Secure Data Aggregation in Wireless Sensor Networks

By

Paresh M. Solanki

08MCE018

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AHMEDABAD-382481

May 2010

Secure Data Aggregation in Wireless Sensor Networks

Major Project

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology in Computer Science and Engineering

By

Paresh M. Solanki

08MCE018

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AHMEDABAD-382481

May 2010

Declaration

This is to certify that

i) The thesis comprises my original work towards the degree of Master of Technology in Computer Science and Engineering at Nirma University and has not been submitted elsewhere for a degree.

ii) Due acknowledgement has been made in the text to all other material used.

Paresh M. Solanki

Certificate

This is to certify that the Major Project entitled "**Secure Data Aggregation in Wireless Sensor Networks**" submitted by **Paresh M. Solanki** (08MCE018), towards the partial fulfillment of the requirements for the degree of Master of Technology in Computer Science and Engineering of Nirma University of Science and Technology, Ahmedabad is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Prof. Gaurang Raval
Associate Professor and Guide,
Department of Comp. Sci. & Engg.,
Institute of Technology,
Nirma University, Ahmedabad

Dr. S.N. Pradhan
Professor ,PG Coordinator,
Department of Comp. Sci. & Engg.,
Institute of Technology,
Nirma University, Ahmedabad

Prof. D. J. Patel
Professor and Head,
Department of Comp. Sci. & Engg.,
Institute of Technology,
Nirma University, Ahmedabad

Dr Ketan Kotecha
Director,
Institute of Technology,
Nirma University, Ahmedabad,

Abstract

The rapid advancement of hardware technology has enabled the development of small, powerful, and inexpensive sensor nodes, which are capable of sensing, computation and wireless communication. This revolutionizes the deployment of wireless sensor network for monitoring some area and collecting regarding information. However, limited energy constraint presents a major challenge such vision to become reality. The main task of such a network is to gather information from node and transmit to the base station for further processing. In WSN 70% of energy consumption is due to data transmission. First, Due to hostile environment and unique properties of wireless sensor network we do not want to send all raw data samples directly to the sink node Hence Data aggregation is required. Second, Wireless sensor nodes are often deployed in an open environment such as a battlefield. Thus, attackers can capture sensor nodes to steal secret data or reprogram them to execute malicious code. Hence secure data aggregation is required. End to End secure encrypted data aggregation is better than hop by hop secure data aggregation because Hop by Hop secure data aggregation is resource consuming.

Currently different schemes are available for end to end secure encrypted data aggregation but they required some more enhancement. Hence implement or enhance the light weight secure data aggregation (minimal security primitives) scheme like Concealed Data Aggregation scheme.

In this Major Project, I have used the JIST/SWANS simulator for implementation. First Implemented the energy model for JIST/SWANS simulator which is measure the energy of each sensor node during idle,receive,transmit mode. Second implemented the Cluster architecture by modified heartbeat application. Third implemented the Secure data aggregation scheme using Privacy Homomorphism which is based on End to End secure data aggregation. I have measured the Energy usage of each sensor nodes during the Data Aggregation and Secure data aggregation Process.

Acknowledgements

Before, I get into thick of things; I would like to add a few heartfelt words for the people who were part of my thesis in numerous ways, people who gave unending support right from the beginning. During this period, the faculty members and my batch mates took keen interest and participated actively. They are very efficient and qualified in their respective disciplines.

I express my sincere gratitude to my thesis guide **Associate Prof. Gaurang Raval**, Computer Science & Engineering Dept., Institute of Technology, Nirma University for all his affectionate encouragement and guidance during the entire Thesis. His views and inputs were very helpful throughout the process.

I would like to thank **Dr. S.N.Pradhan**, M.Tech Co-ordinator, Computer Science & Engineering Dept., Institute of Technology,Nirma University who suggested many related points and is always very constructive and helpful.

I would like to thank **Dr. Ketan Kotecha**, Hon'ble Director, Nirma Institute of Technology, Nirma University for the facilities and environment for research.

I am indebted to my **Father** and **Mother** for their unconditional love, encouragement and support. My parents have always put education as a first priority in my life. They taught me to value honesty, courage, and humility above all other virtues. My parents have always been there for me as an unwavering support.

Lastly I would like to thank my beloved wife **Dharal** and My son **Divy** for their love, support and encouragement that they have given me throughout my life, helping me to persevere in my studies. She has been my source of strength.

In addition, I thank almighty god who made me a normal human being and gave me such strength.

- Paresh M. Solanki

08MCE018

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Before giving any outline of this thesis, I am going to introduce you to the world of wireless sensor networks. I will briefly describe challenges and application of wireless sensor networks. I will also provide you the motivation behind the selection of the Secure data aggregation for this thesis.

1.1 Wireless Sensor Networks

A wireless sensor network is a wireless network consisting of tiny devices which monitor physical or environmental conditions such as temperature, pressure, motion or pollutants etc. at different regions. The tiny device, known as sensor node, consists of a radio transceiver, microcontroller, power supply, and the actual sensor. Initially sensor network were used for military applications but now they are widely used for civilian application area including environment and habitat monitoring, healthcare application and so on.

Normally sensor nodes are spatially distributed throughout the region which has to be monitored; they self-organize in to a network through wireless communication, and collaborate with each other to accomplish the common task. With the going time, sensor nodes are becoming smaller, cheaper, and more powerful which enable us to deploy a large-scale sensor network. Basic features of sensor networks are self-

organizing capabilities, dynamic network topology, limited power, node failures & mobility of nodes, short-range broadcast communication and multi-hop routing, and large scale of deployment[1]. The strength of wireless sensor network lies in their flexibility and scalability. The capability of self-organize and wireless communication made them to be deployed in an ad-hoc fashion in remote or hazardous location without the need of any existing infrastructure. Through multi-hop communication a sensor node can communicate a far away node in the network. This allows the addition of sensor nodes in the network to expand the monitored area and hence proves its scalability & flexibility property.

The basic goals of a WSN are to: (i) determine the value of physical variables at a given location, (ii) detect the occurrence of events of interest, and estimate parameters of the detected events, (iii) classify a detected object, and (iv) track an object. Thus, the important requirements of a WSN are: (i) use of a large number of sensors, (ii) attachment of stationary sensors, (iii) low energy consumption, (iv) self-organisation capability, (v) collaborative signal processing, and (vi) querying ability.[2]

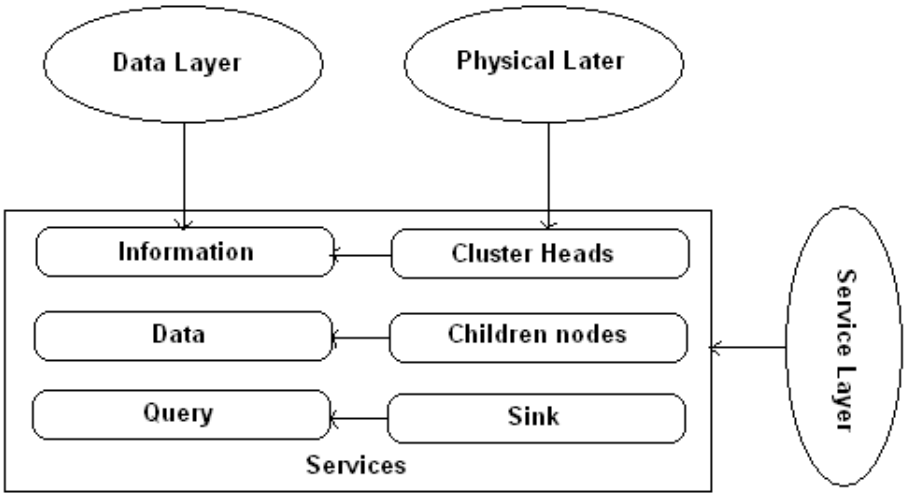

Figure 1.1: Wireless Sensor Network Architecture

Fig. 1.1 shows the general architecture of a sensor network. As can be seen in the figure, the three important layers are the services layer, data layer, and physical layer. The layers provide routing protocol, data dissemination, and aggregation. The

physical layer containing the node defines itself as either a sink node, children node, cluster head. Messages are modeled in the data link layer. Broadcasting of a query is carried out by the use of sink node. The broadcasting can be either to the sensor network or to a designated region depending on the way the query is being used. In response to a change in the physical parameter the sensor nodes, which are close to the sensed object, broadcast this information to their neighbouring sensor nodes. In effect, a cluster head will receive this transmission. The role of a cluster head is to process and aggregate this data and broadcast it to the sink node(s) through the neighboring nodes.

Some of the important application domains of WSNs are listed below.

- Military sensor networks
- Networks for detecting chemical, biological, radiological, nuclear, and explosive material
- Environmental monitoring networks
- Traffic sensor networks
- Surveillance applications
- Parking systems in shopping malls and big markets

1.2 Sensor Network Challenges

Wireless sensor network promise a wide variety of application and to realize these application in real world, we need more efficient protocols and algorithms. Designing a new protocol or algorithm address some challenges which are need to be clearly understood. These challenges are summarized below:

Physical Resource Constraints:

The most important constraint imposed on sensor network is the limited battery power of sensor nodes. The effective lifetime of a sensor node is directly determined by its power supply. Hence lifetime of a sensor network is also determined by the power supply. Hence the energy consumption is main design issue of a protocol. Limited

computational power and memory size is another constraint that affects the amount of data that can be stored in individual sensor nodes. So the protocol should be simple and light-weighted. Communication delay in sensor network can be high due to limited communication channel shared by all nodes within each others transmission range.

Ad-hoc Deployment:

Many application or most of them requires the ad-hoc deployment of sensor nodes in the region. Sensor nodes are randomly deployed over the region without any infrastructure which requires the system to be able to cope up with random distribution and form connection between the nodes. As an example, for fire detection in a forest the nodes typically would be dropped in to the forest from a plane.

Fault-Tolerance:

In a hostile environment, a sensor node may fail due to physical damage or lack of energy (power). If some nodes fail, the protocols that are working upon must accommodate these changes in the network. As an example, for routing or aggregation protocol, they must find suitable paths or aggregation point in case of these kinds of failures.

Scalability:

In a region, depending upon the application, the number of sensor nodes deployed could be in order of hundreds, thousands or more. The protocols must scalable enough to respond and operate with such large number of sensor nodes.

Quality of Service:

Some sensor application are very time critical which means the data should be delivered within a certain period of time from the moment it is sensed, otherwise the data will be careless. So this could be a QOS parameter for some applications.

1.3 Motivation

Wireless sensor network are new age technology which can be used at a place which is possibly hostile & human inaccessible. Wireless sensor network is a class of wireless network which consists of thousands of densely deployed sensor nodes which can be used for a number of applications. Sensor nodes are tiny devices which are composed of a sensing unit, a radio, a processor & a limited battery power. A network of thousands of sensor nodes could be setup for many applications such as environmental monitoring, health monitoring, disaster management, industrial areas, military application and many more.

In wireless sensor network, there are so many challenges & issues as above already been discussed. The main challenges are how to provide maximum lifetime to network, how to provide robustness to network and Security issues. As sensor network totally rely on battery power, the main aim for maximizing lifetime of network is to conserve battery power or energy.

In sensor network, the energy is mainly consumed for three purposes: data transmission, signal processing, and hardware operation. It is said in [4] that 70% of energy consumption is due to data transmission. So for maximizing the network lifetime, the process of data transmission should be optimized. The data transmission can be optimized by using efficient routing protocols and effective ways of data aggregation. Routing protocols have their own ways to save energy of nodes in the network by providing or creating an optimal route from sensor nodes to base station or sink. Data aggregation plays an important role in energy conservation of sensor network. Data aggregation methods are used not only for finding an optimal path from source to destination but also to eliminate the redundancy of data, since transmitting huge volume of raw data is an energy intensive operation, and thus minimizing the number of data transmission. Also multiple sensors may see the same phenomenon, albeit from different view and if this data can be reconciled into a more meaningful form as

it passes through the network, it becomes more useful to an application. One more benefit of data aggregation is that if data is processed as it is passed through the network, it may be compressed thus occupying less bandwidth. This also reduces the amount of transmission power expended by nodes.

The security issues, data confidentiality and integrity, in data aggregation become vital when the sensor network is deployed in a hostile environment. Efficient delivery of sensor readings are reported to an off-site user in such a way that ensures these reported readings have not been altered. So Secure Data Aggregation is more challenging task in wireless sensor network.

1.4 Problem Definition

Wireless sensor networks collect the data from sensor nodes, process it and send it to the base station. more energy consumption is due to data transmission. because of characteristics of sensor nodes not all the data from sensor nodes send to the base station directly but these data is aggregate first then send to the base station. Wireless sensor nodes are often deployed in an open environment such as a battlefield or other similar applications. Attackers may capture sensor nodes to steal secret data or reprogram them to execute malicious code. so Secure Data aggregation is required. hop by hop secure data aggregation is energy consumption because data aggregator nodes first decrypt the data then aggregate it and again encrypt it. Hop by Hop secure data aggregation is resource consuming. So End to End secure encrypted data aggregation is preferred. Because aggregation process is done on encrypted data.

Currently different schemes are available for end to end secure encrypted data aggregation but needs some more attention. Few enhancements cannot be ruled out for the light weight secure data aggregation. My focus is on the same for implementation.

1.5 Organization of work

This Major Project is divided into two part:

In Major Project part-I,

- Study papers on data aggregation and secure data aggregation and Privacy Homomorphism cryptographic scheme.
- Study Wireless Sensor Network simulator - JIST/SWANS.
- Implement Energy Model for JIST/SWANS Simulator.
- Implement Cluster Architecture using Heartbeat Application.

In Major Project part-II,

- Implement Data aggregation scheme
- Implement cluster based secure data aggregation scheme with privacy homomorphism cryptographic method.
- Further enhancement of secure data aggregation scheme.

1.6 OutLine of Thesis Report

The rest of the thesis is organized as follows.

Chapter 2, gives a detailed literature servey related to Secure data aggregation and different existing schemes.

Chapter 3, describe the wireless sensor network simulator: JIST/SWANS.

Chapter 4, provide the details of energy model implementation for jist/swans simulator.

