A NOVEL SCALABLE KEY MANAGEMENT PROTOCOL FOR WIRELESS SENSOR NETWORKS

by

Musfiq Rahman

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
March 2013

DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "A NOVEL SCALABLE KEY MANAGEMENT PROTOCOL FOR WIRELESS SENSOR NETWORKS" by Musfiq Rahman in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Dated: March 26, 2013

External Examiner: Dr. Sagar Naik

Research Supervisor: Dr. Srinivas Sampalli

Examining Committee: Dr. Nur Zincir-Heywood

Dr. Vlado Keselj

Dr. Keith Johnson

Departmental Representative: Dr. Malcolm Heywood

DALHOUSIE UNIVERSITY

DATE: March 26, 2013

AUTHOR: Musfiq Rahman

TITLE: A NOVEL SCALABLE KEY MANAGEMENT PROTOCOL FOR WIRELESS SENSOR NETWORKS

DEPARTMENT OR SCHOOL: Faculty of Computer Science

DEGREE: Ph.D. CONVOCATION: May YEAR: 2013

Signature of Author

Table of Contents

List of Tables

List of Figures

Abstract

Wireless Sensor Networks (WSNs) are ad-hoc networks consisting of tiny battery-operated wireless sensors. The sensor nodes are lightweight in terms of memory, computation, energy and communication. These networks are usually deployed in unsecured, open, and harsh environments, where it is difficult for humans to perform continuous monitoring. Consequently, it is very crucial to provide security mechanisms for authenticating data among sensor nodes. Key management is a pre-requisite for any security mechanism. Efficient distribution and management of keys in WSNs is a challenging task. Many standard key establishment techniques have been proposed using symmetric cryptosystems. Unfortunately, these systems often fail to provide a good trade-off between memory and security and since WSNs are lightweight in nature, these cryptosystems are not feasible. On the other hand, public key infrastructure (PKI) is infeasible in WSNs because of its continuous requirement of a trusted third party and heavy computational demands for certificate verification.

Pairing-Based Cryptography (PBC) has paved the way for how parties can agree on keys without any interaction. It has relaxed the requirement of expensive certificate verification on PKI systems. In this thesis, we propose a new hybrid identity-based non-interactive key management protocol for WSNs, which leverages the benefits of both symmetric key based cryptosystems and pairing-based cryptosystems. The proposed protocol is scalable, suits many applications and can be deployed in multiple types of networks without modifications. We also provide mechanisms for key refresh when the network topology changes. A security analysis is presented to prove that the scheme is resilient to many types of attacks. To validate our scheme, we have implemented it on Crossbow TelosB motes running TinyOS and analyzed the performance in terms of memory, communication, computation and energy consumption. The results indicate that our scheme can be deployed efficiently to provide high level of security in a large-scale network without increasing memory, communication and energy overheads.

List of Abbreviations and Symbols Used

A	$(\lambda + 1) \times N$ Matrix
CP_k	Cached Private Key on Node with ID k
C_n	Number of Member Nodes in a Cluster
D	$(\lambda + 1) \times (\lambda + 1)$ Symmetric Matrix
$E_k(\cdot)$	An Encryption Algorithm with Key k
G	$(\lambda + 1) \times N$ *Vandermonde* Matrix
G_k	Group Session Key
H	$(\lambda + 1) \times N$ Matrix
KM_λ	Vector of $\lambda + 1$ Key Materials
K_{ij}	Common Shared Key between Node i and Node j
R	Vector of Compromised Node IDs
S	Master Secret Key used for PBC Operations
S_x	Private Key used in PBC Operation for Node with ID x
X	Bit Vector of Size N
Φ	A *hashing-and-mapping* Function
$\hat{e}$	Bilinear Pairing Function
λ	Security Tolerance Parameter
$\mathbb{F}_2$	Binary Field
$\mathbb{G}$	Cyclic Groups of Integer Numbers
$\oplus$	XOR Operation
$c_i(\cdot)$	i^{th} Column Vector of a Matrix
d_s	Ratio of the Compromised Nodes to Total Nodes in a Cluster
q	Number of bits in a Large Prime Number
$r_i(\cdot)$	i^{th} Row Vector of a Matrix
AES	Advanced Encryption Standard

BDHP	Bilinear Diffie-Hellman Problem
BS	Base Station
BSD	Berkeley Software Distribution
CA	Certification Authority
CH	Cluster Head
CPU	Central Processing Unit
DoS	Denial-of-Service
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
GF	Gaussian Field
Hash	Cryptographic Hash Function
IBC	Identity-Based Cryptography
IBE	Identity-Based Encryption
ID	Identity or Identifier
IP	Internet Protocol
KM	Key Manager
LED	Light Emitting Diode
M	Number of Cluster or Cluster Head Nodes in the Network
m	Available Memory (in bits) on a Mote

MAC	Message Authentication Code
MANET	Mobile Ad Hoc Network
MDS	Maximum Distance Separable
MiM	Man-in-the-Middle
MN	Member Node
N	Number of Nodes in the Network
NFC	Near Field Communication
PBC	Pairing-Based Cryptography
PKC	Public Key Cryptography
PKG	Private Key Generator
PKI	Public Key Infrastructure
RFID	Radio Frequency IDentification
RISC	Reduced Instruction Set Code
RSA	Rivest, Shamir, and Adleman
TDMA	Time Division Multiple Access
WiFi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access
WSN	Wireless Sensor Network

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor *Dr. Srinivas Sampalli*, who made me believe in myself and guided me throughout my entire Ph.D. study. Without his continuous support, patience, motivation, enthusiasm, and immense knowledge, this thesis would not have been possible.

Besides my supervisor, I would like to thank the rest of my thesis committee members: Dr.Nur Zincir-Heywood, Dr.Vlado Keselj and Dr.Keith Johnson, for willing to be a part of the committee and taking time to read my thesis and providing insightful comments throughout the course of my research. Additionally, I thank Dr.Sagar Naik for becoming the external examiner of the thesis examining committee and providing many invaluable suggestions.

My sincere thanks goes to Raghav Sampangi, for his invaluable support and suggestions to complete this thesis. I also like to thank my fellow lab mates in MYtechlab, for stimulating discussions and for all the fun we have had in the past years.

Last but not least, I would like to thank my family: my wife Marzina Islam, and my lovely daughter Maysun Saiha Rahman, without their moral support it would have been impossible to finish this work. I am grateful to my parents: Dr. Hamidur Rahman and late Mrs. Jinnatun Nessa, for giving birth to me in the first place and supporting me spiritually throughout my life.

Above all, I thank God for giving me the strength and resources to complete my thesis.

Chapter 1

Introduction

1.1 Overview of Wireless Sensor Networks

Wireless sensor networks (WSNs) are essentially wireless ad hoc networks composed of many tiny battery-operated sensors. These sensors are deployed in harsh environments to collect different types of data, such as temperature, pressure, humidity, soil makeup, vehicular movement, noise levels, lighting conditions, the presence or absence of certain kinds of objects or substances, mechanical stress levels on attached objects, and other properties [1, 2, 3]. The collected data is then sent back to one or more special nodes called the base station (BS) either directly or via other sensor nodes [4]. Base stations are usually connected to the Internet or a WiMAX network and send this data to a remote location for further processing (Figure 1.1).

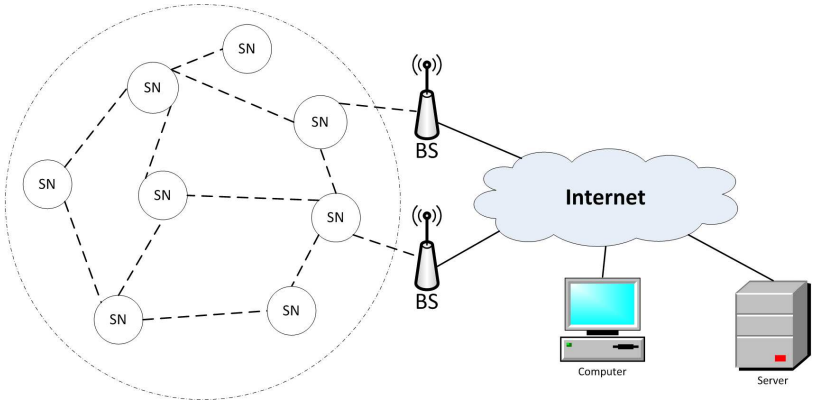

Figure 1.1: Wireless sensor network. (Legend: BS = Base Station; SN = Sensor Node)

WSNs are going to be widely used in the near future due to the breadth of their applications, specifically where it is difficult for humans to perform continuous monitoring. There are many applications of WSNs [5, 6, 7, 8] ranging from simple habitat monitoring in a forest to highly secure military applications. According to the areas of deployment, applications of WSNs can be categorized as follows [5, 7]:

- *Military applications:* Early developments in wireless sensor networks were motivated by military applications, which have the highest security requirements among the various application of WSNs [8]. Military sensing networks are designed to detect and gain as much information as possible about enemy movements, explosions, and other phenomena. Typically, wireless sensor nodes are integrated with military command, control, communications, computing, intelligence, surveillance, reconnaissance and targeting systems. Examples of military wireless sensor network applications include battlefield surveillance, guidance systems for intelligent missiles, detection of attacks by weapons of mass destruction such as nuclear, biological, or chemical weapons and other monitoring applications [9, 10, 11].

- *Environmental monitoring:* Environmental monitoring can vary from indoor to outdoor monitoring. In a large building, sensor nodes can be set up to monitor light, temperature, status of frames (windows, doors), air streams and indoor air pollution [12]. Additionally, WSNs can be used for reducing the impact of fire and earthquake damages. A fire and smoke detector system integrated with light signals indicating exits can be built to help guide the trapped residents through the safest route and save their lives [13]. In another scenario, earthquake damage can be measured by incorporating wireless sensors inside cement blocks during construction, or their attachment to structural units. Inspecting a building after an earthquake using such a system will not only facilitate evaluation of cracks and damages, but will also provide real data for modeling and prediction of structural damages to the building in future events [14].

  Outdoor monitoring is another vast area of application for WSNs, especially for habitat monitoring. One of the most representative examples of such application is Great Duck Island (GDI) [15]. Multiple types of sensors including temperature, barometric pressure, humidity, passive infrared and photo-resistors were used to monitor the natural environment of birds and impact of climate change on their behavior. Another application in the same category focuses on endangered species. A WSN was set up in Volcano National Park to discover why some species can only live in a specific region [16]. The outcome of the study would be useful when trying to prevent endangered species from extinction.

Other applications pertaining to outdoor monitoring includes environmental observations and forecasting weather phenomena.

In a nutshell, WSNs are ideal for remote monitoring, especially monitoring and event detection in geographically large regions or inhospitable areas.

- *Agriculture applications:* WSNs have a wide variety of applications in agriculture. The applications in this category are mainly focused on enhancing the efficiency and growth of cultivations. For example, WSNs are used in a vineyard to monitor temperature of grapes and to perform survey of the microclimate [17]. It turns out that the network was not only helpful for the farmers, but every participant in the wine making process, from the time of growing the grapes to wine production and marketing, benefited from it. A similar system is also used for controlling the water usage in an efficient and economic way by monitoring moisture in soil, air humidity and weather forecasting [18]. Other goals of this system include frost detection and warning as well as pesticide application and disease detection. In general, crop management, lowering costs and increasing quality are in the scope of applying sensor network technology to agriculture.

- *Industrial applications:* Industries can take most benefit out of WSNs. The industrial WSN global survey 2012 [19], stated that 70% of the industrial users are planning WSNs or additional deployments within the next 18 months. Some of the important sectors of industry where WSNs can benefit include factory automation, process control, real-time monitoring of the health of the machinery, detection of liquid/gas leakage, remote monitoring of contaminated areas, and real-time inventory management, etc. In one interesting application, British Petroleum (BP) used WSNs to improve safety and product quality by monitoring warehouses and storage management of barrels [20]. The concept is that motes attached to barrels will be capable of locating nearby objects (other barrels), detecting their content and alerting in case of incompatibility (danger of a chemical reaction), aging effects of the enclosure, etc. Similar to BP, other oil companies installed sensors to measure temperature in their pipeline systems [21]. In another application, BP used a system where each customer's oil

tank is monitored so that the supply department of the company has a knowledge of the quantity of oil remaining [20]. Besides oil industries, electricity energy systems can also benefit from deploying sensor networks. It benefits both consumers and producers of electricity by monitoring and reducing excess electricity usage. Such a scheme is being investigated in CITRIS [22], where smart energy distribution and consumption is deployed.

