Khedo, R. Perseedoss, and A. Mungur, "A Wireless Sensor Network Air Pollution Monitoring System," *Int. J. Wirel. Mob. Networks*, vol. 2, no. 2, pp. 31–45, May 2010.
- [44] K. Bouabdellaha, H. Noureddine, and S. Larbi, "Using wireless sensor networks for reliable forest fires detection," in *Procedia Computer Science*, 2013.
- [45] Y. E. Aslan, I. Korpeoglu, and özgür Ulusoy, "A framework for use of wireless sensor networks in forest fire detection and monitoring," *Comput. Environ. Urban Syst.*, 2012.
- [46] H. Yang, Y. Qin, G. Feng, and H. Ci, "Online Monitoring of Geological CO_2 Storage and Leakage Based on Wireless Sensor Networks," *IEEE Sens. J.*, 2013.
- [47] S. Savazzi, U. Spagnolini, L. Goratti, D. Molteni, M. Latva-aho, and M. Nicoli, "Ultra-wide band sensor networks in oil and gas explorations," *IEEE Commun. Mag.*, vol. 51, no. 4, pp. 150–160, Apr. 2013.
- [48] G. zhu CHEN, Z. cai ZHU, G. bo ZHOU, C. feng SHEN, and Y. jing SUN, "Sensor deployment strategy for chain-type wireless underground mine sensor network," *J. China Univ. Min. Technol.*, 2008.
- [49] S. Bhattacharjee, P. Roy, S. Ghosh, S. Misra, and M. S. Obaidat, "Wireless sensor network-based fire detection, alarming, monitoring and prevention system for Bord-and-Pillar coal mines," *J. Syst. Softw.*, 2012.
- [50] R. Liu, T. Xie, Q. Wang, and H. Li, "Space-earth based Integrated Monitoring System for Water Environment," *Procedia Environ. Sci.*, vol. 2, pp. 1307–1314, 2010.
- [51] D. Li and X. Zhu, "CDMA-Based Remote Wireless Water Quality Monitoring System for Intensive Fish Culture," in *2009 WRI International Conference on Communications and Mobile Computing*, 2009, pp. 380–385.
- [52] N. Nasser, a Ali, L. Karim, and S. Belhaouari, "An efficient Wireless Sensor Network-based water quality monitoring system," *Comput. Syst. Appl. (AICCSA), 2013 ACS Int. Conf.*, pp. 1–4, 2013.
- [53] a. Pascale, M. Nicoli, F. Deflorio, B. Dalla Chiara, and U. Spagnolini, "Wireless sensor networks for traffic management and road safety," *IET Intell. Transp. Syst.*, vol. 6, no. 1, p. 67, 2012.
- [54] R. Ali Khan, S. Ahmed Shah, M. Abdul Aleem, Z. Ali Bhutto, A. Ali Shaikh, and M. Aslam Kumbhar, "Wireless Sensor Networks: A Solution for Smart Transportation," *J. Emerg. Trends Comput. Inf. Sci. Wirel. Sens. Networks A Solut. Smart Transp.*, 2012.
- [55] J. Guevara, F. Barrero, E. Vargas, J. Becerra, and S. Toral, "Environmental wireless sensor network for road traffic applications," *IET Intell. Transp. Syst.*, vol. 6, no. 2, p. 177, 2012.
- [56] A. Gaddam, S. C. Mukhopadhyay, and G. Sen Gupta, "Elder care based on cognitive sensor network," *IEEE Sens. J.*, 2011.
- [57] L. Mo, S. Liu, R. X. Gao, D. John, J. W. Staudenmayer, and P. S. Freedson, "Wireless

- design of a multisensor system for physical activity monitoring,” *IEEE Trans. Biomed. Eng.*, 2012.
- [58] R. S. Dilmaghani, H. Bobarshad, M. Ghavami, S. Choobkar, and C. Wolfe, “Wireless Sensor Networks for Monitoring Physiological Signals of Multiple Patients,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 5, no. 4, pp. 347–356, Aug. 2011.