Chapter 5, Provide the Details of Cluster Architecture Implementation using Heartbeat protocol. and shown the result of analysis of cluster architecture.

Chapter 6, Provide the Details of data aggregation and implementation steps of data aggregation in simulation environment.

Chapter 7, Provide the Details of secure data aggregation and implementation of PH algorithm.

Chapter 8, Provide the Details of Different simulation result analysis of secure data aggregation.

Finally, in **chapter 9** concluding remarks and scope for future work is presented.

Materials (e.g. URLs, books, and research papers) used and studied are given in Reference.

Chapter 2

Literature Survey

There is a strong conflict between security and data aggregation schemes. Security schemes require sensor nodes to encrypt and authenticate any sensed data prior to its transmission and prefer data to be decrypted by the base station. On the other hand, data aggregation protocols prefer plain data to implement data aggregation at every intermediate node so that energy efficiency is maximized. Data aggregation results in alterations in sensor data and therefore it is a challenging task to provide source and data authentication along with data aggregation. Due to these conflicting goals, data aggregation and security schemes must be designed together so that data aggregation can be performed without sacrificing security. Due to hostile environments and unique properties of wireless sensor networks, it is a challenging task to protect sensitive information transmitted by wireless sensor networks.

Suat Ozdemir [3] Security requirements of wireless sensor networks can be satisfied using either symmetric key or asymmetric key cryptography. Symmetric key cryptography is preferable. Hop-by-hop secure data aggregation protocols cannot provide data confidentiality and result in latency because of the decryption/encryption process. Because of the problem of hop by hop SDA protocols, some other protocols were introduced which perform data aggregation without requiring the decryption of the sensor data at data aggregators.

2.1 Secure Data Aggregation using plain sensor data

Here i have given details about different available schemes for secure data aggregation. these schemes required plain sensor data for secure data aggregation.

Secure Information Aggregation (SIA):

SIA uses random sampling mechanisms and interactive proofs to verify aggregated data. It assumes that each sensor node has a unique identifier and shares a separate secret cryptographic key with the base station and with the aggregator. These keys enable data confidentiality, integrity and authentication.

SecureDAV:

This scheme is similar to SIA except that elliptic curve cryptography is used for encryption purposes. The scheme incurs high communication overhead on data validation and supports only the average aggregation function. SecureDAV is a clustered approach.

Witness based Data Aggregation(WDA):

Assures the validation of the data sent from data fusion nodes to the base station. In order to prove the validity of the fusion result, the fusion node has to provide proofs from several witnesses. A witness is one who also conducts data fusion like a data fusion node, but does not forward its result to the base station. Instead, each witness computes the message authentication code (MAC) of the result and then provides it to the data fusion node who must forward the proofs to the base station.

Secure Hop-by-hop Data Aggregation Protocol (SDAP):

SDAP provides data confidentiality, source authentication, and data integrity. Compared to low-level sensor nodes, more trust is placed on the high-level nodes (i.e., nodes closer to the root) during a normal hop-by-hop aggregation process in a tree topology.

Secure and rEliable Data Aggregation protocol (SELDA):

The basic idea behind SELDA is that sensor nodes observe actions of their neighboring nodes to develop trust levels (trustworthiness) for both the environment and the neighboring nodes. SELDA increases the reliability of the aggregated data at the expense of a tolerable communication overhead.

Data Aggregation and Authentication protocol(DAA):

It detects the false data with data aggregation and confidentiality. False data detection and data confidentiality increase the communication overhead.

All of the above secure data protocols use actual sensor data for aggregation and hence require decryption of sensor data at aggregators.

Following protocols does not require plain data for secure data aggregation.

Energy efficient and Secure Pattern based Data Aggregation (ESPDA :)

ESPDA uses pattern codes to perform data aggregation. ESPDA is also secure because cluster heads do not need to decrypt the data for data aggregation and no encryption/decryption key is broadcast. It is Cluster based data aggregation scheme.

Secure Reference-Based Data Aggregation (SRDA) :

It is cluster based approach. In SRDA, raw data sensed by sensor nodes are compared with reference data values and then only the difference data are transmitted. Reference data is taken as the average value of a number of previous sensor readings. The motivation behind SRDA is that it is critical to reduce the number of bits in a transmission because radio communication is the most energy-consuming activity in a sensor node. While data aggregation reduces the number of packets, decreasing the size of the transmitted packets will further improve the energy savings. The downside of ESPDA and SRDA is that they do not allow intermediate nodes to perform data aggregation. That is, sensor data can be aggregated only at the immediate data aggregator which significantly limits the benefit of data aggregation.

2.2 Secure data aggregation using encrypted sensor data

In order to achieve end-to-end data confidentiality and data aggregation together without requiring secret key sharing among data aggregators privacy homomorphic cryptography has been used. A privacy homomorphism is an encryption transformation that allows direct computation on encrypted data.

Concealed Data Aggregation (CDA)

Sensor nodes share a common symmetric key with the base station that is kept hidden from intermediate aggregators. Because of Privacy Homomorphism, Intermediate aggregators do not have to carry out costly decryption and encryption operations. Therefore, data aggregators do not have to store sensitive cryptographic keys which ensure an unrestricted aggregator node election process for each epoch during the wireless sensor network's lifetime.

Domingo-Ferrers asymmetric key based privacy homomorphism is computationally expensive for resource constrained sensor nodes. But this disadvantage is acceptable as CDA advantageously balance the energy consumption. It is preferable to employ CDA's asymmetric key based privacy homomorphism to balance the energy consumption of data aggregators.[7]

Concealed data aggregation protocol(CDAP)

Takes advantage of asymmetric key based privacy homomorphic cryptography to achieve end-to-end data confidentiality and data aggregation together. Asymmetric cryptography based privacy homomorphism incurs high computational overhead which cannot be afforded by regular sensor nodes with scarce resources. CDAP protocol employs a set of resource-rich sensor nodes, called aggregator nodes (AGGNODEs)

Hop by hop secure data aggregation consumes more energy and resources of sensors but provide best security. End to End secure data aggregation is best method for WSN. It does not consume more resources. In end to end SDA, concealed data aggregation (CDA) scheme is efficient way to provide security in WSN. CDA is based on

PH(privacy homomorphism) and it is relatively new in WSN research area. CDA is light weight SDA scheme. CDA provides only confidentiality, other security requirements like integrity, authentication may be added. ESPDA and SRDA are better than CDA but they do not provide data aggregation at intermediate nodes.

Alzaid et al [4] classifies secure data aggregation schemes into two categories(one aggregator model and multiple aggregator model) based on the number of aggregator nodes and the existence of the verification phase. Secure data aggregation (SDA) proposed by [18], it does not provide confidentiality. it is based on hop by hop secure data aggregation, if a parent node and its child nodes are compromised nodes, then data integrity is not guaranteed either. sensor needs buffer for data authentication. Secure information aggregation(SIA) was proposed by [17]. This framework provides resistance against a special type of attack called stealthy attacks aggregate manipulation where the attacker's goal is to make the user accept false aggregation results without revealing its presence to the user. Efficient data aggregation (EDA) proposed by (Castelluccia et al. 2005) uses a modular addition instead of the xor (Exclusive-OR) operation that is found in the stream ciphers. Thus, even if an aggregator is being compromised, original messages can not be revealed by an attacker.

Data transmission accounts for 70% of the energy cost of computation without data aggregation. Data aggregation is helpful to reduce redundant data and number of channel messages transmitted to sink node. Public key cryptography is too expensive for sensor nodes. An adversarial model is proposed by this paper that can be expected in any secure data aggregation scheme. SDA schemes are threaten by passive and active adversaries. (In CDA, passive Adversary exists, the adversary has partial access to the networks, sensor does not keep secret data and adversary classification is light). This paper also explains about Existing SDA schemes and their comparison. Also provide framework for evaluation new schemes.

The general idea of hop-by-hop encrypted data aggregation[5] in WSN is: 1) bootstrapping secure links among the nodes; 2) aggregating data inside the network; 3) authenticating the integrity of aggregation results. The bootstrapping of hop-by-hop encryption can be realized by two methods: 1) pair-wise key distribution among each pair of sensor nodes; 2) group-wise key distribution among a cluster of sensor nodes. In Distributed WSN data confidentiality in aggregation can be protected by pair-wise key distribution schemes. In Hierarchical WSN data confidentiality in aggregation can be protected by group-wise key distribution schemes.

In End-to-end encrypted data aggregation, End-to-end privacy needs to establish a network-wise key between the sink and all the sensor nodes. Network wise key distribution schemes include the master key and public key based solution. Paper also present two frame work: hop by hop and end to end encrypted data aggregation. The framework for end-to-end encrypted data aggregation has higher computation cost on the sensor nodes, but achieves stronger security, in comparison with the framework for hop-by-hop encrypted data aggregation.

The main goal of data aggregation algorithms is to gather and aggregate data in an energy efficient manner so that network lifetime is enhanced. Rajagopalan et al[6] present a survey of data aggregation algorithms in wireless sensor networks. They compare and contrast different algorithms on the basis of performance measures such as lifetime, latency and data accuracy. Data aggregation protocols based on flat network can result in excessive communication and computation burden at the sink node resulting in a faster depletion of its battery power. Hence, in view of scalability and energy efficiency, several hierarchical data aggregation approaches have been proposed. (QOS) aware data aggregation protocols designed to guarantee QOS metrics such as end-to-end reliability and information throughput.

Sensor networks are typically deployed in unsecured areas which makes them vulnerable against physical node capture attacks in which intruders take control of one or more sensor nodes to subvert network's performance. To achieve data aggregation and secure communication together, Ozdemir[7] employs Privacy Homomorphism which

offers end-to-end concealment of data and ability to operate on cipher texts. Privacy homomorphism which is based on symmetric key cryptography is shown to be insecure for chosen plaintext attacks for some specific parameter settings. Therefore, for mission critical networks, asymmetric cryptography based privacy homomorphism should be used instead of symmetric cryptography based privacy homomorphism. However, given the fact that asymmetric cryptography based privacy homomorphism incurs high computational overhead, encryption and aggregation cost of network cannot be afforded by regular senor nodes with scarce resources. Due to high computational overhead, privacy homomorphisms based on asymmetric key algorithms are not feasible for sensor nodes. The privacy homomorphic algorithm introduced by Domingo-Ferrer is symmetric key based and a concealed data aggregation algorithm is proposed using Domingo-Ferrer's privacy homomorphic algorithm. However, the proposed scheme uses secret keys known by all sensor nodes which leads attack. Like If a sensor node is compromised, it can decrypt data of any sensor node which is encrypted by the secret key. Hence asymmetric cryptography based privacy homomorphism is employed where data are encrypted using the base station's public key. concealed data aggregation protocol to improve data aggregation, energy efficiency and bandwidth utilization of sensor networks while providing secure communication.

Westhoff et al[8] provides the brief information about privacy homomorphism with additive and multiplicative PH. PH is based on symmetric and asymmetric key cryptography. Asymmetric PHs are not acceptable in the context of WSNs due to execution times twice as slow than elliptic curve cryptosystems. Encryption schemes like RC5, IDEA or RC4 provide a higher security level and consume much less execution times like any currently available symmetric PH. Unfortunately, applied in WSNs these schemes run into a security/flexibility trade-off.

The Homomorphic Encryption Schemes (HES)[9] provide the most advantage in securing the data on limited-resourced WSN devices by allowing operations to be performed on the ciphertext as if the operations are performed on the plaintext. The other conventional encryption schemes fall short compared with HES because they

demand too much of resources. The HES that secures WSNs data aggregation can conserve the much needed power source while providing the security. When HES is used, the End-to-End encryption is achieved where the information is secured from the source to the destination. this paper conclude that the shorter the key and the message size will yield the better sensor node performance. There are other potential HES such as DF allowing field operations. DF additive and multiplicative and mixed multiplicative also exists. These algorithms could also be feasible for implementation on WSN to find best possible HES for providing WSN security.

Girao et al [10]provide the details about concealed data aggregation for reverse multicast traffic in sensor network. Concealed data aggregation (CDA) schemes that are based on the homomorphic characteristics of a privacy homomorphism (PH) enable end-to-end encryption in wireless sensor networks. They make use of the algebraic properties of the applied PH: Additively homomorphic PHs support additive operations on encrypted data, whereas multiplicatively homomorphic PHs allow for multiplicative operations on the ciphertext. These techniques can be used to achieve the desired security properties in WSNs. Given the fact that communication in WSNs is mainly reverse multicast, nodes that receive sensed data from nodes with a greater distance to the sink node aggregate these values before forwarding the results to a node closer to the sink.CDA-based end-to-end encryption is much more flexible for varying connected backbones over different epochs. With hop-by-hop encryption, only nodes storing the corresponding key can perform the decryption and, thus, aggregate data. With CDA, every node can be elected as an aggregator node, since the aggregating nodes do not need to store the key to operate on the incoming ciphertext message.With CDA, the overall system security level of the WSN increases. Clearly, currently proposed cryptoschemes for WSNs such as RC5, AES, IDEA, or RC4 provide a higher security level and/or require much less execution time compared to any currently available PH. Unfortunately, when applied to WSNs, these schemes run into a security/flexibility tradeoff.

Recently, Castelluccia et al.[11] presented an efficient aggregation of encrypted data in wireless sensor networks. This approach uses different keys per sensor node at the cost of mandatory transmitting of the sensor ID list of the encrypting nodes. Due to the increased message overhead per monitoring node, this approach does not scale for large sensor networks.

2.3 Summary

This Section provides a detailed review of secure data aggregation concept in wireless sensor networks. To give the motivation behind secure data aggregation, first, the security requirements of wireless sensor networks are presented and the relationships between data aggregation concept and these security requirements are explained. Second, an extensive literature survey is presented by summarizing the state-of-the-art secure data aggregation.

the related work for secure data aggregation in WSN and classify them into two general cases: hop-by-hop and end-to-end encrypted data aggregation. First Method consume more energy because aggregator node required cleartext data for data aggregation. then after again perform decryption operation and send it to the base station. Second Method consume less energy comparatively hop-by-hop secure data aggregation. Privacy Homomorphism scheme provide the end to end secure data aggregation.