- *Health-care applications:* WSNs can benefit health science and health care systems. Many medical systems are equipped with a large number of tiny, non-invasive sensors, located on or close to the patient's body for health monitoring. Such systems are being designed to measure diverse physiological parameters such as blood pressure, electrocardiogram, blood oxygen level, activity recognition, etc., and are available in many different forms, including wrist wearable, ambulatory devices and as part of biomedical smart clothes [6]. The term body sensor network (BSN) [23] has been coined to represent these types of applications. One such notable use of WSNs was carried out by Intel in Portland and in Las Vegas (Proactive Health Research) [24] to help monitor and control cognitive disorders (which could perhaps lead to Alzheimer's) at their early stages. The nodes can be used to record recent actions (taking medication, last visitor, etc.), remind senior citizens to take the intended dosage of medication, or monitor for any potential health issues. Similarly human vision restoration by retina prosthesis, early detection of clinical deterioration through real-time patient monitoring in hospitals, and large-scale field studies of human behavior and chronic diseases are few among the plethora of WSNs applications in the area of health care [5].

- *Emergency services:* WSNs can help manage emergency situations during disasters. WSNs have been used to enhance the capability of the first responder to provide emergency care through automatic electronic triage [25, 26]. Triage protocols for emergency medical services [27] already exist for monitoring mass-casualty disasters. The effectiveness of the services can quickly degrade with increasing number of victims. Moreover, there is a need to improve the assessment of the first responders' health status during such mass-casualty disasters.

The WSNs can be used to automatically report the triage levels of numerous victims and continuously track the health status of first responders at the disaster scene more effectively.

1.2 Security Challenges in WSNs

Many applications in WSNs require exchange of sensitive information. Therefore, they need to rely on secure and reliable data transfer from the sensor nodes back to the processing center. Besides, the communication in WSNs is done via the wireless medium, which is inherently insecure and invariably incurs various types of security threats. Many attacks such as denial-of-service, replay, fabrication, sybil, hello flood, wormhole, etc., which are common in other wireless networks (i.e., mobile ad hoc network (MANET), wireless fidelity (WiFi), etc.) are also applicable to WSNs [6, 28]. However, many well-known security mechanisms devised for other wireless networks cannot be applied directly to the resource-limited WSNs because of the architectural disparity of the networks. Furthermore, a majority of sensor networks are deployed in hostile environments with presence of intelligent adversaries. Hence, security is critical for the practical deployment of WSNs [29]. In the next subsections, we describe the major security challenges in WSNs.

1.2.1 Limited Resources

Security comes with an overhead. WSNs consist of tiny sensors which suffer from the lack of processing, memory and battery power [4]. Applying any encryption scheme requires transmission of extra bits, with consequent processing, memory and battery power consumption which are very important resources for the sensor's longevity. Applying encryption can also increase delay and packet loss in wireless sensor networks [30].

1.2.2 Wireless Broadcast

The mode of transmission in WSNs is wireless. Wireless networks are usually more vulnerable to various security threats, as unguided transmission channels are more susceptible to security attacks than guided channels [31, 29]. Moreover, because of

harsh environments where WSNs are typically set up, error and loss of information is very common. Therefore, providing security becomes more challenging.

1.2.3 Lack of Physical Protection

Unlike traditional wireless networks, sensors are expected to be deployed arbitrarily in an enemy territory (in military reconnaissance scenarios) or over dangerous or hazardous areas. In this setup, an adversary can easily capture the physical mote (which is essentially a node in the sensor network. In this thesis, we use mote and node interchangeably.) and read sensitive information such as cryptographic keys stored in the internal memory [29, 32, 33]. In addition, the adversary can take control of and manage the mote remotely, which makes it further difficult to detect physical tampering. Even if the base station resides in a friendly or safe area, the sensor nodes need to be protected from being compromised. Therefore, protocols should have built-in mechanisms to limit the damage in the event of node capture and other physical attacks.

1.3 Key Management in WSNs

Before a WSN can exchange data securely, encryption keys must be established among sensor nodes. One critical issue in the design of secure WSNs is the key management − how the keys are generated, disseminated, revoked, renewed, or assigned on a new sensor for ensuring robust security for the network [2, 34]. Like security, key management extends across multiple layers. A common link layer protocol used in WSNs is IEEE 802.15.4 and a widely used application layer security suite for WSNs is TinySec [35]. Although these protocols require key usage for secure data transmission, they do not provide any mechanism for exchanging keys securely. Besides application and link layers, network layer may also need to exchange keys securely. This leaves open a key management problem, which is the focus of much recent research. Many security-critical applications [8, 9, 10, 11] depend on key management processes to operate but also demand a high-level of fault tolerance when a node is compromised.

1.4 Motivation and Objectives

Securing and authenticating the communication between the nodes in a WSN is very challenging, especially because the nodes are typically deployed unattended, often in conditions unfavorable to human monitoring and lacks a mechanism for secure storage for cryptographic keys, which make them vulnerable to many attacks [36, 37, 29]. Moreover, these tiny sensor nodes are limited in power, computation, memory and bandwidth, which make it harder to implement other existing security infrastructure such as public key cryptography. Therefore, in WSNs, it is necessary for the communicating parties to share encryption keys before a secure communication can take place. This necessitates the use of robust key management protocols that handle the task of distributing, establishing and managing keys between sensor nodes [2]. Many researchers have focused on providing viable solutions to the key management problem in WSNs based on symmetric *key pre-distribution* [38, 39, 40, 41, 42], since symmetric cryptosystems are lighter and faster as compared to asymmetric cryptosystems. However, managing the keys and agreeing upon a shared key requires additional overheads. Moreover, the *key pre-distribution* approach usually cannot guarantee entire connectivity of the network even with high deployment density. Besides, symmetric key cryptosystem fail to provide authentication for the communicating parties.

In Public Key Cryptography, (PKC) communicating parties only have one pair of keys namely, the private and public key. This scheme is scalable and provides authentication service easily. This convenience, though, comes at a price: a method for authenticating the public key must be provided. Traditionally, this has been done by introducing a trusted Certification Authority (CA) and verifying digitally signed certificate by the CA, which is a computationally expensive operation [43]. Since WSNs are battery powered and have limited computational power, many researchers have argued about the suitability of traditional PKC in WSNs [44, 45, 46]. However, recent advances in Pairing-Based Cryptography (PBC) have made it possible to use it in the field of resource-constrained WSNs. PBC is an emerging technology that makes well-known cryptographic protocols more efficient by enabling Identity-Based Encryption (IBE) which in turn facilitates a complete public key based scheme called Identity-Based Cryptography (IBC). The main advantage of an IBC-based system over a PKC is that in traditional PKC there is no correlation between an individual's

ID and their public key. Hence, a trusted third party is needed to establish this correlation. However, in the case of IBE, known information that uniquely identifies the user (such as email address, IP address, etc.) can be used to derive its public key. As a result, keys are self-authenticated and certificates by a trusted third party is unnecessary. Here, the only constraint is that the ID of the user has to be unique and only a trusted authority should be able to generate the ID. In other words, it requires a Private Key Generator (PKG), a trusted entity responsible for generating and escrowing the user's private key and PKG has to be unconditionally trusted by all the entities. Due to this constraint, IBE may not be suitable for many well-known scenarios on the Internet [47]. However, in the case of WSNs, this is not a problem because when the WSNs will be deployed, a trusted authority, namely, the BS of WSN will assign a unique ID to each node and load the required software before the deployment of the network. In this case, we can assume the BS will take the responsibility of assigning unique ID to the nodes.

Using IBE, we can design a non-interactive secure key distribution scheme for WSNs, where each node is issued a unique ID and a unique secret, not shared with any other entity [47, 48]. A pair of nodes can derive a common secret by knowing just each other's ID and not transmitting any messages. Then, they can derive a cryptographic key using their common secret and establish a secure communication with each other. Although the PBC-based schemes are very much viable for WSNs, the implementation is quite difficult and computationally expensive. Leonardo et al. [47] recently pointed out that even with the advanced sensor network hardware, it takes almost two seconds to perform the PBC based operations. Therefore, at least until now, it has not been possible to solely use PBC-based schemes for key management in large WSNs. This is the research gap addressed in this thesis towards the advancement of the state-of-the-art in key management protocols.

1.5 Contributions

The purpose of this research is to provide a light-weight key management infrastructure for resource constrained networks, such as WSNs, that supports scalability without increasing memory overhead. Motivated from the fact that dynamic pair-wise symmetric key is light-weight and pairing-based schemes require less keys, we have

combined these two schemes to propose a new non-interactive hybrid key management protocol for resource constrained networks. The main contributions are given below:

1. We propose an efficient key agreement scheme based on symmetric key encryption. Blom [40] has presented a key distribution method that allows any pair of nodes in a network to be able to compute a pair-wise secret key as long as no more than λ nodes are compromised, where λ is the security parameter proposed in [40]. However, the original Blom's scheme was not proposed for wireless sensor networks and require multiple interactions. We have optimized Blom's scheme to propose a non-interactive efficient key agreement protocol that respects the constraints of WSNs.

2. We have extended the above contribution by providing secure mechanisms for i) new node addition process ii) nodes (compromised) eviction process and iii) key refresh process, thereby making it a complete key management protocol for WSNs.

3. The contribution discussed in (1) suffers from a scalability problem because security parameter λ is directly related to memory of each node and network size. To address this problem, we combine it with pairing based cryptography and proposed a scalable hierarchical key agreement solution that does not increase memory overhead.

4. Finally, we have extended the contribution (2) to provide support with (i) new node addition, (ii) node revocation, and (iii) key refresh.

1.6 Outline

The remainder of the thesis is organized as follows. A literature review of related key management schemes of WSNs is discussed in Chapter 2. We provide a discussion on required background knowledge in Chapter 3. Details of our proposed key management scheme are presented in Chapter 4. Results and performance analysis are discussed in Chapter 5. A detail security analysis of the proposed scheme is presented in Chapter 6. Finally, concluding remarks are presented in Chapter 7.

Chapter 2

Literature Survey

In this chapter, we start with an overview of the key management in WSNs. We then discuss the services provided by the key management protocol followed by a discussion of some threats on the key management protocol in WSNs. After this, we provide a survey of different key management techniques in WSNs. We conclude the chapter by highlighting gaps in the existing literature and a discussion on how to bridge the gaps.

2.1 Key Management in WSNs

A key management protocol is one of the first requirements for setting up a secure communication in WSNs, because most cryptographic operations (encryption, authentication, etc.) depend on keys. Key management is the set of techniques and procedures supporting the key establishment and managing keys between authorized party. Key management includes the following procedure and techniques [34]:

- initialization of the users;
- generation, distribution and installation of the keying materials (secret private information used for generating the cryptographic key);
- control the use of keying materials;
- update, revocation, and destruction of keying material; and
- storage, backup/recovery, and archival of keying material.

Keys may be *symmetric* or *asymmetric*. In symmetric key algorithms, the same key is used for both message encryption and decryption (Figure 2.2). It is very important in a such system that the keys must be chosen, distributed and stored securely. In contrast, asymmetric key systems, use two distinct but mathematically related

keys, namely, *public* and *private* keys, one for encrypting and other for decrypting a message or vice-versa [49], (Figure 2.1). Asymmetric key algorithms rely on a Public Key Infrastructure (PKI), a type of key management system that uses digital certificates to provide validity of the public keys. Public keys need not be stored securely and may be used for encryption and user authentication. In general, the key idea of symmetric key algorithms is based on substitution and permutation of data applied many times, which makes such algorithms faster and easier to compute. On the contrary, asymmetric key algorithms use expensive mathematical operations with large prime factors to encrypt and decrypt messages and therefore are computationally expensive. Nevertheless, in both cases managing keys is crucial to the security of the system.