- [59] J. Martinho, L. Prates, and J. Costa, “Design and Implementation of a Wireless Multiparameter Patient Monitoring System,” *Procedia Technol.*, 2014.
- [60] H. Alemdar and C. Ersoy, “Wireless sensor networks for healthcare: A survey,” *Comput. Networks*, vol. 54, no. 15, pp. 2688–2710, Oct. 2010.
- [61] Kevin Hung and Yuan-Ting Zhang, “Implementation of a WAP-based telemedicine system for patient monitoring,” *IEEE Trans. Inf. Technol. Biomed.*, vol. 7, no. 2, pp. 101–107, Jun. 2003.
- [62] R. Fensli, E. Gunnarson, and O. Hejlesen, “A wireless ECG system for continuous event recording and communication to a clinical alarm station,” in *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2005, vol. 3, pp. 2208–2211.
- [63] S. J. Devi, G. S. S. Devi, G. M. T. Selvan, and T. Nadu, “WIRELESS SENSOR NETWORK INTEGRATING WITH CLOUD COMPUTING FOR PATIENT MONITORING,” *Int. J. Eng. Sci. Emerg. Technol.*, vol. 6, no. 3, pp. 316–323, 2013.
- [64] N. K. Suryadevara, S. C. Mukhopadhyay, S. D. T. Kelly, and S. P. S. Gill, “WSN-Based Smart Sensors and Actuator for Power Management in Intelligent Buildings,” *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 2, pp. 564–571, Apr. 2015.
- [65] H. Grindvoll, O. Vermesan, T. Crosbie, R. Bahr, N. Dawood, and G. M. Revel, “A wireless sensor network for intelligent building energy management based on multi communication standards-A case study,” *Electron. J. Inf. Technol. Constr.*, 2012.
- [66] F. Kausar, “Intelligent Home Monitoring Using RSSI in Wireless Sensor Networks,” *Int. J. Comput. Networks Commun.*, vol. 4, no. 6, pp. 33–46, Nov. 2012.
- [67] N. Barroca, L. M. Borges, F. J. Velez, F. Monteiro, M. Górski, and J. Castro-Gomes, “Wireless sensor networks for temperature and humidity monitoring within concrete structures,” *Constr. Build. Mater.*, 2013.
- [68] S. K. Ghosh, M. Suman, R. Datta, S. Member, and P. K. Biswas, “Power Efficient Event Detection Scheme in Wireless Sensor Networks for Railway Bridge Health Monitoring System,” in *International Conference on Advanced Networks and Telecommunications Systems*, 2014, pp. 1–6.
- [69] L. Lamont, M. Toulgoat, D. Mathieu, and G. Patterson, “Tiered Wireless Sensor Network Architecture for Military Surveillance Applications,” in *The Fifth International Conference on Sensor Technologies and Applications*, 2011, no. c, pp. 288–294.
- [70] G. Padmavathi, D. Shanmugapriya, and M. Kalaivani, “A Study on Vehicle Detection and Tracking Using Wireless Sensor Networks,” *Wirel. Sens. Netw.*, 2010.

- [71] J. I. Huircán *et al.*, “ZigBee-based wireless sensor network localization for cattle monitoring in grazing fields,” *Comput. Electron. Agric.*, vol. 74, no. 2, pp. 258–264, Nov. 2010.
- [72] G. Sasikumar, H. V. Ramamoorthy, and S. N. Mohamed, “An Analysis on Animal Tracking System using Wireless Sensors,” *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 4, no. 9, pp. 155–162, 2014.
- [73] A.-J. Garcia-Sanchez *et al.*, “Wireless Sensor Network Deployment for Monitoring Wildlife Passages,” *Sensors*, vol. 10, no. 8, pp. 7236–7262, Aug. 2010.
- [74] F. Lalooses, H. Susanto, and C. H. Chang, “Approach for Tracking Wildlife Using Wireless Sensor,” *Proc. 7th Int. Conf. New Technologies Distrib. Syst.*, pp. 1–7, 2007.
- [75] J. Polastre, R. Szewczyk, A. Mainwaring, D. Culler, and J. Anderson, “Analysis of Wireless Sensor Networks for Habitat Monitoring,” in *Wireless Sensor Networks*, Boston, MA: Springer US, 2006, pp. 399–423.