Chapter 3

Simulation Tool: JIST/SWANS

Researchers in many avenues of science increasingly depend on software simulations to accurately model both realistic phenomena as well as hypothetical scenarios.

3.1 Java In Simulation Time

JIST is an abbreviation of "Java In Simulation Time" and describes a discrete-event simulator developed by Rimon Barr e.a. at Cornell University[12]. The central idea is to transform the Java virtual machine into a scheduler for events by modifying the way how method calls between simulation entities are conducted. On top of this basis, the authors have created SWANS, the "Scalable Wireless Ad hoc Network Simulator", which provides all the mechanisms needed to simulate MANETs. Specifically, the key motivation behind JiST is to create a simulation system that can execute discrete event simulations both efficiently and transparently, yet to achieve this using only a standard systems language and runtime, where:

Efficient refers to a simulation runtime that compares favorably with existing, highly optimized simulation engines; the ability to execute a given simulation program in parallel and optimistically; dynamically optimizing the configuration of the simulation across the available computational resources to improve processing throughput,

and; reasoning about simulation state and event causality constraints to improve throughput.

Transparent implies that simulation programs are automatically transformed to run with simulation time semantics; simulations are instrumented to support the various concurrency, consistency and reconfiguration protocols necessary for efficient sequential, parallel or speculative execution, without requiring programmer intervention or calls to specialized simulation libraries.

Standard denotes writing simulations in a conventional programming language and running these programs over a conventional runtime, where the term conventional describes a commonly used systems programming language, as opposed to a domain-specific language designed explicitly for simulation.

3.1.1 JIST system architecture

The JiST system architecture, depicted in Figure 3.1, consists of four distinct components: a compiler, a bytecode rewriter, a simulation kernel and a virtual machine. One writes JiST simulation programs in plain, unmodified Java, and compiles them to bytecode using a regular Java language compiler. These compiled classes are then modified, via a bytecode-level rewriter, to run over a simulation kernel and to support the simulation time. The simulation program, the rewriter and the JiST kernel are all written in pure Java. Thus, this entire process occurs within a standard, unmodified Java virtual machine (JVM). a key benefit of the JiST approach is that it allows for the efficient execution of simulation programs, within the context of a modern, popular language[13]. JiST combines simulation semantics, found in custom simulation languages and simulation libraries, with modern language capabilities. This design results in a system that is convenient to use, robust and efficient.

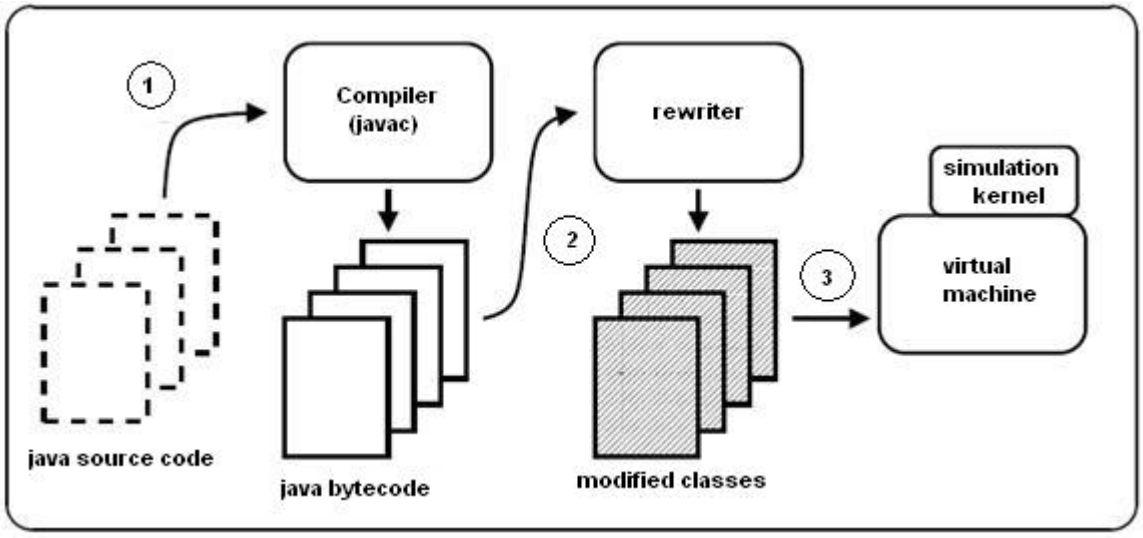

Figure 3.1: The JiST system architecture - simulations are compiled (1),then dynamically instrumented by the rewriter (2) and finally executed(3).The compiler and virtual machine are standard Java language components. Simulation time semantics are introduced by the rewriting classloader and supported at runtime by the Java-based simulation kernel.

In JiST simulations, objects are implemented as ordinary Java objects which means that all the advantages of Java can also be used in JiST. Simulation components are called Entities and are marked with the interface JistAPI.Entity. Each entity has its own entity simulation time t_e which determines,at what time calls to other entities are scheduled. An Entity progresses its t_e by calling the JistAPI.sleep() function. On the other hand, the system time t_s determines what scheduled calls are called in what time-step.

Before execution, the classes of a JiST simulation are transformed using a bytecode modifier. In this step, calls to public methods of entities are asynchronously decoupled. This means that a call will return immediately, the call is queued in an Event queue and the corresponding code in the invoked object is executed when the called entity reaches the same simulation time of the calling entity.

Listing 1: A simple JiST Entity

```
1. import jist.runtime.JistAPI;
2.
3. class hello implements JistAPI.Entity
4. {
5.     public static void main(String[] args)
6.     {
7.         System.out.println("simulation start");
8.         hello h = new hello();
9.         h.myEvent();
10.    }
11.
12.    public void myEvent()
13.    {
14.        JistAPI.sleep(1);
15.        myEvent();
16.        System.out.println("hello world,t=" 17 +JistAPI.getTime());
17.    }
18. }
```

JIST produces the output:

> simulation start

> hello world, t=1

> hello world, t=2

> etc.

Listing 1 shows a basic example of a JiST entity. the beginning($t_s = 0$) , the simulation time for all entities is implicitly set to ($t_e = 0$) Then, method myEvent() is called. This method first increases the local simulation time counter by one time

unit($t_e = 1$) After that, method myEvent() calls itself again. As the simulator engine decouples this call to a public entity method, this is no recursion! Instead, the call returns immediately, the println() method is called and the myEvent() methods returns. As no other method calls are scheduled for($t_s = 0$)simulation time advances to ($t_s = 1$)and the pending call to myEvent() is processed which starts over again. Scheduled calls for($t_s = 2$) are processed and the loop continues infinitely.

Note that non-public methods in entity objects are called directly. Internally, JiST replaces all entity references by proxy objects, so called separators, so that entity calls are never executed directly. Moreover, calls to public methods of entities are not allowed to return any values.

In essence, our method call has been transformed into a simulation event on the hello entity. It is scheduled and invoked by the simulation kernel in simulation time. Note, also, that hello is an entity, not a regular object, since it implements the JistAPI.Entity interface. You may also observe, from the output, that the sleep serves to advance the simulation time of the system. And, clearly, there is no stack overflow. Thus, although we reuse the Java language, as well as its compiler and virtual machine, we extend the object model of the language and execution semantics of the virtual machine, so that simulation programs will run in simulation time.

3.1.2 Object model and execution semantics

JiST simulation programs are written in plain Java, an object-oriented language. As in any object-oriented language the entire simulation program comprises numerous classes that collectively implement the logic of the simulation model. During its execution, the state of the program is contained within individual objects. These objects communicate by passing messages, represented as object method invocations in the language. In order to facilitate the design of simulations, JiST extends this traditional programming model with the notion of simulation entities.In the program

code, entities are defined as instances of classes that implement the JistAPI.Entity interface. Although entities are regular objects to the JVM,they serve to logically encapsulate application objects, as shown in Figure 3.2 and demarcate independent simulation components. Thus, every object should be logically contained within an entity. Conversely, the state of an entity is the combined state of all the objects reachable from it. To enforce this simulation partitioning and prevent object commu-

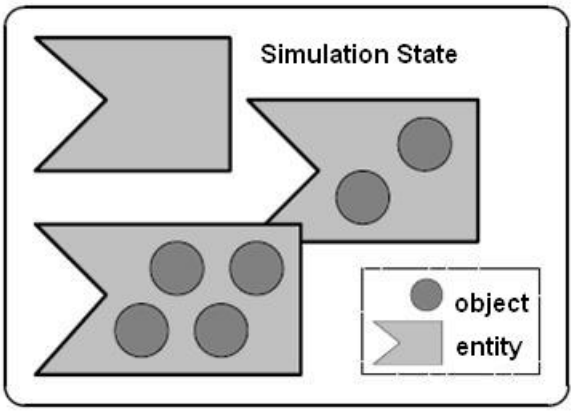

Figure 3.2: Simulation programs are partitioned into entities along object boundaries. Thus, entities do not share any application state and can independently progress through simulation time between interactions.

nication across entity boundaries, each (mutable) object in the system must belong to a single entity and must be entirely encapsulated within it. In Java, this means that all references to an object must originate either directly or indirectly from a single entity. This condition suffices to ensure simulation partitioning, because Java is a safe language.

The JiST kernel manages a simulation at the granularity of its entities. Instructions and method invocations within an entity follow the regular Java control flow and semantics, entirely opaque to the JiST infrastructure. The vast majority of this code is involved with encoding the logic of the simulation model and is entirely unrelated to the notion of simulation time. All the standard Java class libraries are available and behave as expected. In addition, the simulation developer has access to a few

basic JiST primitives, including functions such as getTime and sleep, which allow for interactions with the simulation kernel.

The role of the simulation developer, then, is to codify the simulation model in regular Java and to partition the state of the simulation not only into objects, but also into a set of independent entities along reasonable application boundaries. The JiST infrastructure will transparently execute the program efficiently, while retaining the simulation time semantics.

3.1.3 Simulation time invocation

here i have describe step-by-step execution of hello world program. Omitting unnecessary complexity in the interests of clarity, below are the basic operations that occur when one executes:

java jist.runtime.Main hello

a. The JVM is loaded and initialized.
b. The JiST simulation kernel is loaded into the JVM and initialized. Among other activities, a dynamic class loader is installed, so that JiST verification and rewriting of application occurs on demand.
c. The kernel creates an anonymous, bootstrap simulation entity. The kernel then creates and en queues a bootstrap simulation event at time t = 0, which will invoke the simulation program's main method with any command-line arguments, when the simulation event loop begins.
d. The kernel enters the simulation event loop and our hello simulation begins.
e. Process the bootstrap event at time t = 0:
    (1) Line 7: Output "simulation start".
    (2) Line 8: Create and initialize a new instance of hello. Note that hello is an entity, not a regular object, because it implements the JistAPI.Entity

interface on line 3. Thus, at runtime, local variable h will actually contain a separator object.

(3) Line 9: Perform a simulation time invocation of myEvent. In other words, schedule an event at the current local simulation time t = 0 to invoke myEvent on the entity referenced by h.

(4) End of event.

f. Process the myEvent event at time t = 0.

(1) Line 14: Advance the local simulation time by one time quantum to t=1.

(2) Line 15: Perform a simulation time invocation of myEvent. That is, schedule an event at current local simulation time t = 1 to invoke myEvent on the current entity.

(3) Line 16: Emit the "hello world" message, with the current local simulation time.

(4) End of event.

g. Process the next scheduled myEvent event at time t = 1; 2; 3; ... as above.

3.2 Scalable Wireless Ad hoc Network Simulator

SWANS is a Scalable Wireless Ad hoc Network Simulator built atop the JiST platform, a general-purpose discrete event simulation engine. SWANS was created primarily because existing wireless network simulation tools are not sufficient for current research needs. SWANS also serves as a validation of the virtual machine-based approach to simulator construction. SWANS is a componentized, virtual machine-based simulator. SWANS significantly outperforms ns2 and GloMoSim, both in time and space[14].

SWANS is a collection of components to simulate ad hoc networks based on the JiST simulation engine. Figure 3.3 shows the architecture of typical SWANS simulations. Nodes consist of several entities that are linked together and represent the different stack elements of a network application. The lowest radio entity of each node is connected to a common field entity which is responsible for delivering packets passed down the stack to other nodes in wireless transmission range.

Packets are handed from stack to stack by simply passing references between entities, duplicating the packet via clone() where appropriate. As calling a method and passing a reference comes at virtually no cost, SWANS stays extremely fast, even with large node stacks and high data traffic load. SWANS comes with a basic set of simulation entities covering basic scenario elements like different fading and pathloss models, several modules for positioning and mobility of nodes, radio noise models, an 802.11b MAC layer, an IPv4 layer, DSR, AODV, and ZRP MANET routing protocols, TCP and UDP transport layers. As an application, one can use e.g. a CBR traffic generator or an application proxy that allows any Java network application that uses the socket interface to be used atop of SWANS.

The field implementation of SWANS uses an optimization that uses hierarchical binning in order to further speed up simulations. Whenever a node delivers a packet

to the field, the field must deliver the packet to a set of other nodes that get affected by this packet after considering fading, gain and pathloss. A small subset of all nodes will be in reception range, whereas some other nodes get affected by interference above the sensitivity threshold. Depending on the field size and the transmission power, the majority of nodes will not be affected however, which is exploited by the hierarchical binning approach.

3.2.1 Design Highlights

The SWANS software is organized as independent software components, called entities, that can be composed to form complete wireless network or sensor network simulations, as shown in below Figure.