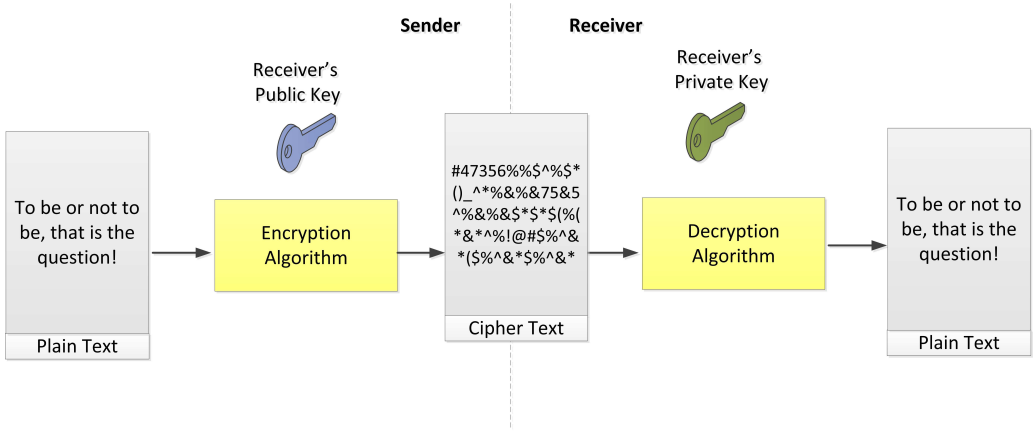

Figure 2.1: Asymmetric key algorithm.

2.2 Objective of Key Management System

The goal of a good cryptographic design is how it can reduce more complex problems by using trusted secured software or hardware to manage a small number of cryptographic keys. The objective of key management is to maintain keys and/or keying material in a manner which counters relevant threats, such as [34]:

1. Compromise of confidentiality of secret keys.
2. Compromise of authenticity of secret or public keys.
3. Unauthorized use of secret or public keys.

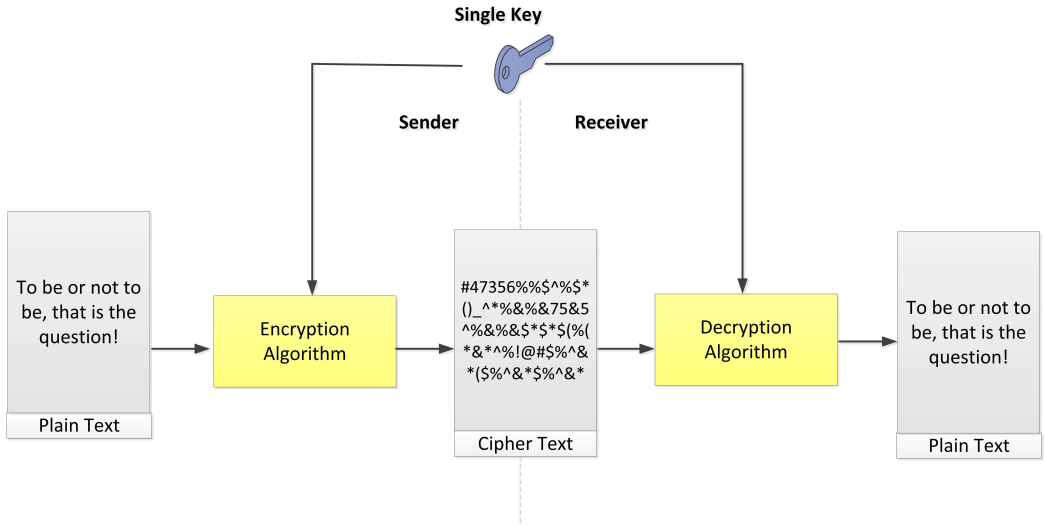

Figure 2.2: Symmetric key algorithm.

2.3 Key Management Services

In this section, we provide basic services provided by a key management protocol.

2.3.1 Key Setup

Before any secure communication can take place, global parameters for cryptographic operations, key length, etc., need to be setup. The key setup phase involves many important steps including user registration, key generation, security parameter generation and key backup. This is an important phase in key management as all communicating parties need to agree on common security parameters before they can take part in secure communication.

2.3.2 Key Exchange

Key exchange is the most important service of a key management system. In this phase, depending on the type of keys, for instance in case of symmetric key system this may require exchanging identical information. In the case of asymmetric key systems, it may only require the exchange of public keys. Although public keys can be openly exchanged, a secure channel is required for exchanging symmetric key because if an interceptor gets the symmetric key, it would enable him/her to decrypt the cipher text.

2.3.3 Key Refresh

In any system, when keys are used for a longer period of time, an adversary may gain knowledge of the keys. To prevent or make such a situation harder for the adversary, the key management system should support refresh of keys, where all the existing keys of the users should be refreshed to a new and unrelated keys. Coupled with this, the key management service should provide efficient and secure mechanisms for supporting network changes, such as adding new users or deleting existing users to and from the network.

2.3.4 Key Revocation

Key revocation is an important part of key management because it allows eviction of compromised nodes from the network. Sometimes it may not be possible to completely prevent keys being compromised. Under those circumstances, the key management system should provide mechanisms by which compromised keys of identified nodes could be revoked or made unusable in the system.

2.3.5 Challenges of Key Management in WSNs

Unlike other contemporary wireless networks, WSNs possess different types of constraints, which makes the deployment of key management exceedingly challenging. Among other challenges, key management in WSNs has the following specific challenges:

- *Lack of physical security*: An adversary can easily capture a node and extract sensitive information including security key stored in the node's memory.
- *Wireless communication*: Broadcast nature of wireless communication makes it harder to prevent unauthorized reception of the message.
- *Scalability*: Nodes in WSNs can easily become dense. With the increase of network size, the complexity of managing and supporting different protocols, and keys also increases.
- *Lack of infrastructure*: The topology of the WSNs can change dynamically, thus, centralized control is difficult. The key management protocol in WSNs

has to consider the fact and adapt with the dynamic change of topology.

- *Resource constraint*: Constraints in resources such as memory, power, and computation in sensor nodes put additional challenges on the key management protocol in WSNs. Hence, resource conscious key management techniques become a necessity in WSNs.

2.3.6 Requirements of a Key Management Protocol in WSNs

The requirements for key management can be divided into *security* and *operational* requirements [50]. The security requirement covers overall security requirements of WSNs. Table 2.1 depicts details of security requirements in WSNs key management [50].

Table 2.1: Security requirements

Services	**Functions**
1) Robustness	When some nodes are compromised, the entire network should not also become compromised. The quantitative value with which this requirement should be satisfied depends on the application.
2) Self-organization	Nodes should be independent and flexible enough to be self-organizing (autonomous) and self-healing (failure tolerant).
3) Confidentiality	Nodes should not reveal any data to any unintended recipients.
4) Authentication	Data used in decision-making processes must originate from the correct source.
5) Integrity	Data should not be changed between transmissions due to the environment or malicious activities.
6) Data Freshness	Old data should not be used as new (i.e., prevent replay attacks).
7) Availability	The network should not fail frequently.
8) Time Synchronization	Collaborative node applications need time synchronization. Time synchronization protocols should not be manipulated to produce inaccurate time.
9) Secure Localization	Nodes should be able to accurately and securely acquire location information.

Apart from the security requirements, performance requirements act as constraints

in the design and realization of key management. Table 2.2 depicts the performance requirements in WSNs key management [50].

Table 2.2: Operational requirements

Requirement	Description
1) Accessibility	Intermediate nodes should be able to perform data aggregation by combining data from different nodes. Neighboring nodes should also be able to passively monitor event signals to prevent large amounts of redundant event signaling information.
2) Flexibility	Node should be replaceable when compromised. On-the-fly addition of nodes should also be supported.
3) Scalability	Even if with the key management in place the WSNs should be scalable in number of nodes it supports.

2.3.7 Attacks on Key Management Protocols in WSNs

Node Capture Attack

Node capture attacks result from the combination of *passive*, *active*, and *physical* attacks by an intelligent adversary. In order to initialize or set up an attack, the adversary will collect information about the WSNs by eavesdropping on message exchanges, either local to a single adversarial device or throughout the network with the aid of a number of adversarial devices deployed throughout the network [37]. Even if message payloads are encrypted, the adversary can extract information about the network operation and state, effectively learning about the network structure and function. In addition to passive learning, the adversary can actively participate in network protocols, probing the network for information and maliciously injecting information into the network. Once a sufficient amount of passive and active learning has taken place, the adversary can physically capture nodes. The gathered information can be used to help the adversary make an informed decision of which sensor nodes to capture in order to optimize the performance of the attack with respect to a specific attack goal.

Wormhole Attacks

In wormhole attacks, a malicious node tries to cheat its neighbours that it is very near the base station. Thus, the malicious node can collect and drop all data from the neighbours and let the real base station not get any information from that area. Typically, the wormhole attacks require two distant malicious nodes, which have an invisible link underlying sensor network. The adversary deploys one malicious node close to the base station and the other close to the interested area. The adversary could convince the nodes of the interested area that the malicious node is near the base station by using the invisible link [50, 31].

Replay Attack

In a replay attack, a malicious node captures some valid protocol messages that have been exchanged between legitimate nodes. At a later point the same messages are replayed. The main intention of the attack is to get illegal access or to break the synchronization of the protocol [31].

Energy Consumption Attack

Due to the limited energy of sensor nodes, security protocols for WSNs must be energy efficient. However, the adversary can deploy some malicious nodes which repeatedly broadcast nonsense messages to consume the energy of normal nodes [32].

Traffic Analysis

Even when the messages transferred are encrypted, it still leaves a high possibility for the analysis of the communication patterns. Sensor activities can potentially reveal enough information to enable an adversary to cause malicious harm to the sensor network [29].

2.4 Symmetric Key Management in WSNs

Symmetric key based schemes are most popular among researchers in WSNs, largely due to their reduced computation times. Therefore, many researchers have focused on providing viable solutions to key management problem in WSNs based on symmetric

key based schemes [38, 45, 39, 40, 41, 42]. Key management schemes in wireless sensor network can be categorized as follows: (i) Single shared key schemes [45], (ii) Pair-wise key pre-distribution schemes [39], (iii) Trusted third-party based schemes [46], (iv) Probabilistic key pre-distribution schemes [38, 51, 52, 38], (v) Matrix-based key pre-distribution schemes [40, 53], (vi) Polynomial-based key pre-distribution schemes [41], (vii) Location-aware key management schemes [54, 39].

2.4.1 Single Shared Key Schemes

Lai et al. [45] proposed an easy and efficient key management solution for WSNs. In their scheme, a master key is pre-distributed and stored in each sensor in the network. A pairwise key can be established by using this master key and a random number exchanged between each sensor. This scheme is infinity scalable and very attractive since it only requires one key to be stored in each node. In addition, new node addition is very simple. However, the drawback is obvious. When the master key of a node is compromised that would compromise all the pair-wise keys of the network, and as a result the whole network would become compromised. Therefore, the scheme lacks resilience against node capture. Moreover, it does not provide any authentication since all nodes in the network uses the same master key.

2.4.2 Pair-wise Key Pre-distribution Schemes

Pair-wise key pre-distribution schemes use different secret keys for each connection with different nodes. In other words, if there are N nodes in the network, this scheme requires each node to store $(N - 1)$ distinct keys in the memory. Chen and Perrig [39] proposed such a system in which a distinct pairwise key between each pair of sensors is pre-distributed and directly stored in each node before sensor deployment. Although this solution provides the most resilience against node capture, it fails to provide scalability. This means that when the network size becomes large, this scheme becomes impractical for memory constrained sensor nodes. In addition, it is difficult to add new sensors because every sensor has to store new key when a new sensor joins the network.

2.4.3 Trusted Third-party Based Schemes

In these schemes, a mediator node acts as a trusted node for two communicating nodes. Chan and Perrig [46] propose peer intermediaries for key establishment in sensor network called "PIKE", where the key establishment between two sensor nodes is based on the common trust of a third node. For any two nodes of A and B, there is a node C that share a key with nodes A and B. The main drawback of this scheme is that sometimes it may be difficult to find such mediator node in the wireless sensor network.