- [76] Ashton K., “That ‘Internet of Things’ Thing: In the real world, things matter more than ideas,” *RFID J.*, p. 4986, 2009.
- [77] A. Bröring *et al.*, “New Generation Sensor Web Enablement,” *Sensors*, vol. 11, no. 3, pp. 2652–2699, Mar. 2011.
- [78] M. Presser, P. Barnaghi, M. Eurich, and C. Villalonga, “The SENSEI project: integrating the physical world with the digital world of the network of the future,” *IEEE Commun. Mag.*, vol. 47, no. 4, pp. 1–4, Apr. 2009.
- [79] R. Buyya, J. Broberg, and A. Goscinski, *Cloud Computing: Principles and Paradigms*. Hoboken, New Jersey, USA: John Wiley & Sons, Inc., 2011.
- [80] C. S. Yeo *et al.*, “Utility Computing on Global Grids Benefits of Utility Computing,” in *The Handbook of Computer Networks*, New York, USA: John Wiley & Sons, 2007, pp. 1–26.
- [81] P. Mell and T. Grance, “The NIST Definition of Cloud Computing - Recommendations of the National Institute of Standards and Technology,” Gaithersburg, Maryland, USA, 2011.
- [82] M. Tulloch, *Understanding Microsoft Virtualization Solutions, 2nd Edition*. Redmond, Washington, USA: Microsoft Press books, 2010.
- [83] VMware, “Understanding Full Virtualization, Paravirtualization, and Hardware Assist,” Palo Alto, California, USA, 2007.
- [84] D. A. B. Fernandes, L. F. B. Soares, J. V. Gomes, M. M. Freire, and P. R. M. Inácio, “Security issues in cloud environments: a survey,” *Int. J. Inf. Secur.*, vol. 13, no. 2, pp. 113–170, 2014.
- [85] G. Privat, “From Smart Devices to Ambient Communication,” in *From RFID to the Internet of Things*, 2006, no. March.

- [86] A. Kansal, S. Nath, J. Liu, and F. Zhao, "SenseWeb: An infrastructure for shared sensing," *IEEE Multimed.*, vol. 14, no. 4, pp. 8–13, 2007.
- [87] R. Dickerson, J. Lu, J. Lu, and K. Whitehouse, "Integrating wireless sensor networks and the internet: a security analysis," in *The Internet of Things*, vol. 19, no. 2, J. Hyuk Park, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 360–375.
- [88] T. Luckenbach, P. Gober, S. Arbanowski, A. Kotsopoulos, and K. Kim, "TinyREST: A protocol for integrating sensor networks into the internet," *Proc. REALWSN 2005*, pp. 101–105, 2005.
- [89] M. R. Kosanovic and M. K. Stojcev, "Implementation of TCP / IP Protocols in Wireless Sensor Networks," *Proc. Int. Sci. Conf. Information, Commun. Energy Syst. Technol.*, pp. 143–146, 2007.
- [90] T. Braun, T. Voigt, and A. Dunkels, "TCP support for sensor networks," *2007 Fourth Annu. Conf. Wirel. Demand Netw. Syst. Serv. WONS'07*, pp. 162–169, 2007.
- [91] M. Domingues, C. Friças, and P. Veiga, "Is global IPv6 deployment on track?," *Internet Res.*, vol. 17, no. 5, pp. 505–518, Oct. 2007.
- [92] B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao, "Tiny Web Services for Sensor Device Interoperability," in *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*, 2008, pp. 567–568.
- [93] J. P. Walters, Z. Liang, W. Shi, and V. Chaudhary, "Wireless Sensor Network Security : A Survey," *Distrib. Grid, Pervasive Comput.*, pp. 1–22, 2007.
- [94] Y. W. Law, J. Doumen, and P. Hartel, "Survey and benchmark of block ciphers for wireless sensor networks," *ACM Trans. Sens. Networks*, vol. 2, no. 1, pp. 65–93, Feb. 2006.