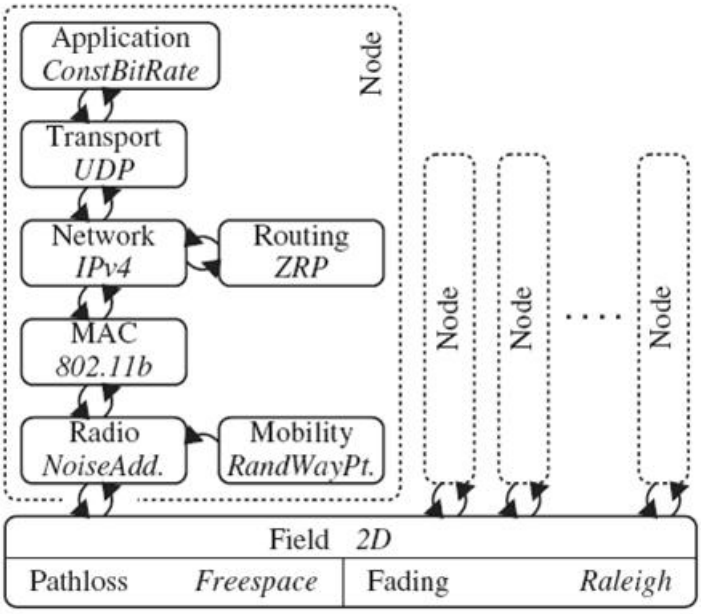

Figure 3.3: The SWANS simulator consists of event-driven components that can be configured and composed to form wireless network simulations.

The SWANS simulator runs atop JiST, a Java-based discrete-event simulation engine that combines the benefits of the traditional systems-based (e.g., ns2) and languages-based (e.g., GloMoSim) approaches to simulation construction. JiST con-

verts a standard virtual machine into a simulation platform by embedding simulation time directly into the Java object model and into the virtual machine execution semantics. JiST extends the Java object model with the notion of simulation entities. The simulation entities represent components of a simulation that can progress independently through simulation time, each encapsulating a disjoint subset of the simulation state. Simulation events are intuitively represented as method invocations across entities. This programming model is convenient, efficient, and flexible. SWANS is able to simulate much larger networks and has a number of other advantages over existing tools. We leverage the JiST design within SWANS to (1) achieve high simulation throughput; (2) reduce its memory footprint; and (3) run existing, Java-based network applications over simulated networks. In addition,SWANS implements a technique called hierarchical binning to modelwireless signal propagation in a scalable manner. The combination of these attributes leads to a flexible and highly efficient simulator.

3.2.2 Components

This section enumerates the various SWANS components that are available.

Physical Layer

The SWANS physical layer components are responsible for modeling signal propagation among radios as well as the mobility of nodes. Radios make transmission downcalls to the simulation "field" and other radios on the "field" receive reception upcalls from it, if they are within range of the signal. Both the pathloss and fading models are functions that depend on the source and destination radio locations. The SWANS radio receives upcalls from the field entity and passes successfully received packets on to the link layer. It also receives downcalls from the link layer entity and passes them on to the field for propagation. Both radios are half-duplex, as in 802.11b. Radios are parameterized by frequency, transmission power, reception sensitivity and

threshold, antenna gain, bandwidth and error model. Error models include bit-error rate and signal-to-noise threshold.

Link Layer

The SWANS link layer entity receives upcalls from the radio entity and passes them to the network entity. It also receives downcalls from the network layer and passes them to the radio entity. The link layer entity is responsible for the implementation of a chosen medium access protocol and for encapsulating the network packet in a frame. Link layer implementations include IEEE 802.11b and a "dumb" protocol. The 802.11b implementation includes the complete DCF functionality, with retransmission, NAV and backoff functionality. It does not include the PCF (access-point), fragmentation or frequency hopping functionality found in the specification. This is on par with the GloMoSim and ns2 implementations. The "dumb" link entity will only transmit a signal if the radio is currently idle.

Network Layer

The SWANS network entity receives upcalls from the link entity and passes them to the appropriate packet handler, based on the packet protocol information. The SWANS network entity also receives downcalls from the routing and transport entities, which it enqueues and eventually passes to the link entity. Thus, the network entity is the nexus of multiple network interfaces and multiple network packet handlers. The network interfaces are indexed sequentially from zero. The packet handlers are associated with IETF standard protocol numbers, but are mapped onto a smaller index space, to conserve memory, through a dynamic protocol mapper that is shared across the entire simulation. Each network interface is associated with a packet queue, which can handle packet priorities and perform RED. The packets are dequeued and sent to the appropriate link entity using a token protocol to ensure that only one packet is transmitted at a time per interface. The network layer sends packets to the routing entity to receive next hop information and allows the routing entity to peek

at all incoming packets. It also encapsulates message with the appropriate IP packet header. The network layer uses an IPv4 implementation. Loopback and broadcast are implemented.

Routing Layer

The routing entity recieves upcalls from the network entity with packets that require next-hop information. It also receives upcalls that allow it to peek at all packets that arrive at a node. It sends downcalls to the network entity with next-hop information when it becomes available. SWANS implements the Zone Routing Protocol (ZRP), Dynamic Source Routing (DSR) and ad hoc On-demand Distance Vector Routing (AODV).

Transport Layer

The SWANS transport entity receives upcalls from the network entity with packets of the appropriate network protocol and passes them on to the appropriate registered transport protocol handler. It also receives downcalls from the application entity, which it passes on to the network entity. The two implemented transport protocols are UDP and TCP, which encapsulate packets with the appropriate packet headers. UDP socket, TCP socket and TCP server socket implementations actually exist within the application entity.

Application Layer

At the top of our network stack, we have the application entities, which make downcalls to the transport layer and receive upcalls from it, usually via SWANS sockets or streams that mimic their Java equivalents. The most generic and useful kind of application entity is a harness for regular Java applications. One can run standard, unmodified Java networking application atop SWANS. These Java applications operate within a context that includes the correct underlying transport implementation for the particular node. Thus, these applications can open regular communication

sockets, which will actually transmit packets from the appropriate simulated node, through the simulated network. SWANS implements numerous socket and stream types in the jist.swans.app.net and jist.swans.app.io packages. Applications can also connect to lower-level entities. The heartbeat node discovery application, for example, operates at the network layer. It circumvents the transport layer and communicates directly with the network entity.

Common for all Layers

There are various interfaces that are common across a number of SWANS layers and tie the system together. The most important interface of this kind is Message. It represents a packet transfered along the network stack and it must be timeless (or immutable). Components at various layers define their own message structures. Many of these instances recursively store messages within their payload, thus forming a message chain that encodes the hierarchical header structure of the message. Other common elements include a node, node location, protocol number mapper and miscellaneous utilities.

3.2.3 Summary

The SWANS simulator runs over JiST, combining the traditional systems-based (e.g., ns2) and languages-based (e.g., GloMoSim) approaches to simulation construction. SWANS is able to simulate much larger networks and has a number of other advantages over existing tools.
We leverage the JiST design within SWANS to:

Achieve high simulation throughput:

Simulation events among the various entities, such as packet transmissions, are performed with no memory copy and no context switch. The system also continuously profiles running simulations and dynamically performs code inlining, constant

propagation and other important optimizations, even across entity boundaries. This is important, because many stable simulation parameters are not known until the simulation is running. Greater than 10x speedups have been observed.

Save memory:

Memory is critical for simulation scalability. Automatic garbage collection of events and entity state not only improves robustness of long-running simulations by preventing memory leaks, it also saves memory by facilitating more sophisticated memory protocols. For example, network packets are modeled as immutable objects, allowing a single copy to be shared across multiple nodes. This saves the memory (and time) of multiple packet copies on every transmission. A different example of memory savings in SWANS is the use of soft references for storing cached computations, such as routing tables. These routing tables can be automatically collected, as necessary, to free up memory.

Run standard Java applications:

SWANS can run existing Java network applications, such as web servers and peer-to-peer applications, over the simulated network without modification. The application is automatically transformed to use simulated sockets and into a continuation-passing style. The original network applications are run within the same process as SWANS, which increases scalability by eliminating the considerable overhead of process-based isolation.

Chapter 4

Implementation of Energy Model for JIST/SWANS

Sensor networks consist of very large number of resource constrained nodes, and to properly evaluate the performance of these networks a scalable ad-hoc wireless network simulator is needed. The JIST/SWANS is a highly scalable wireless network simulator with reported results for a million node network. However, it lacks an energy model[15]. This work presents the design and implementation of an energy model for the JiST/SWANS simulator, which enables it to account for the energy consumed by wireless ad hoc and sensor networks protocols.

An energy model maintains the current energy level and simulates the energy consumption for network events.

The energy model receives up calls from the MAC layer whenever there is a change in the radio mode of operation. The MAC layer has been modified to account for the time duration at any of the radio states. The MAC layer supplies the energy model with the time duration, and the energy model then calculates the energy consumed during that time depending on the state and then deducts it from the current energy of the node. The model marks the node as exhausted when its energy drops below a predefined threshold level.

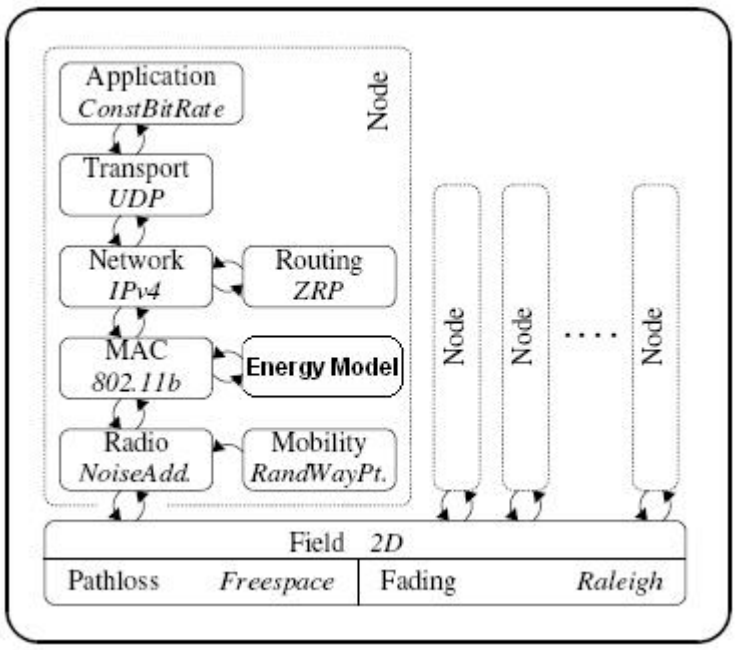

Figure 4.1: Energy model entity connected with Mac layer

In designing the energy model, I consider all radio states namely transmit, receive, idle, and sleep. The power dissipated by the radio can be obtained from the radios specification sheets.

4.1 Energy model implementation

The MAC 802.11b implementation was modified to enable the calculation of the time duration in each state and to call the energy model functions every time the state of the radio changes.

Creating and Initializing Energy Model Object:

During the initialization phase, an energy model object is created for each node that individually maintains the nodes current energy level. These energy models assign an initial energy to their nodes, from which the energy consumed in each radio state will be deducted during the nodes lifetime.

Decrementing energy used in Transmit mode:

To determine the time duration that the radio spent in the transmit mode, we utilized the MAC layer function *transmitTime()* that is responsible for calculating the duration for the packet transmission. Every time the radio changes to the transmit state, the time duration for the packet transmission is calculated and the function *decrementTransmitEnergy()* is called . This function is part of energy model and it calculates the energy consumed for the specific transmission and deducts it from the current energy level for the node. If the energy level drops below the preset threshold value, the energy model will notify the simulator that the node is dead and will no longer be considered as a part of the network.

Decrementing energy used in Idle mode:

To account for the duration in which a node was in idle mode, we have implemented timers. Every time a node changes to idle state, the current time stamp is saved in a variable. When the node changes its state again from idle to any of the other states, a new timestamp is also saved. The difference between these two timestamp values is the idle time for the node. *decrementIdleEnergy()* uses this idle time duration to calculate the energy consumed in the idle mode. Here again, if the energy level drops below the threshold value, the node is marked as exhausted.

Decrementing energy used in Receive mode:

Timers are used to account for the receive time duration. *decrementReceiveEnergy()* uses this receive time duration to calculate the energy consumed for that time duration and deduct this amount from the current energy level and update current energy value. The new energy level is then tested against the threshold.

Decrementing energy used in Sleep mode:

Each time the radio goes into the sleep mode it is operating in the low power

mode where most of the radio circuitry is switched off. Since our goal is to simulate sensor networks, it is necessary to add an energy model with the ability to calculate the energy consumed in the sleep mode. The function *decrementsleepEnergy()* is implemented like the other functions that uses the sleep duration to calculate the energy consumed in that state.

Current energy at a node at any given time:

Each time the radio changes state the energy mo del updates its information about the current energy level of the node. The goal is to transform the JiST/SWANS to a design and evaluation tool for wireless sensor network protocols. For their correct operation, some of these protocols need to know the current energy level of the node. To facilitate that, the energy model exposes the function *getCurrentEnergy()* that can be called from the other layers to query the node for the current energy level.

4.2 Pseudo code for changes in MAC802.11

The simulator driver during the initialization Phase:

Get number of nodes for simulation;
Get simulation time;
Set nodes locations;
Create the simulation field and its parameters;
Add nodes at the specific locations and initialize them;
Assign each node an initial energy;

At the MAC layer after the simulation starts:

At simulation start time set the radio/MAC mode to idle;
Start idle timer;

```
If message is available for transmission
If start idle timer ! = 0
{
   Stop idle timer ;
   Calculate idle duration=stop idle timer - start idle timer;
   Reset start idle timer;
   Call energy Model to Calculate energy used in idle;
}
Set radio mode/MAC mode to transmit;
Calculate the packet transmission time
{
   Get packet size;
   Package transmission time = packet size/bandwidth;
   Total transmission time = synchronization delay + package transmission time;
}
Call energy model to calculate energy used in transmit mode;
Is radio mode idle?
Start idle timer;
If message is available for reception
If start idle timer ! = 0
{
   Stop idle timer;
   Calculate idle duration=stop idle timer - start idle timer
   Call energy model to calculate energy used in idle;
}
Set radio mode/MAC mode to receive;
Start receive timer;
Receive data packet;
Stop receive timer;
```

Calculate receive duration = stop receive timer - start receive timer;

Call energy model to calculate energy used in receive;

Is radio mode idle?