2.4.4 Probabilistic Key Pre-distribution Schemes

The above schemes have many limitations. One such limitation is in large or dynamic networks, each node needs to store too many keys in its memory even though not all keys are needed for communication. In addition, to accommodate node addition and deletion to and from network or to refresh all the keys, a large number of expensive operations have to be performed and a large number of messages need to be exchanged. To solve these problems, Eschenauer and Gligor [38] have proposed a probabilistic key distribution scheme. In this scheme, initially, every node randomly selects a subset of keys from a large key pool and store the keys and their identities in memory; this step is referred as the key pre-distribution phase. The size of the random subset can be pre-determined using random graphs [51, 52, 38]. The technique aims to provide a connected graph with desired probability. If two neighbouring nodes want to communicate securely, they first exchange and compare the list of identities in their key-chains. If they happen to find a match, then they can establish a direct secure link between these two nodes — this is called *shared-key discovery* phase. In case of a mismatch, the scheme uses a *path-key establishment* phase, where it selects some intermediate nodes with a common key between the two sensor nodes to establish a common session key. That session key is then used as a path key to the selected sensor-node pairs.

This approach is better in terms of resilience to node capture since in case a malicious attacker captures a node it only reveals a subset of keys. However, this scheme still requires a large number of keys to be stored in the sensor node and a large number of messages need to be exchanged between the nodes to get shared key.

Moreover, this approach does not define any process for revoking and refreshing keys.

A simple improvement of this scheme is proposed in the Hashed Random Key Pre-distribution [42] scheme. In this scheme the keys are hashed from the key pool a different number of times for distinct nodes. The i^{th} node receives its $(i-1)$-times hashed version of key, as well as the value i. If any two nodes, say, A and B want to discover a shared key, then they need to exchange their key ID's and the value of i. For example, if nodes A and B are loaded with $K_A = Hash^{i^a}(K_i)$ and $K_B = Hash^{i^b}(K_i)$ respectively, where $(i_a > i_b)$. Then, B can easily compute $K_A = Hash^{i^a - i^b}(K_B)$. The advantage of this modification is that if a node is captured, then it reveals fewer number of keys compared to Gligor's scheme [38]. In a nutshell, it provides the same functionality of Gligor's scheme [38] with a little extra resilience against node capture with additional computation overhead.

Another improvement over Gligor's scheme [38] is proposed by Chen et al. [46]. In Chen's scheme, they shared at least q pre-distributed keys where $1 < q < n$, instead of just a single one, to establish a pair-wise key between two nodes. Therefore, an attacker now needs to compromise more nodes than in the original Gligor's approach [38] to compromise the network.

2.4.5 Matrix-based Key Pre-distribution Schemes

Blom [40] has proposed another major pair-wise key distribution solution that avoids some of the communication overhead involved in *shared-key discovery* phase in Gligor's scheme [38]. The basic idea of this scheme is based on polynomial based key generation scheme. In this scheme, at key *pre-distribution* phase, a public $(\lambda+1)\times n$ matrix M and a secret $(\lambda+1)\times(\lambda+1)$ symmetric random matrix D are generated. Each node i stores row_i of $(D \cdot M)^T$ matrix as private information, and col_i , the i^{th} column of M, which is used as public information. After deployment, in the key discovery phase, all nodes broadcast their public column instances of M, allowing any pair of nodes i and j to compute $K_{ij} = row_i \times col_j = row_j \times col_i = K_{ji}$. This works because of the symmetric property of matrix $K = (D \cdot M)^T \cdot M$, where $K_{ij} = K_{ji}$, computed from the public and private instances of all the nodes. Blom's scheme is λ secure, meaning that capturing up to λ nodes does not reveal the network key. However, the capture of more than λ nodes would compromise the entire network.

This scheme has attractive properties. Since it allows creation of pair-wise keys, node authentication and key revocation are made easy. It also provides good key connectivity while respecting the resource constraints of sensor nodes. Additionally, the resilience of the node capture can be adjusted by choosing the λ value appropriately. However, the size of λ is directly proportional to the memory size of the node. Thus, one has to take care in choosing λ to select adequate trade-off between security and efficiency.

An optimized version of Blom's scheme is proposed in [53], where the authors optimize the generation of M matrix using *Vandermonde* matrix and using node ID as seed for the *Vandermonde* matrix. As a result, nodes do not need to store the public column vector; instead they can generate the value of public column from the seed. In addition, a substantial improvement of their protocol is that it completely avoids the *key discovery* phase and any pair of nodes can generate the secret key only knowing each other IDs without requiring additional communications. They also provide a complete key management protocol including key-refresh and key revocation schemes.

Du et al. [55] combined Blom's [40] and Gligor's scheme [38] to provide a solution called Multiple-Space Key Pre-Distribution Scheme. It provides more resilience against node capture with less memory overhead. Like Blom's scheme [40] , they use a $(\lambda + 1) \times n$ public matrix M and a set containing ω secret random $n \times (\lambda + 1)$ matrices $D_i, 1 \leq i \leq \omega$, which defines a set of ω spaces $(D_i \cdot M)$. Before the network is deployed, a trusted authority generates and distributes initial key materials to every node in the network. Every sensor node j stores j^{th} column of matrix M, in addition it selects τ number of randomly chosen rows from matrix $(D_j \cdot M)^T$. When two nodes want to establish a common shared key between them then one node broadcasts its column instance together with the τ IDs of the spaces it carries. If the other node shares a common space among these τ spaces, they can establish a pair-wise key following Blom's Scheme [40]; otherwise, a common key can be generated using the Gligor's *path-key establishment* protocol [38]. Although this scheme provides very good resilience, it requires large storage and communication overhead. Moreover, due to the complex steps involved in the scheme it is difficult to implement.

Another extension of the Blom's scheme [40] is proposed by Chien et al. [54]. They

use a master secret key K together with information loaded by Blom's scheme [40]. After deployment, a node j selects a random number R_j which is $XORed$ with K and broadcast with its col_j instances. Other nodes can re-constitute R_j by performing XOR with the master key K already stored in their memory. After computing the common key $K_{ij} = row_i \times col_j = row_j \times col_i = K_{ji}$, following Blom's scheme [40], it creates a reinforced shared key $K^r_{ij} = Hash(K_{ij} \| R_i \| R_j)$ with all its neighbours. Then, it deletes the instances of col_j, row_i and K from its memory, preventing an attacker from using such information in the future. This scheme provides good resilience against node capture. If an attacker compromises more than λ nodes after the keys are erased, it will not gain any information about the keys or reinforced keys. However, the deletion of keys also introduces some drawbacks, since operations such as new node addition and key refresh will become difficult as the previous keys or key materials would have been deleted by the nodes.

2.4.6 Polynomial-based Key Pre-distribution Schemes

Blundo et al. [41] have analyzed the Blom's protocol [40] and proposed the polynomial-based key pre-distribution scheme. In this scheme, a λ-degree bi-variate symmetric polynomial

$$f(x, y) = \sum_{i,j=0}^{\lambda} a_{ij} x^i y^j$$

over $GF(q)$ is used, where q is a large prime number over Gaussian Field. In the key *pre-distribution* phase for each sensor node i, the Key Generator (KG), generates and stores a partial polynomial $f(i, y)$ evaluated at node index i. In this manner each node stores $(\lambda + 1) \cdot \log_2(q)$ polynomial shares in their memory. In the *key discovery* phase, node i can establish a common key with node j by evaluating *f(i,y)* at node j and vice versa. Since they use symmetric functions, they both generate the same key.

Similar to Blom's protocol [40], this scheme has the same λ-secure property and good key connectivity, as well as it can identify and authenticate individual nodes. However, since each node needed to store polynomial shares, it has higher memory overhead, this can increase dramatically when the network size increases to facilitate better security.

Liu and Ning [56] extended Blundo's scheme [41] by combining the Gligor's scheme [38] with it. They proposed the Polynomial Pool-based Key Pre-distribution scheme which uses a set containing ω randomly generated λ-degree bi-variate symmetric polynomial

$$f(x,y) = \sum_{i,j=0}^{\lambda} a_{ij} x^i y^j \ \ over\ GF(q)$$

Broadly speaking, this scheme is similar to Du's scheme [55] except that instead of using matrices they use a set of polynomials. On the one hand, when the set of polynomial contains single polynomial their scheme degenerates to Blundo's scheme [41]. On the other hand, when the degree of the polynomial is one, this scheme becomes Gligor's scheme [38]. This scheme shares some features from Blundo's scheme [41], such as λ-secure property, node-to-node authentication capability, and dynamic node addition. Although this scheme provides good secure connectivity, it increases memory overheads and communication overheads as the network size increases.

2.4.7 Location-aware Key Management Schemes

Other researchers have tried to improve the efficiency by relaxing some parameters such as adding location information or prior deployment knowledge to the key management problems. One notable example in this category is the Du et al. [39], Group-based Deployment Scheme, where they assume that nodes will be deployed as a group over a rectangular area. In their scheme, they use Gligor's approach [38] with location information and divided the key pool such that known neighbours will have higher possibility of sharing common keys, thus improving the overall performance. In another approach Liu and Ning [57] have added location information to the Random Pair-wise Scheme [39] and are able to improve the scalability and key connectivity without impacting on the amount of memory needed to store keys. The authors in [58] have combined Blundo's approach [41] with deployment knowledge, by assuming the nodes are deployed in groups and following Gaussian distribution around the point of deployment. The network is then modelled as hexagonal grid that covers all the nodes. Each hexagon receives a coordinate and associated with three cells. Finally, each cell is given a λ-degree polynomial and thus each node stores three polynomial shares and can establish keys in the same manner as in Blundo's protocol [41].

2.5 Asymmetric Key Management in WSNs

Some researchers believe that asymmetric key based schemes are computationally expensive for WSNs because of their constraints. However, motivated by recent improvement in sensor hardware and optimization of cryptographic algorithms, several research groups [59, 60, 61] have shown impressive results which indicate asymmetric key scheme may also be used in wireless sensor networks.

2.5.1 RSA-based Key Management for WSNs

A notable RSA-based public key scheme has been proposed by Watro et al. [60] called TinyPK. It allows authentication and key agreement between resource constrained sensors. This scheme uses basic RSA to generate private and public key pair for each node in the network. In this scheme, they assume that there is a certification authority (CA) available before the start of the protocol and any third party that wishes to interact with the motes also requires its own public/private key pair and must have its public key signed by the CA's private key, thus establishing its identity. To perform authentication, the external party submits its signed public key and some text signed with its private key. The protocol operation starts when the third party provides a challenge to the sensor network. This challenge consists of two parts: the first part consists of the public key of the node, signed by the CA's private key; the second is a compound object consisting of a nonce (a time-stamp) and a message checksum, signed with the third party's own private key. This information is sent as clear text and the nonce serves to detect any replay attacks, wherein a malicious party records previous valid messages and re-broadcasts the message in order to provide false identification or otherwise attack a system. The checksum is used to ensure message integrity.

When the message is received by a node, it uses the preloaded CA public key to verify the first part of the challenge and extract the third party's public key. It then uses this public key to verify the second part of the message and extract the nonce and checksum. The nonce and checksum are validated. If they pass validation, the third party has been successfully authenticated to the sensor network and is considered an authorized entity for sensor data. Then, the sensor node uses this key to encrypt the

session key plus the received nonce using the third party's public key. Following this, the message is sent back to the third party, which decrypts it using its private key, checks that the nonce is the same as the one it sent, and if so, can record the session key for future use.

2.5.2 ECC-based Key Management WSNs

Malan et al. [59] present a public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. They have implemented the ECC based public key infrastructures in WSNs mote on TinySec [35]. They argue that public-key infrastructure is viable for TinySec keys distribution. Improving ECC based public key scheme Ren et al. [61] proposed another public key scheme for WSNs. They have proposed $n - authentication$, where if a user can successfully authenticate with any subset of sensors out of a set of n sensors, the $n - authentication$ succeeds. In this scheme, each user of the WSN is equipped with a public/private key pair (PK/SK), and signs every message he/she broadcasts with the SK using a digital signature. To prove the user's ownership over his/her public key, the sink (network planner) is also equipped with a public/private key pair and serves as the certificate authority. For the purpose of message authentication, sensor nodes are preloaded with PK_{Sink} before the network deployment; and message verification contains two steps: the user certificate verification and the message signature verification. Although ECC based schemes require less storage, it is computationally expensive for resource-constrained wireless sensor motes. In addition, a trusted third party (that is CA) must be available to certify the user's public key using digital signature. Verifying the digital signature requires expensive mathematical operations. Therefore, practically it is not feasible to apply in a WSN.