- [95] Kyung Jun Choi and Jong-In Song, "Investigation of feasible cryptographic algorithms for wireless sensor network," in *2006 8th International Conference Advanced Communication Technology*, 2006, pp. 3 pp. – 1381.
- [96] A. Liu and P. Ning, "TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks," in *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*, 2008, pp. 245–256.
- [97] R. Roman, J. Lopez, and S. Gritzalis, "Situation awareness mechanisms for wireless sensor networks," *IEEE Commun. Mag.*, vol. 46, no. 4, pp. 102–107, Apr. 2008.
- [98] G. Montenegro and N. Kushalnagar, "RFC 4944: transmission of IPv6 packets over IEEE 802.15.4 networks," vol. 12, no. 235, p. 245, 2007.
- [99] V. Gupta *et al.*, "Sizzle: A standards-based end-to-end security architecture for the embedded Internet," *Pervasive Mob. Comput.*, vol. 1, no. 4, pp. 425–445, Dec. 2005.
- [100] K. Ahmed and M. Gregory, "Integrating Wireless Sensor Networks with Cloud Computing," *2011 Seventh Int. Conf. Mob. Ad-hoc Sens. Networks*, pp. 364–366, 2011.

- [101] W. Chung, P. Yu, and C. Huang, “Cloud computing system based on wireless sensor network,” in *Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2013, pp. 877–880.
- [102] A. Freeman and J. C. Rattz, *Pro LINQ: Language Integrated Query in C# 2008*. Berkeley, CA: Apress, 2008.
- [103] R. Piyare *et al.*, “Integrating Wireless Sensor Network into Cloud services for real-time data collection,” in *2013 International Conference on ICT Convergence (ICTC)*, 2013, pp. 752–756.
- [104] W. Kurschl and W. Beer, “Combining cloud computing and wireless sensor networks,” in *Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services - iiWAS '09*, 2009, p. 512.
- [105] N. A. B. Gray, “Performance of Java middleware - Java RMI , JAXRPC , and CORBA,” in *Proceedings of The Sixth Australasian Workshop on Software and System Architectures (AWSA 2005)*, 2005, no. March.
- [106] S. Artemio, B. Leonardo, J.-H. Hugo, M.-M. J. Carlos, A.-F. Marco A., and P. J. Carlos, “Evaluation of CORBA and Web Services in distributed applications,” in *CONIELECOMP 2012, 22nd International Conference on Electrical Communications and Computers*, 2012, pp. 97–100.
- [107] J. A. Manrique, J. S. Rueda-Rueda, and J. M. T. Portocarrero, “Contrasting Internet of Things and Wireless Sensor Network from a Conceptual Overview,” in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2016, pp. 252–257.
- [108] C. Neuman, “Scale in Distributed Systems,” in *Readings in Distributed Computing Systems.*, Los Alamitos, CA, USA: IEEE Computer Society Press, 1994.
- [109] [Http://web.mit.edu/smadnick/www/wp/2013-01.pdf](http://web.mit.edu/smadnick/www/wp/2013-01.pdf), “Cloud Computing Models,” *Last Accessed March*, 2018. .
- [110] V. Edmondson, M. Cerny, M. Lim, B. Gledson, S. Lockley, and J. Woodward, “A smart sewer asset information model to enable an ‘Internet of Things’ for operational wastewater management,” *Autom. Constr.*, vol. 91, pp. 193–205, Jul. 2018.
- [111] M. Tebaa, S. El Hajji, a El Ghazi, S. El Hajji, and A. El Ghazi, “Homomorphic Encryption Applied to the Cloud Computing Security,” in *Proceedings of the World Congress on Engineering*, 2012, vol. 1, pp. 536–539.
- [112] J. Domingo-Ferrer, “A Provably Secure Additive and Multiplicative Privacy Homomorphism*,” in *Information Security*, Springer, Berlin, Heidelberg, 2002, pp. 471–483.
- [113] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.
- [114] R. L. Rivest, L. Adleman, and M. L. Dertouzos, “On Data Banks and Privacy

- Homomorphisms,” *Found. Secur. Comput.*, pp. 169–180, 1978.