Start idle timer;

At the Energy Model during simulation:

Call from MAC to calculate and deduct Idle Energy:

Energy used in idle mode = idle power * time duration in idle mode;

Energy remaining = current energy - energy used in idle mode;

Call from MAC to calculate and deduct transmission energy:

Energy used in transmit mode = transmit power * time duration in transmit mode;

Energy remaining = current energy - energy used in transmit mode;

Call from MAC to calculate and deduct receive energy:

Energy used in transmit mode = receive power * time duration in receive mode;

Energy remaining = current energy - energy used in transmit mode;

Example

Application: Heartbeat

Simulation Parameter:

Number of nodes: 3

Field Area: 100m X 100m

Simulation Time: 5 Second

Simulation Run:

swans driver.heartbeat 3 100 5

Simulation Output:

nodes = 3, field size = 100m x 100m, time = 5 seconds
Creating simulation nodes... done.
the current energy level is =99.99
IN transmitTime
the energy used for this transmission is = 0.00624
the current energy level is =99.99
IN receive Data
the energy used for this reception is = 0.00624
the current energy level is =99.98
IN receive Data
the energy used for this reception is = 0.00624
the current energy level is =99.98
(1) found neighbour: 2, t=2.52 sec
(0) found neighbour: 2, t=2.52 sec
...
(0) found neighbour: 1, t=3.15 sec
(2) found neighbour: 1, t=3.15 sec
...

4.3 Summary

This chapter shows the implementation of an energy model for the JiST/SWANS simulator, which enables it to account for the energy consumed by wireless sensor networks. Energy consumption accounting is done at all possible radio states, i.e., transmit, receive, idle, and sleep, and considering the different energy costs associated with each of them. The energy model we introduced here will enable the nodes to maintain and provide such information for the JiST/SWANS simulator.

Chapter 5

Implementation of Cluster architecture

5.1 Clustering in Wireless Sensor Networks

It is widely accepted that the energy consumed in one bit of data transfer can be used to perform a large number of arithmetic operations in the sensor processor. Moreover in a densely deployed sensor network the physical environment would produce very similar data in close-by sensor nodes and transmitting such data is more or less redundant. Therefore, all these facts encourage using some kind of grouping of nodes such that data from sensor nodes of a group can be combined or compressed together in an intelligent way and transmit only compact data. This can not only reduce the global data to be transmitted and localized most traffic to within each individual group, but reduces the traffic and hence contention in a wireless sensor network. This process of grouping of sensor nodes in a densely deployed large-scale sensor network is known as clustering. In clustering each group agrees on a central node, called the cluster head, which is responsible for gathering the sensory data of all group members, aggregating it and sending it to the base station(s). One major goal of clustering is to allow in network pre-processing (aggregation or compression),

assuming that cluster heads (and other intermediate nodes) collect multiple data packets and relay only one aggregated/compressed packet.[1]

Cluster Architecture

Figure 5.1 shows the cluster architecture of wireless sensor networks. which (a) Organizes the sensor nodes into clusters (b) Each cluster is governed by a cluster-head (c) Only heads send messages to a BS (d) Suitable for data fusion (e) Self-organizing.

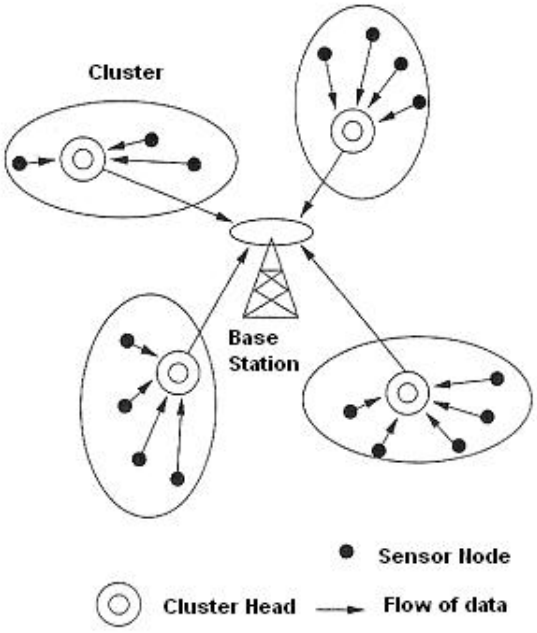

Figure 5.1: Clustered network architecture

Clustering is particularly useful for :

- Applications that require scalability to hundreds or thousands of nodes. (need for load balancing and efficient resource utilization.)
- Applications requiring efficient data aggregation.
- One-to-many, many-to-one or one-to-all (broadcast) communication.

5.2 Implementation steps

Here we have develop Single hop with clustering. The basic code of heartbeat application was modified to implement the clustering strategy with dynamic selection of clusters. First, assign ID number to each sensor nodes which is start from zero. Sensor Node ID Zero work as Base Station.

To create a cluster architecture using heartbeat Application:

- Randomly select nodes which are working as cluster heads. This cluster heads are responsible to receive data from their neighbors.
- Each cluster head send message to all other nodes.
- Those nodes which are in the range of cluster head they send the back message to cluster head and join with this cluster head for further process.
- Cluster Head send the data to the Base Station.
- Measure Times to create a cluster Heads.
- Measure energy usage by any cluster heads.

5.3 Simulation Result

5.3.1 Simulation Environment

Application : Heartbeat
Field Length(meter) : 1000m x 1000m
Number of Nodes : 50, 100
Simulation Time : 10 Second
Radio Frequency : 2.4 GHz
Radio Bandwidth : 1Mb/s
Transmission strength (units: dBm) :
TRANSMIT_DEFAULT = 15.0; (approximately 100 meter)

5.3.2 Simulation Result for 50 nodes

Table 5.1: Cluster for 50 nodes and Field length 1000 $meter^2$

No. of Nodes	Field Length	Cluster Head(ID)	No. of Neighbour Nodes	Time Measure in Second	Energy Usage
50	$1000m^2$	15	8	5.76	99.97
		31	3	10.96	99.70
		30	8	13.61	99.56
		2	6	13.83	99.29
		3	5	15.34	99.07
		1	9	19.74	98.87
		27	6	24.61	98.57

5.3.3 Simulation Result for 100 nodes

Table 5.2: Cluster for 100 nodes and Field length 1000 $meter^2$

No. of Nodes	Field Length	Cluster Head(ID)	No. of Neighbour Nodes	Time Measure in Second	Energy Usage
100	$1000m^2$	18	8	2.21	99.65
		22	9	6.86	99.61
		50	8	8.81	99.51
		2	9	17.30	99.46
		72	10	21.01	98.42
		48	14	23.85	97.37
		65	14	25.03	97.26
		14	6	25.93	99.17
		79	14	27.45	97.07
		74	9	32.13	98.98

5.4 Summary

Wireless sensor networks (WSNs) have attracted significant attention over the past few years. A growing list of civil and military applications can employ WSNs for increased effectiveness; especially in hostile and remote areas. Examples include disaster management, border protection. In these applications a large number of sensors are expected, requiring careful architecture and management of the network. Grouping nodes into clusters has been the most popular approach for support scalability in WSNs.

In wireless sensor network all leaf nodes send their data to center node called as cluster head and center node send data to the base station. this over all architecture is a cluster based architecture. Cluster-based approaches organize the network in to several clusters each with a cluster-head which responsible for aggregating data locally before forwarding towards sink. this chapter shown the implementation steps of clustering using heartbeat application and shown simulation results.

Chapter 6

Data Aggregation in WSN

Wireless sensor network have recently been the focus of many research efforts and emerged as an important new area in wireless technology. this chapter presents an overview of data aggregation and Simulation results of Data aggregation technique.

6.1 In-Network Aggregation

In a typical sensor network scenario, different node collect data from the environment and then send it to some central node or sink which analyze and process the data and then send it to the application. But in many cases, data produced by different node can be jointly processed while being forwarded to the sink node. So in-network aggregation deals with this distributed processing of data within the network.

We define the In network aggregation process as follows:
In-network aggregation is the global process of gathering and routing information through a multi-hop network, processing data at intermediate nodes with the objective of reducing resource consumption (in particular energy). Data aggregation and fusion widely used in WSNs, which can reduce the communication and increase the network lifetime.[16]

One natural approach is for sensors to send their values to certain special nodes, i.e., aggregators. Each aggregator then condenses the data prior to sending it on.the aggregators can either be special or regular sensor nodes. here i have select Cluster Head as aggregator. In this setting, since sensors have very limited capabilities, aggregation must be simple and not involve any expensive or complex computations. Ideally, it would require only a few simple arithmetic operations, such as additions or multiplications.
There are two approaches for in-network aggregation:

- ***with size reduction*** refers to the process of combining & compressing the data packets received by a node from its neighbors in order to reduce the packet length to be transmitted or forwarded towards sink. As an example,when a node receives two packets which have a spatial correlated data then it is worthless to send both packets. Instead of that one should apply any function like AVG, MAX, MIN and then send a single packet. This approach saving a lots of energy.

- ***without size reduction*** refers to the process merging data packets received from different neighbors in to a single data packet but without processing the value of data. As an example, two packets may contain different physical quantities (like temperature & humidity) and they can be merged in to a single packet [8].

In cluster-based approach, whole network is divided in to several clusters. Each cluster has a cluster-head which do the role of aggregator. aggregator node aggregate the data which is received from cluster members(leaf node) and then transmit the result to sink node(Base Station). following figure shows the data aggregation process. Figure 6.1 shows the data aggregation process model of wireless sensor network. there are two cluster head and their neighbors shown. cluster heads are send the data to the base station.

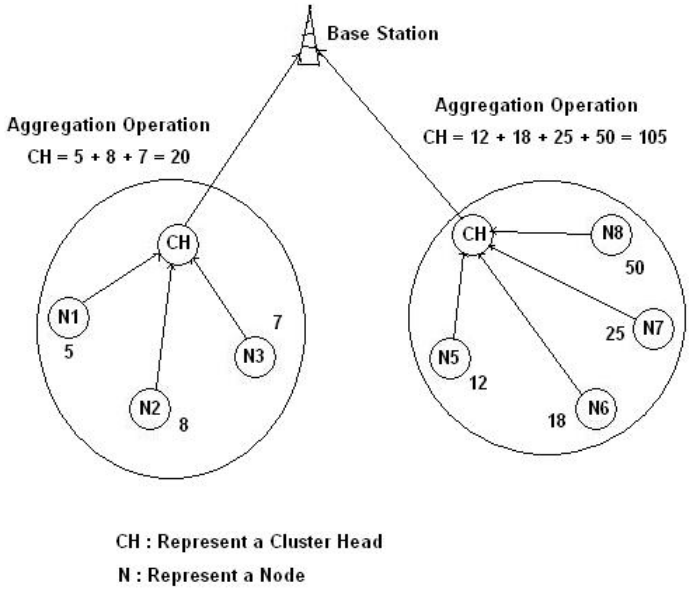

Figure 6.1: Data aggregation process Model

6.1.1 Data Aggregation Steps

- **In Round One**
  - randomly placed n numbers of sensor nodes. sensor node zero(0) work as base station. each sensor node sense the data from environment.
  - randomly selected Cluster Heads(aggregators) receive data from their neighbors.
  - first, Data aggregation combines data by using functions such as suppression (eliminating duplicates).
  - second,Aggregator perform Aggregation operation on Received Data.

- **In Round Two**
  - Each aggregator node send data(aggregation result) to the base station.
  - Base Station Use this received aggregation data for further process.
  - Energy Model Measure the Energy usage of Cluster Head.

6.2 Simulation of Data Aggregation

Simulation setup

A square field of 1000m X 1000m is taken where 50 nodes and 100 nodes are randomly deployed in different simulation environment. Five nodes are designated as cluster-head (CH) for 50 nodes,Ten nodes are designated as cluster-head (CH) for 100 nodes. node zero(ID) is designated as Base Station. Energy Model measure the energy usage of each node during the Receive, Transmit, sleep and Idle Mode. Parameter of Energy Model is define in the *Energy model package of JIST/SWANS simulator.*

Parameters used are:

Simulation Time : 10 Sec
MAC type : 802.11
Radio Frequency :
FREQUENCY_DEFAULT = 2.4 GHz
Radio Bandwidth :
BANDWIDTH_DEFAULT = 1Mb/s
Transmission strength (units: dBm) :
TRANSMIT_DEFAULT = 15.0; (approximately 100 meter)

Simulation Run:

swans driver.heartbeat 50 1000 10

swans driver.heartbeat 100 1000 10

6.2.1 Cluster Head Vs Energy Usage Graph

Fig. 6.2 and 6.3 shows the Difference between cluster head Energy usage with aggregation process and without aggregation process. In without aggregation process, energy usage is less comparatively with aggregation process at Cluster Head. In without aggregation process when data send to the base station from cluster head it consume more energy. In with aggregation process when data send to the base station from cluster head it consume less energy because less communication required.