2.5.3 ID-based Key Management in WSNs

The motivation of identity-based encryption is to simplify the certificate-based public key encryption system. In the certificate-based public key encryption system, a user has to verify another user's certificate before using his/her public key. As a result, each user requires a large storage and computing time to store and verify each other's public keys and the corresponding certificate. The basic idea of the identity-based encryption

scheme is that an arbitrary string can work as a public key. As a consequence, a user can use any ID, such as email, to calculate a public key, rather than extracting from the certificate issued by a certificate authority [62].

Shamir [63] first introduced identity based cryptography to simplify the management of public keys in a public key based crypto-system, in which he tried to address how a party's public key can be generated from the party's identity. Although his idea was very appealing, for a long time it was an open research problem to obtain an efficient and secure IBE scheme. Recently, Sakai [64] and Joux [65] have shown an excellent improvement and a way how IBE can be used efficiently in a variety of applications. Almost at the same time, another remarkable research was conducted by Boneh and Franklin [66] in the same area and able to come up with promising results to use the IBE scheme in different cryptographic applications. The building blocks of IBE is based on the concept of bilinear pairing — or pairing for short. The followings are necessary facts about the pairing using the notation of [67].

- $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are multiplicative groups of prime order p.
- g_1 is a generator of $\mathbb{G}_1$and g_2 is a generator of $\mathbb{G}_2$.
- ψ is an isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ with $\psi(g_2) = g_1$.
- $\hat{e}$ is a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

The map $\hat{e}$ must have the following properties:

- *Bilinear*: $\forall u \in \mathbb{G}_1$, $\forall v \in \mathbb{G}_2$ and $\forall a, b \in \mathbb{Z}$ we have $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$
- *Non-degenerate*: $\hat{e} : (g_1, g_2) \neq 1$.
- *Computable:* there is an efficient algorithm to compute $\hat{e}(u, v)$ for all $u \in \mathbb{G}_1$ and $v \in \mathbb{G}_2$.

Note that the map always exists, but the issue here is whether or not it can be efficiently computed within an acceptable time limit. For the sake of simplicity and without loss of generality we could write $g_1 = g_2$ i.e., $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$.

Using the IBE, it is possible to design a non-interactive secure key distribution scheme for WSNs, where each node is given a unique ID and a unique secret, not

shared with any other entity on the network. A few recent notable research works focusing on the use of IBE in key management of WSNs can be found in [68, 69, 70, 71].

2.6 General Discussion

Although symmetric key schemes are light-weight and widely used in WSNs, there are some notable drawbacks. Firstly, managing the keys and agreeing on a shared key requires additional overhead. Secondly, the *key pre-distribution* approach usually cannot guarantee entire connectivity of the network even with high deployment density. Finally, symmetric key cryptosystems fail to provide authentication for the communicating parties. In contrast, in asymmetric key schemes, communicating parties only have one pair of keys namely, the *private* and *public* key. This scheme is scalable and provides authentication service easily. This convenience though comes at a price: a method for authenticating the public key must be provided. This traditionally been done by introducing a trusted CA and verifying digitally signed certificate by the CA, which is a computationally expensive operation [43]. However, using an ID-based scheme, known information that uniquely identifies the user can be used to derive its public key. As a result, keys are self-authenticated and certificates by a trusted third party is thus unnecessary. Here, the only constraint is that the ID of the user has to be unique and only a trusted authority should be able to generate the ID. Although ID based schemes are promising in WSNs, a recent research [47] shows that the most efficient implementation of IBE on most state-of-the-art sensor nodes still takes about *two* seconds to perform one complete key generation, which in our opinion still not feasible solution for a large WSNs. Therefore, there remains a gap in providing key management solution in WSNs that not only light-wight, but also performs well in a large size network.

In this thesis, we propose a hybrid solution to the key management problem in WSNs where we combine the matrix-based key pre-distribution scheme with pairing-based scheme to provide efficient and scalable key management protocol. Although we combined pairing-based technique in the proposed scheme, it is used rarely, so that the overall computational overhead of pairing-based calculations are minimized. The proposed key management protocol supports multiple types of networks and provide scalability without increasing memory overhead. In addition, it provides flexible

mechanisms for key refresh, key revocation and new node addition.

Chapter 3

Background Knowledge

In this chapter, we elaborate the background techniques used in the proposed key management protocol. First, we present details of the matrix based key pre-distribution scheme [40] followed by a discussion on security analysis of the scheme. Following this, we elaborate the identity-based encryption schemes [67, 72] and provide their security analysis. Finally, we conclude with a classification of different clustering techniques in WSNs.

3.1 Blom's Scheme

Blom [40] has presented a key distribution method that allows any pair of nodes in a network to be able to find a pair-wise secret key as long as no more than λ nodes are compromised. It is a simplified version of Blundo's polynomial-based approach [41] and can be optimized for non-interactive and identity based scheme. Each party requires only an index i, $1 \leq i \leq N$, which uniquely identifies the party with which it is to form a shared key. Each user is assigned a secret vector of initial keying material (*base key*) from which it is then able to compute a pairwise secret (*derived key*) with each other user.

The scheme is efficient in terms of memory and may be engineered to provide unconditional security against coalitions of a specified maximum size (λ). The initial keying material assigned to each user (a row of A values, corresponding to λ keys) allows computation of a larger number of derived keys (a row of λ values, providing N keys), one per each other user. Storage savings result from choosing λ less than N. It involves two phases, *key pre-distribution* and *key agreement*.

3.1.1 Key Pre-distribution

In this phase a trusted third party first constructs a $(\lambda) \times N$ matrix G over a finite field $GF(q)$, where N is the size of the network and $GF(q)$ is a q-bit prime number large

enough to accommodate the key size. G is considered public and everyone may have information about G matrix. To achieve the λ-security the columns of the G matrix must be linearly independent. One such matrix could be a *Vandermonde* matrix with each term of a geometric progression [73]. For example, an $\lambda \times n$ Vandermonde matrix is shown below:

$$V = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ s & s^2 & s^3 & \dots & s^n \\ s^2 & (s^2)^2 & (s^3)^2 & \dots & (s^n)^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s^\lambda & (s^2)^\lambda & (s^3)^\lambda & \dots & (s^n)^\lambda \end{pmatrix}$$

The trusted third party then creates a random $\lambda \times \lambda$ symmetric matrix D over $GF(q)$, and computes $N \times \lambda$ matrix $A = (D \times G)^T$, where $(D \times G)^T$ is the transpose of $D \times G$. Matrix D is the secret information and should not be disclosed to anyone (although only one row of A will be distributed to each sensor node). Since D is symmetric, the key matrix $K = A \times G$ can be written as:

$$K = A \times G = (D \times G)^T \times G = G^T \times D^T \times G = G^T \times D \times G = (A \times G)^T = K^T$$

This means that K is also a symmetric matrix. Therefore, $K_{ij} = K_{ji}$, where K_{ij} is the element in K located in the i^{th} row and j^{th} column. Node i uses K_{ij} and node j uses K_{ji} as pair-wise key to communicate with each other. Figure 3.1 depicts an illustration of the pair-wise key $K_{ij} = K_{ji}$ generation process. Finally, the base station distributes, for $k = 1 \dots N$: k^{th} row and k^{th} column of matrix A and G respectively to node k as its key material.

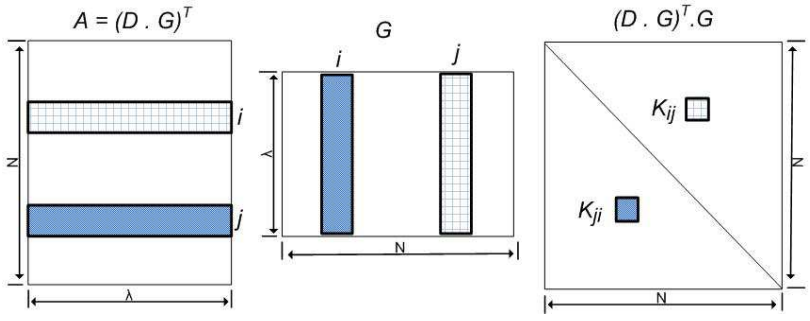

Figure 3.1: Pair-wise key genaration in Blom's scheme

3.1.2 Key Agreement

When any pair of nodes i and j want to find a pair-wise key between them, they first exchange their column of G, then they can compute K_{ij} and K_{ji}, respectively, using their private rows of A. Since G is allowed to be known by everyone, there is no harm in transmitting its columns in plain-text.

We summarize the scheme in Algorithm 1.

Algorithm 1 Blom's symmetric key pre-distribution system adopted from [34]

Input: Each of n users is given initial secret keying material and public data.

Output: Each pair of users U_i, U_j may compute an m-bit pairwise secret key K_{ij}.

1: A $\lambda \times n$ generator matrix G of an (n, λ) MDS code (see 3.1.3) over a finite field $\mathbb{F}_q$ of order q is made known to all n system users.

2: A trusted party T creates a random $\lambda \times \lambda$ symmetric matrix D over $\mathbb{F}_q$.

3: T gives to each user U_i the secret key S_i, defined as row i of the $n \times \lambda$ matrix $A = (D \cdot G)^T$. (A_i is a λ-tuple over $\mathbb{F}_q$ of $\lambda \cdot lg(q)$ bits, allowing user U_i to compute any entry in row i of $(D \times G)^T \times G$).

4: Users U_i and U_j compute the common secret $K_{ij} = K_{ji}$ of bit length $m = lg(q)$ as follows:

- Using A_i and column j of G, user U_i computes $(i, j)_{th}$ entry of the $n \times n$ symmetric matrix $K = (D \times G)^T \times G)$.
- Using A_j and column i of G, user U_j similarly computes $(j, i)_{th}$ entry (which is equal to $(i, j)_{th}$ entry since matrix K is symmetric).

3.1.3 Security Analysis of Blom's Scheme

Background on Maximum Distance Separable Codes

The motivation for Blom's scheme (Algorithm 1) arises from well-known concepts in linear error-correcting codes, which are summarized here. Let $G = [I_\lambda A]$ be a $\lambda \times n$ matrix where each row is an n-tuple over $\mathbb{F}_q$ (for q a prime or prime power). I_λ is the $\lambda \times \lambda$ identity matrix. The set of n-tuples obtained by taking all linear combinations (over $\mathbb{F}_q$) of rows of G is the *linear code* C. Each of these q^λ n-tuples is

a code-word, and $C = \{c : c = mG, m = (m_1, m_2, \ldots m_\lambda), m_i \in \mathbb{F}_q\}$. G is a generator matrix for the linear (n, λ) code C. The *distance* between two codewords c, c' is the number of components they differ in. The distance d of the code is the minimum such distance over all pairs of distinct codewords. A code of distance d can correct $e = \lfloor b(d1)/2 \rfloor$ component errors in a codeword, and for linear codes $d \leq n\lambda + 1$ (*the Singleton bound*). Codes meeting this bound with equality ($d = n - \lambda + 1$) have the largest possible distance for fixed n and λ, and are called maximum distance separable (MDS) codes [34].

λ-secure Property

The condition $d = n - \lambda + 1$ dening MDS codes can be shown equivalent to the condition that every set of λ columns of G is linearly independent. From this, two facts about MDS code follows [34]:

(i) any λ components uniquely define a codeword;

(ii) any $j \leq \lambda - 1$ components provide no information about other components.

In case of Blom's scheme, the choice of the λ is crucial because if λ or more users are compromised, then they are able to recover the secret keys of all other users. Since, λ conspirators may compute λ rows of K, or equivalently λ columns, corresponding to λ components in each row. Each row is a codeword in the MDS code generated by G, and corresponds to the key of another user, and by the above remark λ components thus define all remaining components of that row. However, if fewer than λ users conspire, they obtain no information whatsoever about the keys of any other user (by similar reasoning). Thus Blom's scheme is j-secure for $j \leq \lambda - 1$.

3.2 Identity-based Encryption

3.2.1 General Overview of IBE

IBE is another form of public-key encryption technology that allows a user to calculate a public key from an arbitrary string. One can change the arbitrary string to avoid a user having the same IBE key forever. Usually, it is useful to include some information about the user's identity or validity period of the key in the string. The

ability to calculate keys as needed gives IBE systems different properties than those of traditional public-key systems, and these properties provide significant practical advantages in some situations. Although there are probably few situations in which it is impossible to solve any problem with traditional public-key technologies that can be solved with IBE, the solutions that use IBE may be much simpler to implement and much less expensive to support than alternatives.