- [115] S. Goldwasser and S. Micali, “Probabilistic encryption & how to play mental poker keeping secret all partial information,” *STOC '82 Proc. fourteenth Annu. ACM Symp. Theory Comput.*, pp. 365–377, 1982.
- [116] T. Elgamal, “A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms,” *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, 1985.
- [117] W. Diffie, W. Diffie, and M. E. Hellman, “New Directions in Cryptography,” *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, 1976.
- [118] V. S. Miller, “Use of Elliptic Curves in Cryptography,” *Adv. Cryptol. — CRYPTO '85 Proc.*, pp. 417–426, 1985.
- [119] N. Koblitz, “Elliptic curve cryptosystems,” *Math. Comput.*, vol. 48, no. 177, pp. 203–203, 1987.
- [120] D. Boneh, E.-J. Goh, and K. Nissim, “Evaluating 2-DNF Formulas on Ciphertexts,” pp. 325–341, 2005.
- [121] M. Fellows and N. Koblitz, “Combinatorial cryptosystems galore,” *Contemp. Math.*, vol. 168, no. July 1993, pp. 51–61, 1993.
- [122] Josh Benaloh, “Dense Probabilistic Encryption,” in *In Proceedings of the Workshop on Selected Areas of Cryptography*, 1994, pp. 120–128.
- [123] L. Fousse, P. Lafourcade, and M. Alnuaimi, “Benaloh’s Dense Probabilistic Encryption Revisited,” pp. 348–362, 2011.
- [124] D. Naccache and J. Stern, “A new public-key cryptosystem,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 1233, pp. 27–36, 1997.
- [125] D. Naccache and J. Stern, “A new public key cryptosystem based on higher residues,” in *Proceedings of the 5th ACM conference on ...*, 1998, vol. 10324, no. 24, pp. 59–66.
- [126] T. Okamoto and S. Uchiyama, “A New Public-Key Cryptosystem as Secure as Factoring,” in *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, 1998, vol. 1403, pp. 308–318.
- [127] J.-S. Coron, D. Naccache, and P. Paillier, “Accelerating Okamoto-Uchiyama public-key cryptosystem,” *Electron. Lett.*, vol. 35, no. 4, p. 291, 1999.
- [128] P. Paillier, “Public Key Crypto Systems Based on Composite Degree Residuosity Classes,” *Adv. Cryptogr. -- {EUROCRYPT'99}*, vol. 1592, pp. 223–238, 1999.
- [129] D. Catalano, R. G. N. Howgrave-graham, and P. Q. Nguyen, “Paillier ’ s Cryptosystem Revisited,” *Proc. 8th ACM Conf. Comput. Commun. Secur.*, pp. 206–214, 2001.
- [130] T. Sander and A. Young, “Non-Interactive CryptoComputing for NC,” pp. 1–26, 1999.

- [131] D. Choi and S. Choi, "Improvement of probabilistic public key cryptosystems using discrete logarithm," *4th Int. Conf. Seoul*, pp. 72–80, 2001.
- [132] K. Sakurai and T. Takagi, "On the Security of a Modified Paillier Public-Key Primitive," pp. 436–448, 2002.
- [133] I. Damgård and M. J. Jurik, "A Generalisation, a Simplification and some Applications of Paillier's Probabilistic Public-Key System," *PKC 2001 Public Key Cryptogr.*, no. December, pp. 119–136, 2001.
- [134] P. Paillier, "Trapdooring Discrete Logarithms on Elliptic Curves over Rings," *Adv. Cryptol. --{ASIA}\-CRYPT}* '2000, vol. 1976, pp. 572–584, 2000.
- [135] S. D. Galbraith, "Elliptic curve Paillier schemes," *J. Cryptol.*, pp. 1–10, 2002.
- [136] K. Schmidt-samoa and T. Takagi, "Paillier's Cryptosystem Modulo p^2q and Its," pp. 296–313, 2005.
- [137] C. Gentry, "a Fully Homomorphic Encryption Scheme," Stanford, 2009.