6.2.2 Simulation Results

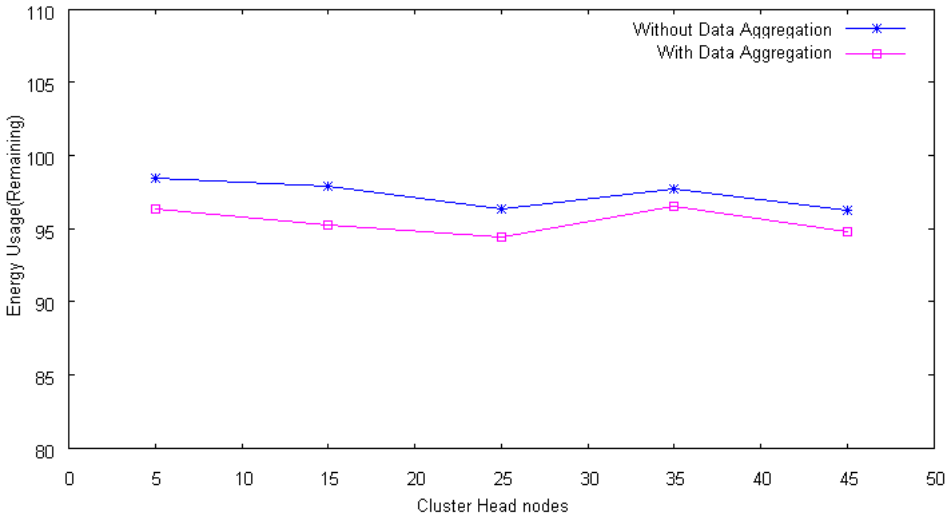

Figure 6.2: Energy Usage for 50 nodes

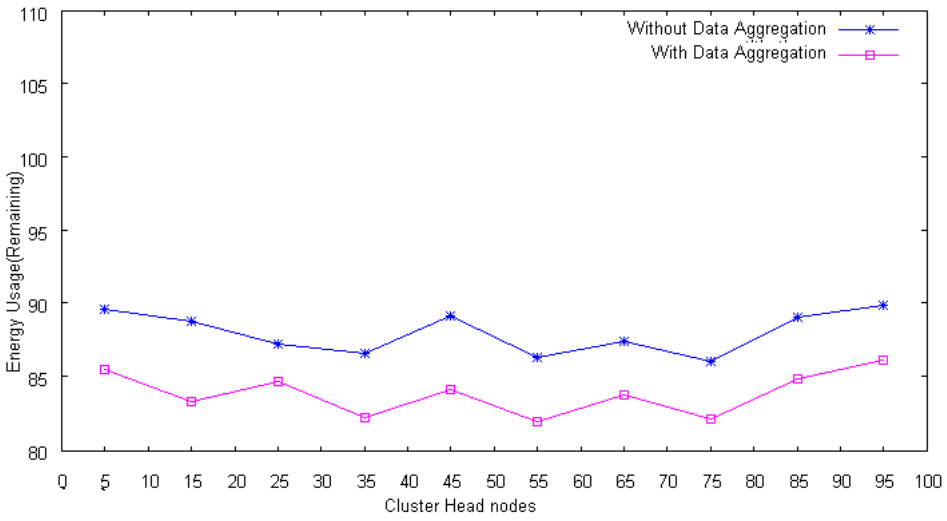

Figure 6.3: Energy Usage for 100 nodes

6.3 Summary

One of the main design aspects for sensor network architectures is energy efficiency, to keep the network operational as long as possible.Therefore,aggregation techniques are an essential building block,as they aim at reducing the number of transmissions required for data collection which,in turn, reduces energy consumption. I have implemented the data aggregation technique which first remove the duplicate values and then perform data aggregation at Cluster Head. also found the minimum and maximum values of each cluster heads.

Chapter 7

Secure Data Aggregation

Wireless sensor networks are energy constrained and bandwidth limited, reducing communications between sensors and base stations has a significant effect on power conservation and bandwidth utilization. Aggregated sensor networks serve this purpose by introducing designated nodes called aggregators that provide efficient data collection and transmission. An aggregator can sense its own data while aggregating received results from children nodes, which in turn may be leaf sensors or aggregators as well. Aggregated wireless sensor networks provide better power conservation and efficient use of communication channels but also introduce additional security concerns. Most existing schemes for data aggregation are subject to attack. Once a single node is compromised, it is easy for an adversary to inject false data into the network and mislead the aggregator to accept false readings. Because of this, the need for secure data aggregation is raised and its importance needs to be highlighted.

A general definition for secure data aggregation is *"the efficient delivery of the summary of sensor readings that are reported to an off-site user in such a way that ensures these reported readings have not been altered"*[17]

The data security requirements in the WSNs are similar to those in traditional networks . However, there are some unique specifications that can only be found

in WSNs that required more attention during design process. below provide the information about requirements for data aggregation security.

a. ***Data Confidentiality*** ensures that information content is never revealed to anyone who is not authorized to receive it. it is minimal security requirement of WSN.
b. ***Data Integrity*** ensures that the content of a message has not been altered, either maliciously or by accident, during transmission process.
c. ***Data Freshness*** ensures that the data are recent and that no old messages have been replayed to protect data aggregation schemes against replay attack.
d. ***Data Availability*** ensures that the network is alive and that data are accessible.
e. ***Authentication*** ensures that reported data is the same as original one. receiver verify that received message is sent by the claimed sender or not.

The security Methods classified according to their underlying encryption schemes into end-to-end and hop-by-hop secure data aggregation.[10]

In **hop-by-hop secure data aggregation,** leaf sensor nodes Encrypt the data and then send it to the data aggregator. data aggregator first decrypt the data and then perform the aggregation operation on clear text data. next Encrypt result of data aggregation and then send to the base station. This increases the computational demand at the inner nodes(aggregator) a lot though they are the most important ones and should save on energy as much as possible.

Thus it would be desirable to find a way that allows to process data without having to decrypt it while preserving the content. The aggregating node does not necessarily need to interpret the data, it only has to be able to work with it. A concept which meets the above requirements is called a privacy homomorphism and has been introduced by Rivest, Adelman and Dertouzos. This PH Method Used into end-to-end secure data aggregation.

In **End-to-End secure data aggregation,** aggregator perform aggregation operation on decrypted values which is received from leaf nodes. so there is no need to perform decryption operation again for data aggregation. following Figure shows End-to-End Secure Data aggregation.

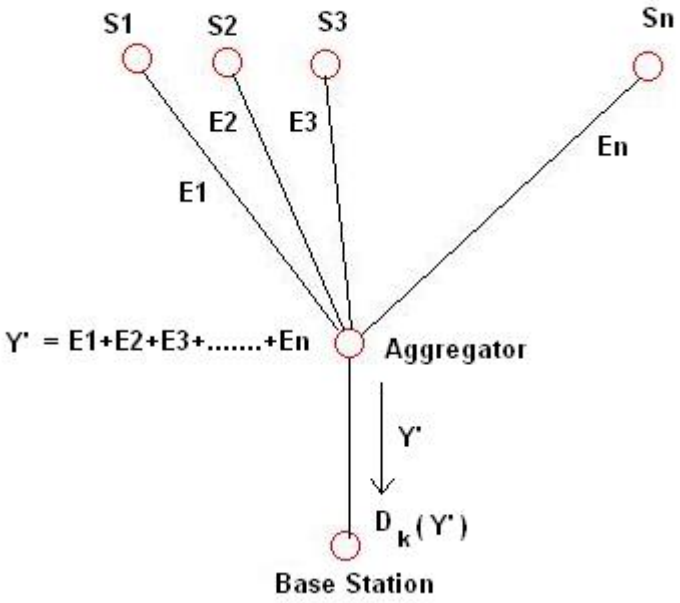

Figure 7.1: End to End Secure Data Aggregation

7.1 Privacy Homomorphism

A privacy homomorphism is an encryption function which allows operations like e.g., additions or multiplications on the encrypted data. The result will yield an encrypted codeword which is similar to the codeword that would be obtained by applying the operation on the cleartext first and encrypting the result afterwards. In more mathematical terms we can say that the encryption function distributes over the operation. Additions or multiplications are of particular interest in this context. An instance of such a privacy homomorphism (PH form now on) was suggested by Domingo-Ferrer in A Provably Secure Additive and Multiplicative Privacy Homomor-

phism. A privacy homomorphism (PH) is an encryption transformation that allows direct computation on encrypted data.It is a **symmetric encryption** scheme which uses the same key for encryption and decryption. Let Q and R denote two rings, + denote addition and X denote multiplication on both. Let K be the keyspace. We denote an encryption transformation E : K X Q → R and the corresponding decryption transformation D : K X R → Q. Given a, b ∈ Q and k ∈ K we term additively homomorphic and multiplicatively homomorphic.

$$a + b = D_k(E_k(a) + E_k(b)) \tag{7.1}$$

$$a * b = D_k(E_k(a) * E_k(b)) \tag{7.2}$$

RSA is a multiplicative PH. while Domingo-Ferrer presented an additive and multiplicative PH which is a symmetric scheme and secure against chosen ciphertext attacks. asymmetric PH are not acceptable in the context of WSNs due to execution times.

7.1.1 An additive Privacy Homomorphism

here i have describe the parameter settings, encryption transformation and decryption transformation of the PH proposed by Domingo-Ferrer. The PH is probabilistic which means that the encryption transformation involves some randomness that chooses the ciphertext corresponding to a given cleartext from a set of possible ciphertexts.[11]

Parameter Settings

First, we will go into details on the generation of the key. According to Domingo-Ferrer a large integer g has to be found which holds the following two properties:

- It should consist of a large number of divisors. This can easily be ensured by repeatedly multiplying prime numbers. The same prime can also be multiplied several times. In effect, g simply is a product of integers.

- The resulting number should be chosen such that many integers r $< g$ can be found which have a so-called modulo inverse. The large g should be chosen such that many integers r can be found for which an r_{inv} exists so that r x r_{inv} mod $g = 1$. E.g., 2 is modulo invertible with regard to the divisor 9, since 2 x 5 = 10 mod 9 = 1

For the g chosen above we need a g' which divides g with no remainder. Domingo-Ferrer states that g x $\log_{g'}$ was a hint to the security level. Furthermore, a particular r $< g$ has to be chosen with r x r_{inv} mod $g = 1$. Finally, an integer d >= 2 is needed. The secret key is k = (r, g').

Encryption

a cleartext a will be decomposed into d such that a = $\sum_{j=1}^{d}$ a_j mod g'
Compute

$$E_k(a) = (a_1 r \ mod \ g, a_2 r^2 \ mod \ g, ..., a_d r^d \ mod \ g) \tag{7.3}$$

so we get, $(e_1(a), e_2(a), ...e_d(a))$

Decryption

multiplying r_{inv} in the first component, ${r^2}_{inv}$ in the second (and so on) and applying modulo g unless the result is smaller than g.

$$D_k(E_k(a)) = (e_1(a) r_{inv} \ mod \ g, e_2(a) {r^2}_{inv} \ mod \ g, ..., e_d(a) {r^d}_{inv} \ mod \ g) \tag{7.4}$$

Of course, in order to obtain a again the components a_j have to be added:

$$D_k(E_k(a)) = \sum_{j=1}^{d} a_j \ mod \ g'. \tag{7.5}$$

ENCRYPTED DATA AGGREGATION

Sensors S_1 to S_n encrypt their data s_1 to s_n resulting in $s'_1 = E_{(r,g')}(s_1)$ to $s'_n = E_{(r,g')}(s_n)$ before transmitting data to the A. Then, A operates on the encrypted data and computes $y' = f(s'_1,...,s'_n)$ Subsequently, the aggregator A transmits y' to the R which decrypts the y' and derives the accumulated data $y = D_{(r,g')}(y')$ Figure 7.2 illustrates the approach.

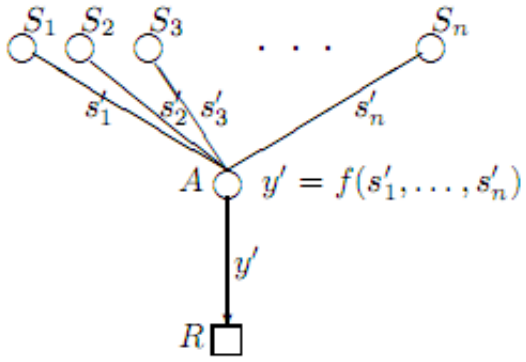

Figure 7.2: Concealed Data Aggregation for WSNs with PH

the concealed data aggregation for a WSN with the reference PH works as follows:

a. We consider (r, g') to be known to S_1,...,S_n and at the R.

b. The values d and g are public and known to A.

c. At S_i with $1 \leq i \leq n$: split S_i into d parts such that $s_i = \sum_{j=1}^{d} a_j \bmod g'$ and Perform Encryption operation.

d. At A: Compute on base of the additive and multiplicative homomorphic operations + and x the aggregation function $y' = f(s'_1,...,s'_n)$ and transmit y' to R.

e. At R: Compute Decryption operation.

Example: Sum Computation

Suppose two sensor nodes send their data(in encrypted form) to the aggregator and aggregator perform sum operation on encrypted data and send resultant sum operation to the base station. base station perform decryption operation to retrieve cleartext value. sum computation shown below:

$g = 416$, $g' = 208$, d = 2, r = 7, $r_{inv} = 119$

a = 12, b = 14 (Data as cleartext) (sensor nodes)

First decompose a and b as follows:

12 = 5+7 , 14 = 4+10

Both sensor nodes perform the following encoding:

E(12) = ($(5x7^1)$ mod 416, $(7x7^2)$ mod 416)

E(12) = (35,343)

E(14) = ($(4x7^1)$ mod 416, $(10x7^2)$ mod 416)

E(14) = (28,490)

Aggregator receive this encryption value and perform sum operation.

E(12) + E(14) = ((35,343) + (28,490)) mod 416

E(12) + E(14) = (63,1)

Next, Base Station receive secure aggregation results from different aggregators and perform Decryption operation.

D(E(12)+E(14)) = $63x119^1$ mod 416 + $1x119^2$ mod 416

D(E(12)+E(14)) = 9 + 17 = 26

7.2 Implementation of Secure Data Aggregation Algorithm

Parameter:
1. Public Key: integer d>=2,large integer g
2. Secret Key: (r,g')
3. large g should be chosen such that many integer r can be found for which r_{inv} exists so that r x r_{inv} mod $g = 1$
4. for the g chosen above we need a g' which divides g with no remainder.

Encryption:
1. cleartext a will be decomposed into d such that a = $\sum_{j=1}^{d}$ a_j mod g'
2. $E_k(a) = (a_1 r \ mod \ g, a_2 r^2 \ mod \ g, ..., a_d r^d \ mod \ g)$
3. so we get, $(e_1(a), e_2(a), ...e_d(a))$

Decryption: $D_k(E_k(a)) = (e_1(a) r_{inv} \ mod \ g, e_2(a) {r^2}_{inv} \ mod \ g, ..., e_d(a) {r^d}_{inv} \ mod \ g)$

Aggregation:
1. Scalar addition modulo g
2. Suppose we have $(e_1(a), e_2(a), ...e_d(a))$ and $(c_1(a), c_2(a), ...c_d(a))$ so scalar addition is, $e_1(a) + c_1(a)$ mod g and so on.

It is a symmetric algorithm that requires the same secret key for encryption and decryption. The aggregation is performed with a key that can be publicly known, i.e., the aggregator nodes do not need to be able to decrypt the encrypted messages. However, it is required that the same secret key is applied on every node in the network that needs to encrypt data. For very secure parameter combinations (d > 100), the messages become very big. However,with reasonable parameters it also fits the needs of constrained devices.

7.3 Implementation steps

I have implemented the secure data aggregation scheme which is based on end to end secure data aggregation in JIST/SWANS simulator. following steps shows the implementation of secure data aggregation.