In a traditional public-key system, either the user or an agent working on behalf of user randomly generates public-private key pair. After it is created, the public key, along with the user's identity need to be registered with a CA. The CA is responsible to verify the authenticity of the user's information before it can digitally sign the user's public key to generate a digital certificate. A copy of the digital certificate is then sent to the owner of the private key and another copy is stored in a public certificate repository that is accessible by others who might need to get a user's key. The verification of the user's identity is carefully handled by the CA before a digital certificate is issued to the user, a process that is typically very expensive. The process of generating public-private key pairs can also be computationally expensive. Generating two 512-bit prime numbers that are suitable for use in creating a $1,024$-bit RSA private key is certainly feasible, but generating larger primes gets progressively more expensive [72]. Creating two $7,680$-bit primes that are suitable for use in creating a $15,360$-bit RSA private key is not an operation that widely used computers can easily perform, yet such keys are needed to securely transport the 256-bit AES keys that are used today [72]. Since generating keys and verifying user's identities can be expensive, digital certificates are often issued with fairly long validity periods, often between one and three years. Due to the relatively long validity period of the public keys managed by digital certificates, it is often necessary to check the key in a certificate for validity before using it.

To use the public key that is signed by a CA, a user must verify that the certificate is not expired or revoked. This can be done either by sending queries to the public repository for new certificate or by checking a list of invalid certificates or by querying an online service that returns validity status of the certificate. After any necessary validity checking is done, the user can use the public key to encrypt information and send it to the owner of the pubic key. Since only the recipient has the private key

corresponding to the public key, he can decrypt and get the information [72].

An IBE system has similarities with traditional public-key systems. However, it can be different in other ways. While in traditional public-key system, the certificate contains all the necessary parameters to use the key, to use an IBE system, typically a user needs to contact a trusted third party and get all the necessary parameters to generate the key. When he receives necessary parameters, it can generate IBE public key of any user and use it to encrypt information.

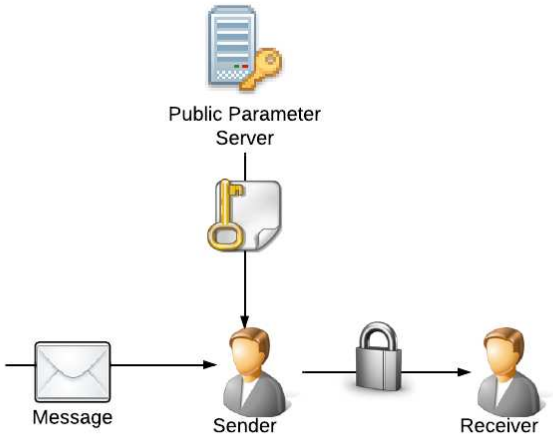

Figure 3.2: Generation of keys in an IBE system. (adapted from [72].)

When an IBE encrypted message is received by a user, the user contacts PKG and authenticates himself in some way. The PKG is a trusted third party responsible for generating IBE private key that corresponds to the IBE public key used to generate the encrypted message. The PKG typically uses some secret information called master key and user's information to generate the IBE private key. After the private key is generated, it is securely distributed to the authenticated user. The authenticated user then uses the IBE private key to decrypt the information. This is shown in Figure 3.2. The building blocks of IBE is based on the concept of bilinear pairing — or pairing for short.

3.2.2 Bilinear Pairing

In this section, we describe general overview of necessary facts about the pairing using the notation of [67]. Let r be a prime number. Let $\mathbb{G}_1$, and $\mathbb{G}_T$ be cyclic groups of order r. Let $\mathbb{G}_2$ be a group where each element has order dividing r. In particular, $\mathbb{G}_2$ is not necessarily cyclic. Again, we use multiplicative group notation. A bilinear pairing $\hat{e}$ is an efficiently computable function:

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

such that

- *Bilinear*: $\forall g_1 \in \mathbb{G}_1$, $\forall g_2 \in \mathbb{G}_2$ and $\forall a, b \in \mathbb{Z}$ we have $\hat{e}({g_1}^a, {g_2}^b) = \hat{e}(g_1, g_2)^{ab}$
- *Non-degenerate*: $(g_1, g_2) = 1_{\mathbb{G}_T}$ for all $g_2 \in \mathbb{G}_2$ if and only if $g_1 = 1_{\mathbb{G}_1}$, and similarly $(g_1, g_2) = 1_{\mathbb{G}_T}$ for all $g_1 \in \mathbb{G}_1$ if and only if $g_2 = 1_{\mathbb{G}_2}$.
- *Computable:* there is an efficient algorithm to compute $\hat{e}(g_1, g_2)$ for all $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$.

Note that the map always exists, but the issue here is whether or not it can be efficiently computed. For the sake of simplicity and without loss of generality we could write $g_1 = g_2$ i.e., $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$.

3.2.3 Boneh-Franklin IBE Scheme

Source	**Private value**	**Public value**
System Parameter	s	sP
Alice	r	rP
Bob	$s \cdot tP = sQ_{ID}$	$tP = sQ_{ID}$

Table 3.1: Summary of public and private values in Boneh-Franklin IBE. (adapted from [72].)

So, after calculating the shared secret $\hat{e}(P, P)^{rst}$, Alice hashes the shared secret into a format compatible with the plain-text. The value of $\hat{e}(P, P)^{rst}$ is an element of some $\mathbb{F}_q$, for example, while a typical message is an element of $(0, 1)^*$, so that $\hat{e}(P, P)^{rst}$ needs to be mapped into $(0, 1)^*$ so that it can be combined with the plain-text to

produce the cipher-text. So, Alice hashes the shared secret $\hat{e}(P,P)^{rst}$ to the message space and combines the resulting hash with the plain-text M to get the cipher-text $C = M \oplus Hash(\hat{e}(P,P)^{rst})$. Bob then calculates the shared secret $\hat{e}(P,P)^{rst}$, hashes it to the message space, and recovers $M = C \oplus Hash(\hat{e}(P,P)^{rst})$.

In Algorithms 2, 3, 4, 5, we provide the summary of the steps in Boneh-Franklin IBE scheme.

Algorithm 2 Boneh-Franklin IBE Setup adapted from [72]

Input: A security parameter k , an elliptic curve E, a plaintext bit length n.

Output: $BFParams = (G_1, G_T, \hat{e}, n, P, sP, H_1, H_2, H_3, H_4)$ and master secret s.

1: Select a prime p and prime power q with $p \mid \#E(\mathbb{F}_q)$ and $p^2 \nmid \#E(\mathbb{F}q)$ and such that the bit security level provided by p and q meets the required security parameter k. For best performance, p should be a Solinas prime.

2: Select a random $P \in E(\mathbb{F}_q)[p]$ and let $G_1 =< P >$.

3: Let k be the embedding degree of $E/\mathbb{F}_q$; select a pairing $\hat{e} : G_1 \times G_1 \to \mathbb{F}^*_{q^k}$.

4: Let $G_T =< \hat{e} < P, P >>$.

5: Select a random $s \in \mathbb{Z}^*_p$ and calculate sP.

6: Select appropriate cryptographic hash functions $H_1 : \{0,1\}^* \to G_1$, $H_2 : G_T \to \{0,1\}^n$, $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}^*_p$ and $H_4 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}^*_p$.

7: The master secret is the value s.

8: The public parameters are $BFParams = (G_1, G_T, \hat{e}, n, P, sP, H_1, H_2, H_3, H_4)$.

Algorithm 3 Boneh-Franklin Key Extraction adapted from [72]

Input: A string ID representing an identity and a set of public parameters $BFParams = (G_1, G_T, \hat{e}, n, P, sP, H_1, H_2, H_3, H_4)$.

Output: The private key sQ_{ID}.

1: Calculate $sQ_{ID} = sH_1(ID)$.

3.2.4 Security of IBE

Note that we can write $Q_{ID} = tP$ for some (unknown) t, so we have $\hat{e}(rQ_{ID}, sP) = \hat{e}(rtP, sP) = \hat{e}(P,P)^{rst}$. So, we can also think of the ciphertext as being $C =$

Algorithm 4 Boneh-Franklin Encryption adapted from [72]

Input: A plaintext message M of length n bits, a string ID representing the identity of the recipient of the ciphertext, a set of public parameters $BFParams = (G_1, G_T, \hat{e}, n, P, sP, H_1, H_2, H_3, H_4)$.

Output: A ciphertext $C = (C_1, C_2, C_3)$.

1: Calculate $Q_{ID} = H_1(ID)$.
2: Select a random $\sigma \in \{0,1\}^n$.
3: Calculate $r = H_3(\sigma, M)$.
4: Calculate $C_1 = rP$.
5: Calculate $C_2 = \sigma \oplus H_2(\hat{e}(rQ_{ID}, sP))$.
6: Calculate $C_3 = M \oplus H_4(\sigma)$.

Algorithm 5 Boneh-Franklin Decryption adapted from [72]

Input: A ciphertext $C = (C_1, C_2, C_3)$, a set of public parameters $BFParams = (G_1, G_T, \hat{e}, n, P, sP, H_1, H_2, H_3, H_4)$, a private key sQ_{ID}.

Output: A plaintext message M or an error condition.

1: Calculate $\sigma = C_2 \oplus H_2(\hat{e}(sQ_{ID}, C_1))$.
2: Calculate $M = C_3 \oplus H_4(\sigma)$.
3: Calculate $r = H_3(\sigma, M)$ and then calculate rP. If $C_1 \neq rP$ then raise an error condition that indicates an invalid ciphertext. Otherwise, return the plaintext M.

$(rP, M \oplus H_2(\hat{e}(P,P)^{rst})$. An adversary can obtain P and sP from the public parameters, can calculate $Q_{ID} = tP$ from the recipients identity, and observes rP in the ciphertext. If he can calculate $\hat{e}(P,P)^{rst}$ from P, rP, sP, and tP then he can recover the plaintext message M by calculating $(M \oplus H_2(\hat{e}(P,P)^{rst})) \oplus H_2(\hat{e}(P,P)^{rst} = M$, but calculating $\hat{e}(P,P)^{rst}$ in this way is exactly the BDHP. So, if the BDHP is sufficiently difficult, then it will be difficult for an adversary to recover a plaintext message from a corresponding ciphertext.

Although IBE is a very interesting technique, it has some shortcomings. IBE requires incessant availability of the PKG and PKG needed to be trusted by all the users. It also requires some sort of secure and authenticated channel between PKG and a user to send the IBE private key. In the context of the Internet these requirements are difficult to implement and therefore IBE has not been very attractive solutions in Internet. However, this is not a problem in WSNs. Usually, WSNs are deployed and managed by a single authority and a centralized control is assumed through the base station node.

3.3 Network Clustering in WSNs

Routing is critical to the operation of WSNs. Since WSNs have unique characteristics, routing protocols from other networks such as mobile ad hoc networks or cellular networks are not suitable. Firstly, WSNs are highly resource constrained in terms of power supply, processing capability and transmission bandwidth. Secondly, a global addressing scheme such as IP is difficult to design in WSNs because address updating in a large-scale or dynamic WSNs can result in heavy overhead. Thirdly, many sensor nodes usually collect data in the same geographic area. As a result there is high probability of data redundancy, which must be considered by routing protocols. Finally, most applications of WSNs require many-to-one communication scheme, i.e., from multiple sources to one particular sink, rather than multicast or peer-to-peer communications.