- [138] L. Wang, L. Wang, Y. Pan, Z. Zhang, and Y. Yang, "Discrete logarithm based additively homomorphic encryption and secure data aggregation," *Inf. Sci. (Ny)*, vol. 181, no. 16, pp. 3308–3322, 2011.
- [139] O. Ugus, D. Westhoff, R. Laue, A. Shoufan, and S. A. Huss, "Optimized Implementation of Elliptic Curve Based Additive Homomorphic Encryption for Wireless Sensor Networks," *Arxiv Prepr. arXiv09033900*, Mar. 2009.
- [140] M. Elhoseny, H. Elminir, A. Riad, and X. Yuan, "A secure data routing schema for WSN using Elliptic Curve Cryptography and homomorphic encryption," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 28, no. 3, pp. 262–275, 2016.
- [141] K. Xie *et al.*, "An efficient privacy-preserving compressive data gathering scheme in WSNs," *Inf. Sci. (Ny)*, vol. 390, pp. 82–94, 2017.
- [142] S. Roy, M. Conti, S. Setia, and S. Jajodia, "Secure data aggregation in wireless sensor networks," *IEEE Trans. Inf. Forensics Secur.*, vol. 7, no. 3, pp. 1040–1052, 2012.
- [143] L. Ertaul and J. H. Yang, "Implementation of Domingo Ferrer's a New Privacy Homomorphism (DF a New PH) in Securing Wireless Sensor Networks (WSN).," *Secur. Manag.*, no. January 2008, pp. 498–504, 2008.
- [144] P. Zhang, S. Wang, K. Guo, and J. Wang, "A secure data collection scheme based on compressive sensing in wireless sensor networks," *Ad Hoc Networks*, vol. 70, pp. 73–84, Mar. 2018.
- [145] P. Zhang, J. Wang, K. Guo, F. Wu, and G. Min, "Multi-functional secure data aggregation schemes for WSNs," *Ad Hoc Networks*, vol. 69, pp. 86–99, 2018.
- [146] K. Ahmed and M. Gregory, "Integrating Wireless Sensor Networks with Cloud Computing," in *2011 Seventh International Conference on Mobile Ad-hoc and Sensor Networks*, 2011, pp. 364–366.

- [147] C. I. Ngabo and O. E. L. Beqqali, “Real-time Lighting Poles Monitoring by using Wireless Sensor Networks Applied to the Smart Cities,” in *ACM International Conference Proceeding Series*, 2016, vol. Part F1263.
- [148] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, “Fully homomorphic encryption over the integers,” in *Advances in Cryptology— ...*, 2010, pp. 1–28.
- [149] M. Johnson *et al.*, “A Comparative Review of Wireless Sensor Network Mote Technologies A Comparative Review of Wireless Sensor Network Mote Technologies,” in *SENSORS, 2009 IEEE*, 2009, no. November, pp. 1439–1442.
- [150] E. Holohan and M. Schukat, “Authentication using virtual certificate authorities: A new security paradigm for wireless sensor networks,” in *Proceedings - 2010 9th IEEE International Symposium on Network Computing and Applications, NCA 2010*, 2010, pp. 92–99.
- [151] M. Y. Malik, “Efficient Implementation of Elliptic Curve Cryptography Using Low-power Digital Signal Processor,” in *Advanced Communication Technology (ICACT)*, 2011, vol. 2, no. x, pp. 1464–1468.
- [152] C. Castelluccia, E. Mykletun, and G. Tsudik, “Efficient aggregation of encrypted data in wireless sensor networks,” in *MobiQuitous 2005: Second Annual International Conference on Mobile and Ubiquitous Systems -Networking and Services*, 2005, pp. 109–117.
- [153] S. Goldwasser and S. Micali, “Probabilistic encryption,” *J. Comput. Syst. Sci.*, vol. 28, no. 2, pp. 270–299, 1984.
- [154] K. Rabah, “Theory and Implementation of Elliptic Curve Cryptography,” *J. Appl. Sci.*, vol. 5, no. 4, pp. 604–633, 2005.
- [155] M. M. Potey, C. A. Dhote, and D. H. Sharma, *Networking Communication and Data Knowledge Engineering*, vol. 4. Singapore: Springer Singapore, 2018.