- randomly place the n number of sensor nodes. sensor node ID zero work as cluster head and its position at the center of field area of simulation
- randomly select some nodes as a Cluster Head. leaf nodes join with cluster heads.
- leaf node first encrypt the data and send the data packet to the cluster head.
- each cluster head perform secure data aggregation process on received encrypted values from their leaf nodes. each cluster head also found the minimum and maximum values from encrypted values and send to the base station.
- resultant data of secure data aggregation now send to the base station.
- base station now perform decryption operation on resultant data of SDA.
- Measure energy usage of each cluster heads.

7.4 Summary

Data aggregation is a widely used technique in wireless sensor networks. The security issues, data confidentiality and integrity, in data aggregation become vital when the sensor network is deployed in a hostile environment. Generally, two methods can be used for secure data aggregation in WSN: hop-by-hop encrypted data aggregation and end-to-end encrypted data aggregation. In the former, data is encrypted by the sensing nodes and decrypted by the aggregator nodes. The aggregator nodes then aggregate the data and encrypt the aggregation result again. At last the sink node gets the final encrypted aggregation result and decrypt it. In the latter, the intermediate aggregator nodes havent decryption keys and can only do aggregations on the encrypted data.

Chapter 8

SDA: Simulation Results

8.1 Simulation of Secure Data Aggregation

Simulation setup

A square field of 1000m X 1000m is taken where 50 nodes and 100 nodes are randomly deployed in different simulation environment. Three nodes are designated as cluster-head (CH) for 50 nodes,Five nodes are designated as cluster-head (CH) for 100 nodes. node zero(ID) is designated as Base Station. Energy Model measure the energy usage of each node during the Receive, Transmit, sleep and Idle Mode. Parameter of Energy Model is define in the *Energy model package of JIST/SWANS simulator.*

Parameters used are:

Simulation Time : 10 Sec

MAC type : 802.11

Radio Frequency :

FREQUENCY_DEFAULT = 2.4 GHz

Radio Bandwidth :

BANDWIDTH_DEFAULT = 1Mb/s

Transmission strength (units: dBm) :

TRANSMIT_DEFAULT = 15.0; (approximately 100 meter)

Simulation Run:

swans driver.heartbeat 50 1000 10

swans driver.heartbeat 100 1000 10

8.1.1 Simulation results

Here we have taken two types of simulation results. First static location of cluster head in both data aggregation and secure data aggregation process and second dynamic location of cluster head in both data aggregation and secure data aggregation process.

following table shows the resulting data of secure data aggregation. Simulator assigns ID to each sensor nodes. Here I have developed privacy homomorphism security algorithm. Each sensor data divide into two parts. Each part of the value is encrypted and send to the data aggregator. Now data aggregators perform the pair wise aggregation operation (SUM) on these decrypted values and send to the base station. Next, base station performs decryption operation and receives clear text data.

We have also found the minimum and maximum values from the encrypted values of each cluster head. This result is also shown into the table.

Table 8.1: Secure Data aggregation result for 50 Nodes and 3 cluster head

Cluster Head (ID)	Neighbour Nodes (ID)	Encryption part1 E1	Encryption part2 E2	Received Data	Energy Usage
5	18	28	98	6	99.63
	29	154	98	24	99.27
	34	91	98	15	98.58
	3	168	98	26	98.18
	42	161	98	25	97.56
	44	623	98	91	97.02
	28	224	98	34	96.55
	38	581	98	85	95.13
	1	546	98	80	94.54
	10	623	98	91	94.02
	43	385	98	57	92.85
	11	504	98	74	91.53
	16	91	98	15	90.89
	14	21	98	5	90.73
	46	504	98	74	89.8
	17	693	98	101	89.49
	33	574	98	84	87.88
	4	434	98	64	87.2
Cluster Head 5 Received data: 6405 and 1764(encrypted value)					
Base Station 0 Received data: 771(cleartext value)					
Maximum Value: 101					
Minimum Value: 5					

Cluster Head (ID)	Neighbour Nodes (ID)	Encryption part1 E1	Encryption part2 E2	Received Data	Energy Usage
25	45	448	98	66	99.09
	31	98	98	16	97.24
	32	14	98	4	95.69
	49	315	98	47	93.57
	40	287	98	43	91.95
	24	630	98	92	90.43
	20	651	98	95	90.03
	9	315	98	47	88.26
	41	518	98	76	86.9
	23	595	98	87	86.37
	30	595	98	87	85.89
	15	413	98	61	85.73
45	8	238	98	36	98.41
	25	154	98	24	96.04
	47	357	98	53	93.06
	2	238	98	36	92.25
	7	329	98	49	86.5
Cluster Head 25 Received data: 4879 and 1176(encrypted value)					
Base Station 0 Received data: 587(cleartext value)					
Maximum Value: 95					
Minimum Value: 4					
Cluster Head 45 Received data: 1316 and 490(encrypted value)					
Base Station 0 Received data: 162(cleartext value)					
Maximum Value: 53					
Minimum Value: 24					

Table 8.2: Secure Data aggregation result for 100 Nodes and 5 cluster head

Cluster Head (ID)	Neighbour Nodes (ID)	Encryption part1 E1	Encryption part2 E2	Received Data	Energy Usage
20	7	35	98	7	99.91
	39	441	98	65	98.59
	86	399	98	59	96.05
	63	455	98	67	94.3
	56	392	98	58	89.03
	61	483	98	71	88.49
	27	266	98	40	86.42
	58	413	98	61	81.66
	31	35	98	7	80.63
	19	427	98	63	78.22
	23	308	98	46	77.66
	25	546	98	80	75.55
	30	315	98	47	73.03
	4	217	98	33	71.74
	22	259	98	39	69.11
	66	413	98	61	68.41
	75	154	98	24	67.04
	48	98	98	16	64.99
	92	77	98	13	63.46
Cluster Head 20 Received data: 5733 and 1862(encrypted value)					
Base Station 0 Received data: 789(cleartext value)					
Maximum Value: 80					
Minimum Value: 7					

Cluster Head (ID)	Neighbour Nodes (ID)	Encryption part1 E1	Encryption part2 E2	Received Data	Energy Usage
35	38	413	98	61	98.56
	11	252	98	38	97.88
	80	434	98	64	96.85
	95	406	98	60	91.18
	91	322	98	48	90.89
	28	91	98	15	85.84
	46	336	98	50	85.68
	6	371	98	55	75.26
	33	7	98	3	72.69
	9	553	98	81	66.89
	1	266	98	40	65.36
55	24	196	98	30	99.24
	60	140	98	22	94.24
	2	455	98	67	92.64
	17	84	98	14	89.55
	3	287	98	43	82.73
	8	189	98	29	81.91
	42	203	98	31	81.42
	45	413	98	61	78.91
	99	14	98	4	78.33
	90	490	98	72	76.51
	76	364	98	54	75.89
	96	378	98	56	75.08
	32	336	98	50	73.41
	14	91	98	15	69.26
	74	98	98	16	64.81
Cluster Head 35 Received data: 3451 and 1078(encrypted value)					
Base Station 0 Received data: 515(cleartext value)					
Maximum Value: 81					
Minimum Value: 3					
Cluster Head 55 Received data: 3738 and 1470(encrypted value)					
Base Station 0 Received data: 564(cleartext value)					
Maximum Value: 72					
Minimum Value: 4					

Cluster Head (ID)	Neighbour Nodes (ID)	Encryption part1 E1	Encryption part2 E2	Received Data	Energy Usage
75	41	434	98	64	97.68
	52	455	98	67	96.36
	47	518	98	76	94.6
	15	140	98	22	93.78
	69	413	98	61	89.92
	57	14	98	4	86.66
	93	315	98	47	82.75
	40	203	98	31	81.1
	62	490	98	72	78.91
	49	252	98	38	76.65
	81	63	98	11	74.21
	51	518	98	76	73.31
	20	266	98	40	71.04
	94	511	98	75	66.7
	77	45	98	67	65.3
95	97	413	98	61	97
	36	98	98	16	95.8
	35	427	98	63	93.1
	88	511	98	75	88.99
	53	28	98	6	83.37
	71	21	98	5	82.3
	84	364	98	54	69.82
	85	357	98	53	67.95
Cluster Head 75 Received data: 4637 and 1470(encrypted value)					
Base Station 0 Received data: 608(cleartext value)					
Maximum Value: 76					
Minimum Value: 4					
Cluster Head 95 Received data: 2219 and 784(encrypted value)					
Base Station 0 Received data: 333(cleartext value)					
Maximum Value: 75					
Minimum Value: 5					

8.1.2 Cluster Head Vs Energy Usage graph

Following graph shows the energy usage at the cluster head during the Data aggregation and Secure data aggregation process. Figure 8.1 and 8.2 shows the energy usage at the static location of the cluster head and Figure 8.3, 8.4 and 8.5 shows the energy usage at the dynamic location of the cluster head. The graph suggests that secure data aggregation consumes more energy than data aggregation at each cluster heads.

Static Location of Cluster Head

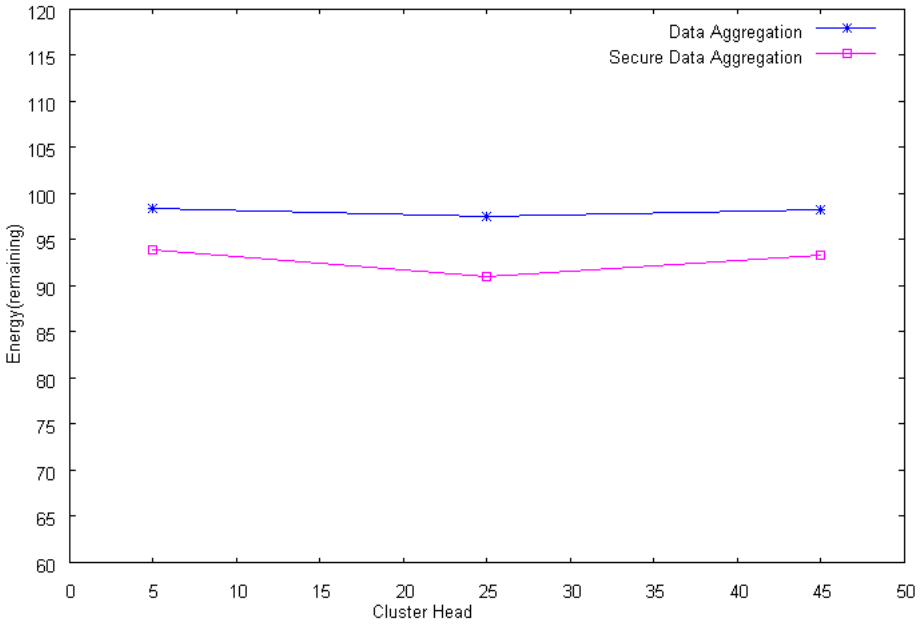

Figure 8.1: Remaining Energy Graph...50 sensor nodes, 1000m^2 area, 3 Cluster Head

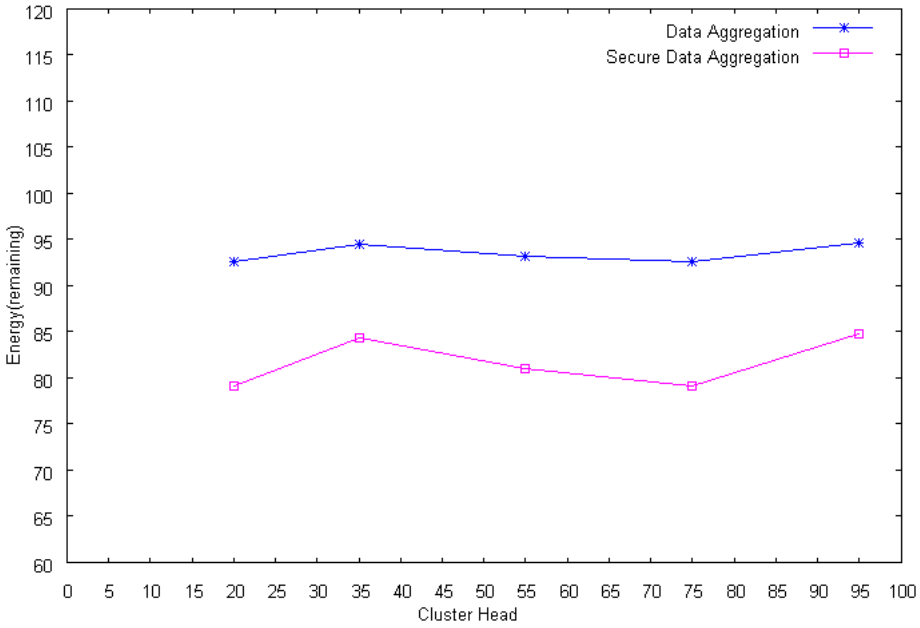

Figure 8.2: Remaining Energy Graph...100 sensor nodes, 1000m^2 area, 5 Cluster Head

Dynamic Location of Cluster Head

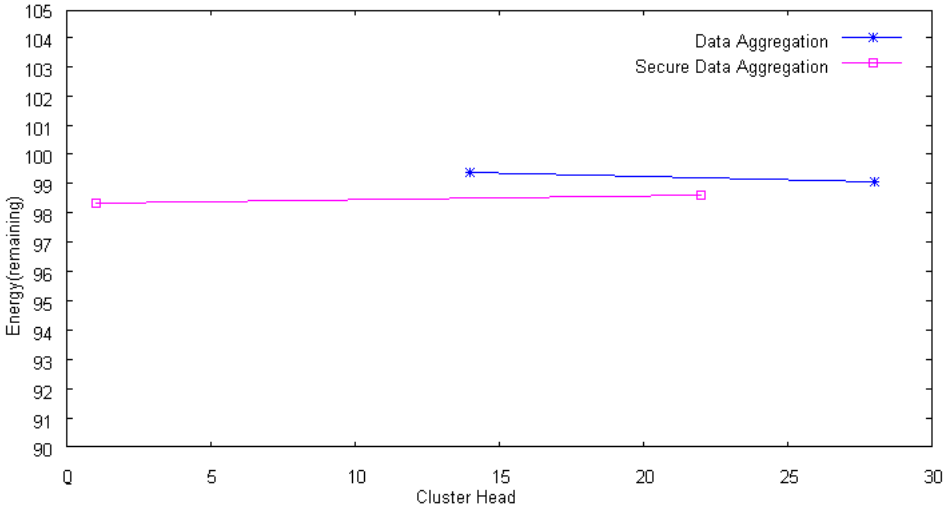

Figure 8.3: Remaining Energy Graph...30 sensor nodes, 1000m^{2}area, 2 Cluster Head