Based on network structure, routing in WSNs can be broadly classified into two categories: *flat routing*, and *hierarchical routing*. In a flat topology, nodes perform identical tasks and have similar functionalities in the network. All sensor nodes collaborate together to perform the sensing task and data transmission is performed

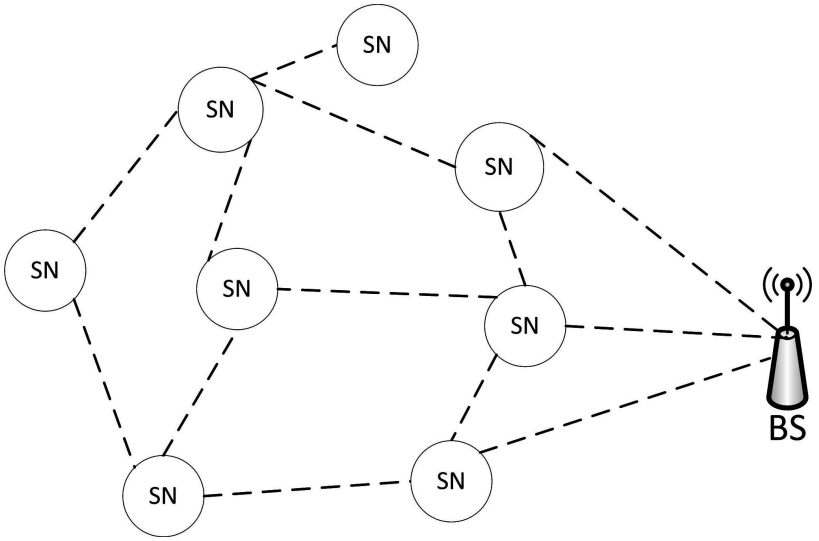

Figure 3.3: Example of flat WSNs. Legend: BS = Base Station; SN = Sensor Network

hop-by-hop basis usually using the form of flooding [74]. Figure 3.3 shows an example of WSNs with flat routing. Some of the common routing protocols in the flat routing includes [75, 76, 77]. Since in flat routing nodes use more bandwidth and energy, it is not very suitable for a large scale networks. In contrast, in a hierarchical topology, nodes perform different tasks in WSNs and typically are organized into groups or clusters according to specific requirements or metrics. Often a single cluster head (CH) is then identified within each group and made responsible for collecting and processing data from all group member nodes which are then sent to one or more base stations [78]. In general, nodes with higher energy act as CH and perform the task of data processing and information transmission, while nodes with low energy act as member nodes (MNs) and perform the task of information sensing. Figure 3.4 depicts an example of hierarchical routing in WSNs. Some typical clustering routing includes [79, 80, 81, 82, 78, 83, 84].

Hierarchical routing based on clustering is becoming more active branch of routing technology in WSNs. Clustering provides a variety of advantages, such as data aggregation, scalability, load-balancing, less energy consumption, etc. Data aggregation is a very important optimization, which may increase the lifetime of WSNs. It is any process in which information is gathered and expressed in a summary form with minimum information loss [85]. This method can eliminate redundant data transmissions thus reduce energy consumption in wireless sensor networks. The most popular

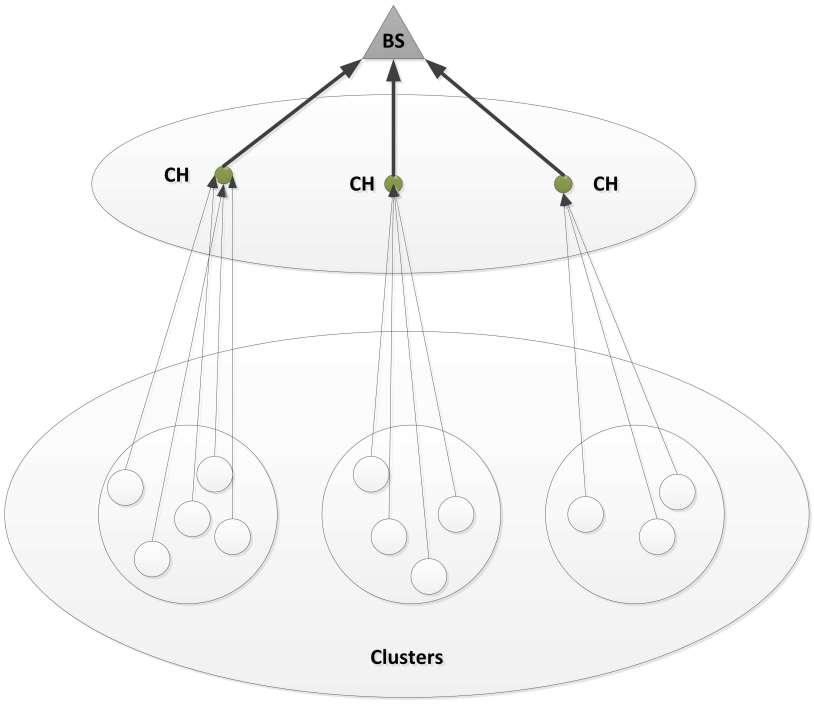

Figure 3.4: Example of hierarchical WSNs. Legend: CH = Cluster Head; BS = Base Station

data aggregation/fusion method is clustering data aggregation, in which each CH aggregates the collected data and transmits the fused data to the BS. In Table 3.2, we provide a summary of pros and cons of flat vs. hierarchical routing techniques in WSNs.

Table 3.2: Pros and cons of flat vs. hierarchical routing.

	Flat Routing	**Hierarchical Routing**
Optimal Routing	No	Yes
Scalability	Poor	Very Good
Network Lifetime	Good	Very Good
Resource Management	Poor	Good
Support for dynamic topology	Easy	Difficult

Clustering approaches in WSNs can be broadly divided into two main categories: *cluster-head first* approach and *cluster first* approach. Figure 3.5 illustrates an graphical example of clustering. In the cluster-head first approach, as the name suggests, cluster heads are elected first based on certain metrics, and they agree on how to assign other nodes to different clusters. In contrast, in cluster first approach, all the

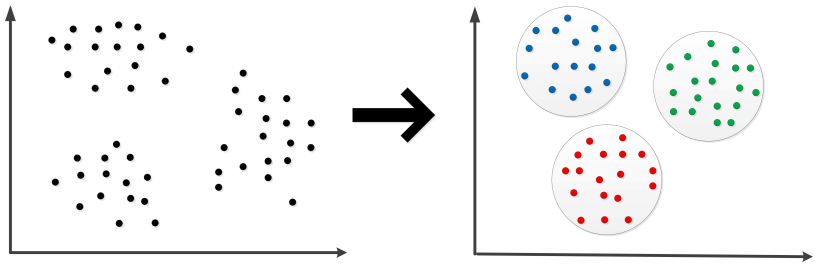

Figure 3.5: Clustering example

sensor nodes first form clusters, and each cluster then elects its cluster head [78]. In the next section, for each category, we will discuss most dominant approach in WSNs.

3.3.1 Cluster-head First Approach

LEACH [79] is one of the most popular cluster-head first clustering approaches in WSNs. It is a distributed, adaptive clustering algorithm where nodes make autonomous decisions without any centralized control. LEACH protocol runs multiple rounds. Each round contains two phases: cluster *setup phase*(when the cluster is organized) and *steady phase* (when data is transferred to the BS). More specifically the setup phase can be divided into three more steps: advertisement phase, cluster setup phase and schedule creation phase.

Advertisement phase: In this phase, each node decides based on a formula whether or not to become a cluster head for the current round. The formula is shown in Equation 3.1.

$$T(i) = \begin{cases} \frac{p}{1-p*(r*mod(1/p)} & if i \in G \\ 0 & otherwise \end{cases} \tag{3.1}$$

where variable p allow us to decide the desired percentage of CH node in the sensor population, r is the current round number and G is the set of nodes that have not been CHs in the last $1/p$ rounds. Now each node has to choose a random number between 0 and 1. If the random number is less than the calculate threshold, this node will be a good candidate. After this, each node that is elected as a CH will send a broadcast message advertising all nodes. In the next step, each non-cluster-head node decides the cluster to which it will belong for this round depending on the signal strength or the distance.

Joining phase: At this point, once the cluster head is selected, all nodes join the corresponding cluster by sending a message to the CH informing that it will be a member of that cluster. The decision is made based on the distance between the CH and the respective node considering the broadcast signal strength.

Schedule creation: CH receives all messages from nodes that would like to be in its cluster. Once the CH knows the number of member nodes, it creates a TDMA schedule, such that only one node will transmit in each time slot. Then, CH broadcast the schedule to its member nodes of the cluster. At this point the setup phase is completed and network enters a steady phase.

The steady phase is divided into slots, all nodes send their data to the cluster head at most once per frame during their allocated TDMA transmission slot. When all the data has been received (data aggregation), the CH will compress the information and transmits this to the base station. When the BS receives all the data aggregation from the CH, the CH sends a message to all member nodes to start another round. Figure 3.6 depicts a wireless sensor network protocol based on LEACH, which is divided into four clusters, the black circle in each cluster represents the cluster head, and the rest of the gray circles indicate non-cluster head nodes. Each cluster has a cluster head node. The protocol randomly selects cluster head node circularly, the energy of the entire network load is equally distributed to each sensor node which can achieve lower energy consumption and improve the network lifetime.

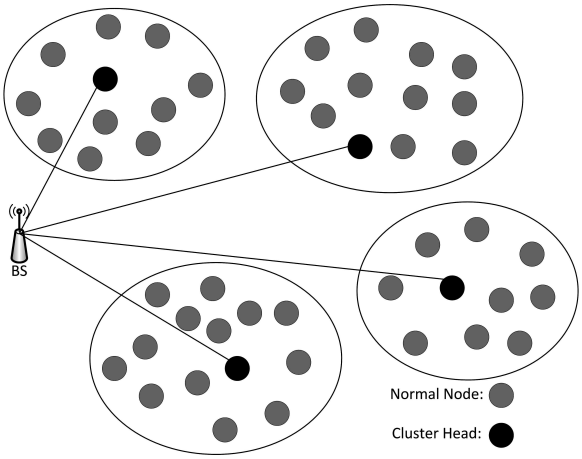

Figure 3.6: Architecture of LEACH

Unfortunately, LEACH assumes that all CH nodes have a one-hop connection with

BS which makes it impractical in geographically large wireless sensor networks where it is not possible to communicate with the BS in a single-hop manner. Moreover, the cluster heads are elected randomly, so the optimal number and distribution of cluster heads cannot be ensured. The nodes with low remnant energy have the same priority to be a cluster head as the node with high remnant energy. Therefore, to mitigate these limitations, some research [81, 86, 80, 87] works have extended this protocol for making it practically usable in WSNs.

3.3.2 Cluster First Approach

Clique [78] is another seminal approach in WSNs clustering where instead of selecting the cluster head first it starts with forming clusters with local information and then elects cluster head, thus it belongs to the cluster first set. All nodes in the same cluster must agree with the elected cluster head. In other words, all nodes have to be consistent with the view of its cluster (clique) and its cluster head. The protocol aims to divide the sensor network into multiple small groups and guarantee that all the nodes in each clique agree on the same clique membership. The protocol has the following properties:

- It is fully distributed and uses local information only. Each node computes its clique only using information from its 1-hop neighbors.
- It is guaranteed to terminate.
- After the protocol terminates, all nodes are divided into mutually disjoint clique. They have consistent view on their clique membership.

The protocol assumes that each node knows its 1-hop neighbors and they have and unique ID. Clique has four steps:

Many other techniques based on similar idea are proposed in [88, 83, 84]. One notable approach in this category is PANEL [82], is a position-based clustering routing protocol for WSNs. With respect to other CH election protocols, PANEL supports asynchronous sensor network applications where the sensor node readings are fetched by the BSs. The main goal of PANEL is to elect aggregators, i.e., CHs, for reliable and persistent data storage applications. PANEL assumes that the nodes are deployed

Algorithm 6 CLIQUE clustering steps

Input: A wireless sensor network with N nodes.

Output: A mutually disjoint set of cliques (clusters).

1: Each node exchanges its neighbor list with its neighbors, and computes its local maximum clique.

2: Each node exchanges its local maximum clique with its neighbors, and update its maximum clique according to its neighbor nodes local maximum clique.

3: Each node exchanges the update clique with its neighbors and derive its final clique.

4: Each node exchanges the final clique with its neighbors and check if they are consistent or not.

in a bounded area, which is partitioned into geographical clusters. The clustering is determined before the deployment of the network, and each node is pre-loaded with the geographical information of the cluster to which it belongs. PANEL is an energy-efficient protocol that ensures load balancing because each node is an elected aggregator, i.e., CH, nearly equally frequently. Besides, data aggregation is performed and communication load is reduced, accordingly PANEL can prolong the network lifetime. However, PANEL assumes that geographic position of the nodes are available at the node, it also assumes that clusters are determined forehand, which makes it impractical to support many types of dynamic WSNs applications.