- [156] C. R. E. Newman, *Computer and Network Security Essentials*. Cham: Springer International Publishing, 2018.
- [157] K. Ireland and M. Rosen, *Graduate Texts in Mathematics: A Classical Introduction to Modern Number Theory*, 2nd ed. Springer-Verlag, 1990.
- [158] M. Zappatore, A. Longo, and M. A. Bochicchio, “Crowd-sensing our Smart Cities: a Platform for Noise Monitoring and Acoustic Urban Planning,” *J. Commun. Softw. Syst.*, vol. 13, no. 2, p. 53, Jun. 2017.
- [159] C. R. Farrar and K. Worden, “An Introduction to Structural Health Monitoring,” in *CISM International Centre for Mechanical Sciences, Courses and Lectures*, 2010, pp. 1–17.
- [160] L. Alonso, J. Barbarán, J. Chen, M. Díaz, L. Llopis, and B. Rubio, “Middleware and communication technologies for structural health monitoring of critical infrastructures: A survey,” *Comput. Stand. Interfaces*, vol. 56, pp. 83–100, Feb. 2018.
- [161] E. Zaloshnja *et al.*, “Reducing injuries among Native Americans: Five cost-outcome

- analyses,” *Accid. Anal. Prev.*, 2003.
- [162] B. A. J. Clark, “Outdoor lighting and crime, Part 2: coupled growth,” *Society*, 2003.
- [163] R. Clarke, “Improving street lighting to reduce crime in residential areas,” 2008.
- [164] D. Biswas *et al.*, “Real-time Arm Movement Recognition using FPGA,” *2015 IEEE Int. Symp. Circuits Syst.*, pp. 766–769, 2015.
- [165] A. M. Khan, Y. K. Lee, S. Y. Lee, and T. S. Kim, “A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer,” *IEEE Trans. Inf. Technol. Biomed.*, vol. 14, no. 5, pp. 1166–1172, 2010.
- [166] P. Sarcevic, Z. Kincses, and S. Pletl, “Comparison of different classifiers in movement recognition using WSN-based wrist-mounted sensors,” *SAS 2015 - 2015 IEEE Sensors Appl. Symp. Proc.*, 2015.
- [167] W. Sriborrurux, P. Leamsunran, and P. Dan-Klang, “Real-time system for monitoring activity among the elderly using an RF SoC device with triaxial accelerometer data over a wireless sensor network,” *Conf. Proc. - 2014 IEEE MTT-S Int. Microw. Work. Ser. RF Wirel. Technol. Biomed. Healthc. Appl. IMWS-Bio 2014*, 2015.
- [168] M. A. Awan, Z. Guangbin, and S. D. Kim, “A dynamic approach to recognize activities in WSN,” *Int. J. Distrib. Sens. Networks*, vol. 2013, 2013.
- [169] N. Xu *et al.*, “A wireless sensor network For structural monitoring,” *Proc. 2nd Int. Conf. Embed. networked Sens. Syst. - SenSys '04*, p. 13, 2004.
- [170] J. P. J. Paek, K. Chintalapudi, R. Govindan, J. Caffrey, and S. Masri, “A Wireless Sensor Network for Structural Health Monitoring: Performance and Experience,” *Second IEEE Work. Embed. Networked Sensors 2005 EmNetSII*, pp. 1–10, 2005.
- [171] B. S. Chowdhry, F. K. Shaikh, S. Ali, and E. Felemban, “Experimental Evaluation of Vibration Response Based Bridge Damage Detection Using Wireless Sensor Networks,” *Wirel. Pers. Commun.*, vol. 85, no. 2, pp. 499–510, 2015.
- [172] S. Gaikwad, A. Khandare, and A. Rai, “Design WSN Node For Protection Of Forest Trees Against Poaching based on ZigBee,” in *International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 2015, pp. 3–6.
- [173] Libelium Comunicaciones Distribuidas S.L., “Waspote Datasheet,” 2016. [Online]. Available: http://www.libelium.com/downloads/documentation/waspote_datasheet.pdf. [Accessed: 24-Mar-2019].