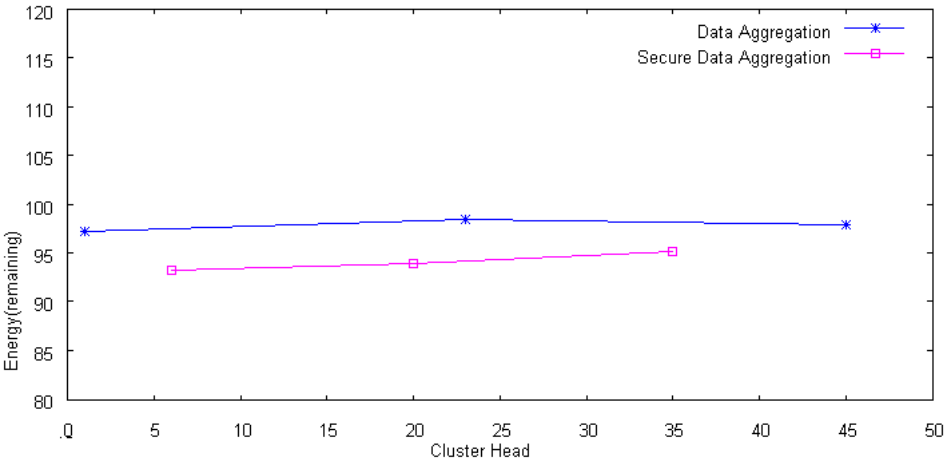

Figure 8.4: Remaining Energy Graph...50 sensor nodes, 1000m^{2}area, 3 Cluster Head

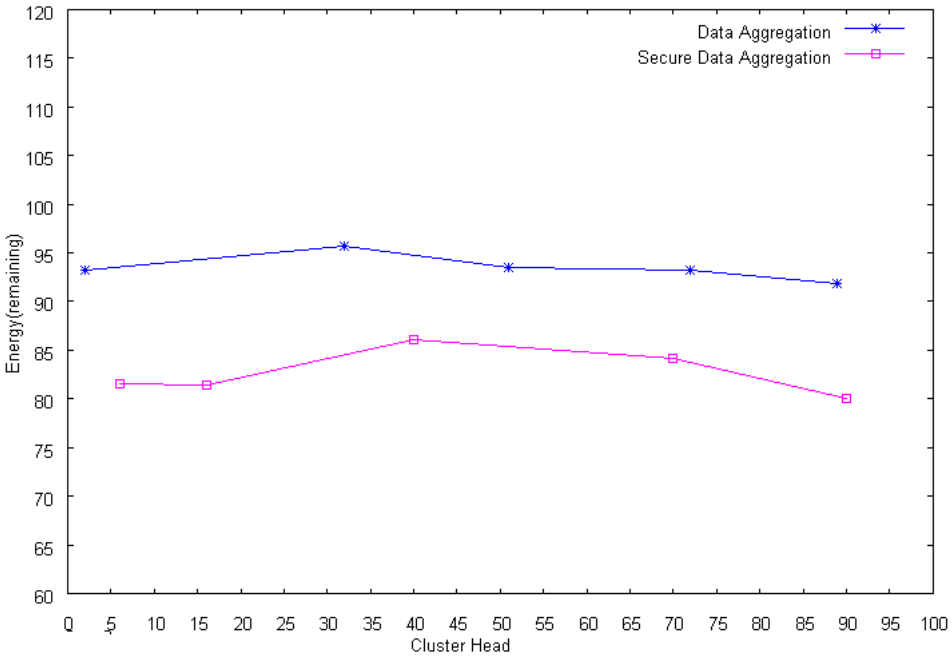

Figure 8.5: Remaining Energy Graph...100 sensor nodes, 1000m^{2}area, 5 Cluster Head

It can be observed that the dynamic allocation of cluster head strategy selects any node as cluster head as per the given measures. As the number of sensor nodes are increased the residual energy of the cluster heads starts dropping accordingly as it has to do more work compared to the plain aggregation.

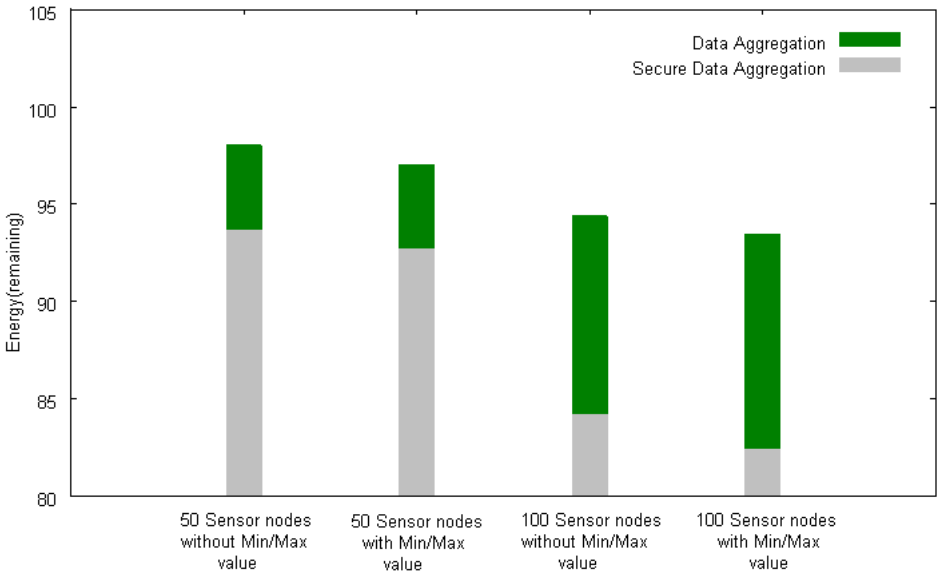

Figure 8.6: Remaining Energy graph...With and Without Minimum/Maximum value.

Fig. 8.6 shows the Remaining Energy of sensor nodes with 50 and 100 sensor nodes configured to collect and propagate minimum and maximum values from the cluster to the base station. The readings were taken for the same set of nodes and clusters but without collecting and passing minimum and maximum values. The minimum and maximum values are collected from the cluster so as to identify the key points where some action may be required. The energy consumption is higher of this strategy as the number of bits to be transmitted by the sensor nodes and cluster head is higher.

Chapter 9

Conclusion and Future Scope

9.1 Conclusion

The primary use of wireless sensor networks (WSNs) is to collect and process data. 70% of energy consumption is due to data transmission. Because of the hostile environment and unique properties of wireless sensor network all raw data samples are not directly sent to the sink node, but data aggregation is applied. Also, Wireless sensor nodes are often deployed in an open environment such as a battlefield or other similar applications. Data confidentiality and integrity are vital issues in such conditions, hence secure aggregation is required. Hop by hop secure data aggregation is resource consuming compared with end to end secure encrypted data aggregation.

By using traditional symmetric key cryptography algorithms, it is not possible to achieve end-to-end confidentiality and in-network data aggregation together. If the application of symmetric key based cryptography algorithms is combined with data aggregation, then the messages must be encrypted hop-by-hop. This means that, to perform data aggregation, intermediate nodes have to decrypt each received message, then aggregate the messages according to the corresponding aggregation function, and finally encrypt the aggregation result before forwarding it. Clearly, this is not an energy efficient way of performing secure data aggregation and it may result in consid-

erable delay. In addition, this process requires neighboring data aggregators to share secret keys for decryption and encryption. In order to achieve end-to-end data confidentiality and data aggregation together without requiring secret key sharing among data aggregators, privacy homomorphism cryptography has been used. End-to-end encryption solutions for convergecast traffic in wireless sensor networks that support in-network processing at forwarding intermediate nodes is known as Concealed Data Aggregation. Sensor nodes share a common symmetric key with the base station that is kept hidden from intermediate aggregators.

The major contribution of this work is the provision of end to end encryption for reverse multicast traffic between the sensors and the base station. Data aggregators carry out aggregation functions that are applied to cipher texts (encrypted data). This provides the advantage that intermediate aggregators do not have to carry out costly decryption and encryption operations. Therefore, data aggregators do not have to store a sensitive cryptographic key which ensures an unrestricted aggregator node election process for each epoch during the wireless sensor networks lifetime. Aggregation of data is very crucial in sensor networks because of the scarcity of energy. Due to the deployment in open environments sensor nodes are vulnerable to number of attacks so aggregation process must be supplemented with strong security support. End to end secure data aggregation protocols with privacy homomorphism are most suitable for WSNs as there is no compromise with data privacy at intermediate levels. Privacy homomorphism based aggregation seems most promising method for secure aggregation in WSNs as it tries to reduce the energy consumption with provision of data confidentiality. It also increases system flexibility against changing routes.

In our Implementation we have used Jist/SWANS simulator. The basic code of heartbeat application was modified to implement the clustering strategy with dynamic selection of clusters. The energy model was hooked in to the Jist/SWANS simulator. Separate application was developed for both plain aggregation and secure aggregation with Privacy Homomorphism method integration. The energy consumption of the network increases in the secure aggregation method compared to the plain

aggregation. As the number of nodes increases the energy level drops further of cluster heads. We have tested our methods with 30 sensor nodes to 100 sensor nodes in the area of 1000 square meters. The implemented method is more secure and less energy consuming compared to hop by hop secure aggregation.

9.2 Future work

Although method provides the robust security cover to the data moving in the network the energy consumption is still an issue as the network grows. The protocol can be made more scalable and fine tuned with the multi level clustering where the cluster can have two to three level tree so the cluster can cover more number of nodes with lower energy consumption. The clusters are presently chosen without considering the location aspects of the other nodes. So the energy consumption can be further controlled with location aware clustering which will cover almost entire network in a uniform way. We believe that our work will encourage other researchers to consider the vital problem of secure information aggregation in sensor networks.

Appendix A

List of web sites

1. http://jist.ece.cornell.edu/
2. http://www.cs.technion.ac.il/ gabik/Jist-Swans/index.html
3. http://en.wikipedia.org/wiki/Wireless_sensor_network
4. http://www.ece.northwestern.edu/ocg474/SIDnet.html

Appendix B

List of Publication

Paper titled **"Secure Data Aggregation in Wireless Sensor Networks - A Survey"** published at "International Conference on SENSORS and RELATED NETWORKS (SENNET-09)" organized by "VIT University, Vellore,Tamilnadu" held during December 07-10, 2009.

References

[1] Pranay Tiwari,"Data Aggregation in Cluster-based Wireless Sensor Networks",*Thesis,IIIT Allhabad,* July 2008.

[2] N. P. Mahalik,"Sensor Networks and Configuration",*book,Springer*, 2007.

[3] Suat Ozdemir, Yang Xiao, "Secure data aggregation in wireless sensor networks: A comprehensive overview",*Elsevier B.V. doi:10.1016/j.comnet.2009.02.023*,2009

[4] Hani Alzaid, Ernest Foo, Juan Gonzalez Nieto, "Secure Data Aggregation in Wireless Sensor Network: A Survey", *ACSC2008,Australia*, January 2008.

[5] Yingpeng sang, Hong shen,"Secure data aggregation in wireless sensor networks: a survey",*PDCAT '06,seventh international conference,*Dec 2006.

[6] Ramesh Rajagopalan, Pramod K. Varshney,"Data aggregation techniques in sensor networks: A survey",*Communications Surveys & Tutorials,IEEE,* vol. 8, no. 4, pp. 4863, 2006.

[7] Suat Ozdemir, "Concealed Data Aggregation in Heterogeneous sensor networks using privacy homomorphism",*ICPS07: IEEE International Conference on Pervasive Services,*Istanbul, Turkey, pp. 165-168,2007.

[8] Joao Girao,Dirk Westhoff, Markus Schneider,"CDA:concealed data aggregation for reverse multicast traffic in wireless sensor networks",*40th International Conference on Communications,IEEE ICC 2005,*May 2005.

[9] Levent Ertaul, Johan H. Yang,"Implementation of Domingo Ferrers a new privacy homomorphism in securing wireless sensor networks(WSN)",2007.

[10] Dirk westhoff,Joao Girao,Mithun acharya,"concealed data aggregation for reverse multicast traffic in sensor networks:encryption,key distribution and routing adaption", *IEEE Trans. Mobile Comput. 5 (10) 1417-1431,*2006

[11] Castelluccia,Aldar chan,gene tsudik,"Efficient and Provably Secure Aggregation of Encrypted Data in Wireless Sensor Networks", *ACM Transactions on Sensor Networks,Vol.5,No.3,Article 20,*May 2009.

[12] R. Barr, Z. J. Haas, R. van Renesse, "JiST: Embedding Simulation Time into a Virtual Machine." *5th EUROSIM Congress on Modeling and Simulation, Paris,France,* September 2004

[13] Rimon Barr,Zygmunt J. Haas,Robbert van Renesse,"JiST: An efficient approach to simulation using virtual machines",*Software Practice and experience, 00:1-7*, 2004

[14] Rimon Barr,"SWANSScalable Wireless Ad hoc Network Simulator", *User Guide*,March 19, 2004

[15] Trishla Sutaria, Imad Mahgoub, Ali Humos, Ahmed Badi,"Implementation of an Energy Model for JiST/SWANS Wireless Network Simulator",*Sixth International Conference,ICN '07,*April 2007

[16] Elena Fasolo,Michele Rossi,Jorg Widmer,Michele Zorzi,"In-network aggregation techniques for wireless sensor networks: A Survey",*IEEE Communication magazine*,April 2007

[17] Przydatek,Song D.,Perrig A.,"SIA: Secure Information Aggregation in Sensor Networks",*ACM, pp. 255265*,2003

[18] Hu L.,Evans D.,"Secure aggregation for wireless network",*SAINT Workshops,IEEE Computer Society, pp. 384394*,2003

Index