Chapter 4

Proposed Key Management Protocol

In this chapter, we present the proposed key management protocol where we combine matrix-based key pre-distribution and pairing-based techniques. We begin with a general overview and assumptions of the network, and then we present mode of operations of the proposed scheme. Following this, we show how the proposed scheme can be set up in different types of networks, namely, in *flat* and *hierarchical* networks. We then introduce the different phases of key agreement processes involved in the protocol and present techniques for key refresh, key revocation and new node addition. Furthermore, we present some application scenarios where the proposed scheme can be used. We conclude with two techniques for optimizing the key agreement process of the proposed scheme.

4.1 Overview and Assumptions

In our proposed scheme, we assume that all sensor nodes are homogeneous in terms of computational capabilities and memory capacities. Although we assume a hierarchical deployment structure where a Key Manager (KM) node will manage some Member Nodes (MN), we do not assume the KM node to be any special or powerful node. Any node can act as a KM node for a particular time by following a KM node selection scheme [89, 90]. We only assume that BS node is trusted and powerful in terms of processing, memory, and power supply.

4.1.1 Optimization of Blom's Protocol

Blom [40] has presented a key distribution method that allows any pair of nodes in a network to be able to find a pair-wise secret key as long as no more than λ nodes are compromised. However, the original scheme proposed by Blom was not proposed for sensor networks. Therefore, to use it in resource constrained sensor network, a slight modification was done by [55]. The *key pre-distribution phase* of original Blom's

scheme (described in Section 3.1.1) uses $(\lambda + 1) \times N$ matrix G over a finite field $GF(q)$, where N is the size of the network and $GF(q)$ is a prime number large enough to accommodate the key size. In our scheme, G is chosen as *Vandermonde* matrix because it is well known that any $\lambda+1$ columns of G are linearly independent when the elements in the seeds (i.e., second row) of G are all distinct [91]. Everyone may have information about G matrix in our system. In our scheme, we used the node ID as seeds of G. Since all the node IDs are distinct, they constitute a linearly independent G matrix. Figure 4.1 illustrates the construction of the G matrix. Selecting G as *Vandermonde* matrix with node ID as seeds provide two fold benefits in our scheme:

(i) Each node now does not need to store the whole column of G matrix; instead knowing the ID (i.e., seed for that column), it can compute the whole column.

(ii) When a pair of nodes want to find a pair-wise key between them, they do not need to share the columns of the G matrix beforehand.

$$\begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 3 & \dots & N \\ (1)^2 & (2)^2 & (3)^2 & \dots & N^2 \\ (1)^3 & (2)^3 & (3)^3 & \dots & N^3 \\ & & & \vdots & \\ (1)^\lambda & (2)^\lambda & (3)^\lambda & \dots & N^\lambda \end{pmatrix}$$

Figure 4.1: G matrix construction.

4.1.2 Protocol Data Structure

Each sensor node in our scheme maintains data structures to compute pair-wise keys with other nodes. Table 4.1 provides a summary of data structures used in our protocol with their sizes in bits.

4.2 Mode of Operation

In our scheme, we support adaptive clustering and provide flexibility of setting up the networks for different type of applications and operations. Therefore, based on

Table 4.1: Summary of protocol data structure.

Acroname	Description	Size (Bits)
ID	Node ID	16
KM_λ	Vector of $\lambda + 1$ key materials	64 $\times(\lambda + 1)$
S_x	Private key for IBE	64
CP_x	Cache private key	64

application, the network administrator may need to determine two key features of the network: (i) the maximum number of nodes in each cluster; (ii) number of initial clusters in the network.

4.2.1 Determining Maximum Cluster Size

The size of a cluster is related to the desirable security threshold λ for that cluster and available memory size m of the sensor mote. The value of λ states that to compromise the whole cluster, an adversary needs to capture $\lambda + 1$ nodes. If a secure application requires a key size of q-bits and if motes have m bits of RAM for both programs and data storage. Assuming a portion of it (determined by a factor p, $0 < p \leq 1$), is used for storing keys, then using Equation 4.1, we can easily determine the theoretical security threshold value for that cluster.

$$\lambda = \lfloor \frac{m * p}{q} \rfloor \tag{4.1}$$

As an example, if an application requires $q = 64$-bits keys and uses a mote with $m = 10$ kB of memory (a typical MicaZ mote [92]), half of the total memory, that is $p = 0.5$, is used for storing keys then the security threshold, λ, would be calculated as, $\lambda = 81,920 * 0.5/64 = 640$.

There is no absolute equation for determining the cluster size, however, the value of λ provides a very good guideline to determine the cluster size. Based on the desired level of security of the application, one might decide on a cluster size. In general, if d_s is the ratio of the total number of compromised nodes in a cluster to C_n (which is the total number of nodes in that cluster), then $\lambda \geq C_n/2$ if d_s needs to be at least 0.5. The *maximum theoretical threshold* for a cluster size (C_n) is then can be derived as:

$$C_n = d_s * \lambda \tag{4.2}$$

Following the previous example, with 10 kB mote memory and 64-bits key size, theoretically a cluster would have maximum of $C_n = 640 * 2 = 1280$ nodes.

4.2.2 Determining Number of Clusters

Our protocol is flexible to support different types of network layer protocols. It supports both the cluster based hierarchical networks and flat networks. Based on maximum theoretical threshold for number of nodes in a cluster, initially one can choose how many clusters the network would have. If the underlying network layer requires no clustering, our system can be easily adapted to operate in that mode. Whereas, without any modification, our system can be also used with an adaptive clustering based network layer.

4.3 Setup Phase

Before the deployment of the network, the network administrator would determine the number of clusters in the network. Depending on application requirements and underlying network layers, one can select either flat or adaptive multiple-cluster setup phase.

A flat network setup only contains member nodes. The KM node is not required since no inter-cluster communication is necessary. In this setup, BS follows Blom's *key pre-distribution* phase with optimization discussed in Section 4.1.1 to calculate KM_λ ($\lambda + 1$ key materials or private shares) for each node in the network, where λ is a settable threshold for tolerance on the number of compromised nodes in the network and can be calculated using Equation 4.1. The BS then securely stores this KM_λ in the node's memory. An example of this setup is shown in Figure 4.2.

In an adaptive multiple-cluster network, the BS performs the same steps as described in flat cluster setup, separately for each cluster. In addition, the BS preloads KM_λ ($(\lambda + 1)$ secret shares) in the KM node as well as one more secret key (S_x) to perform pairing-based operations. To generate this additional key, the BS selects a master secret key S and calculates each KM node's private key. To do this, it

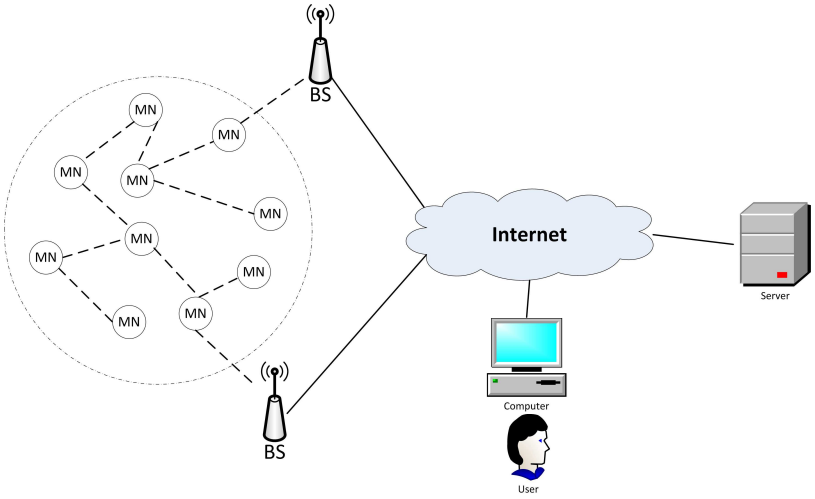

Figure 4.2: A example of flat network setup. Legend: MN = Member Node

first maps each KM node's identity to a point on the elliptic curve, via a public *hashing-and-mapping* function Φ. For example, if a KM node's ID is x, BS generates $P_x = \Phi(x)$. Following this, it generates its private secret key as $S_x = [s]P_x$. Then it preloads this S_x on the KM nodes as an additional secret. Figure 4.3 shows an example of three cluster network setup. Note that lines between KM nodes represent a logical connection. An actual path of the KM nodes may include other intermediate nodes.

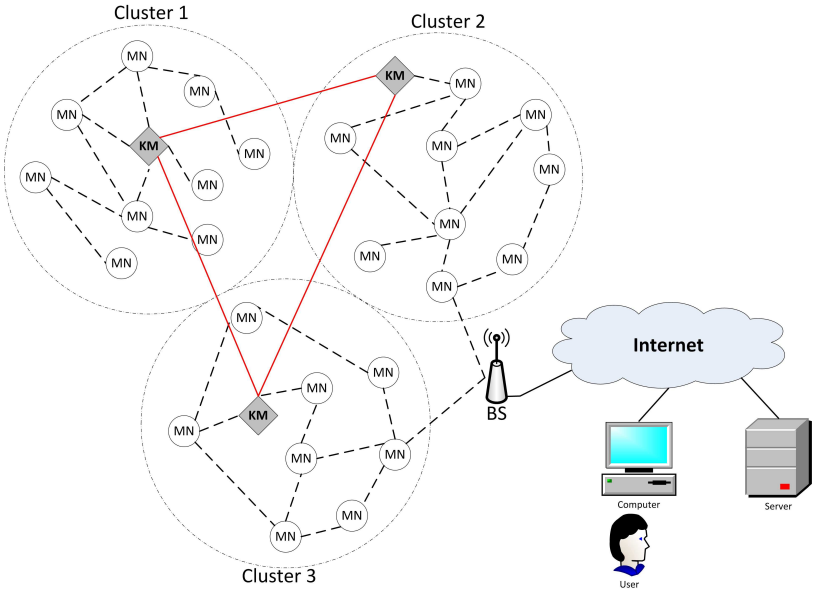

Figure 4.3: A example of multi-cluster network setup. Legend: MN = Member Node; KM = Key Manager; BS = Base Station

4.4 Key Agreement

Depending on setup, our scheme can have three key agreement phases as follows:

4.4.1 Key Agreement between Nodes of the Same Cluster

In this phase, assume two MN or KM nodes i and j, in the same cluster know the IDs of each other. They can compute the same secret without requiring any prior communication. This process is shown in Algorithm 7.

Algorithm 7 Computation of secret common key K_{ij} at $Node_i$.

Input: Node ID j.

Output: Secret common key K_{ij}.

1: Generates the column $c_j(G)$ of G matrix using j as the seed. Where G is a *Vandermonde* matrix constructed as described in section 4.1.1.

2: Compute $K_{ij} = c_j(G) \cdot r_i(A)$, where $r_i(A)$ is the i^{th} row of matrix A, which are secret information stored in node i.

When node j follows the same steps of the Algorithm 7, it can also generate K_{ji}. Note that $K_{ij} = K_{ji}$ since K is symmetric matrix [40]. Thus both the nodes i and j now can establish a secure communication using this shared secret.

4.4.2 Key Agreement among KM Nodes

Suppose two KM nodes, i and j from different clusters have to agree on a common cryptographic key. Using pairing-based operation and only knowing each other's ID, they can derive a common secret as follows:

$$\hat{e}(S_i, P_j) = \hat{e}([s]P_i, P_j) = \hat{e}(P_i, P_j)^S = \hat{e}(P_i, [s]P_j) = \hat{e}(P_i, S_j) = \hat{e}(S_j, P_i) \qquad (4.3)$$

Note that node i possesses S_i (see the setup phase) and can compute $P_j = \Phi(ID_j)$. Likewise, node j possesses S_j and can compute $P_i = \Phi(ID_i)$. Therefore, both i and j are able to compute the same secret as: $K_{ij} = \hat{e}(S_i, P_j) = \hat{e}(S_j, P_i)$.

4.4.3 Key Agreement between Nodes from Different Clusters

When two member nodes from different clusters want to communicate with each other, they need to establish a secure path via some KM nodes. Let us assume that node i from one cluster wants to communicate with node j in another cluster. In such a scenario, node i performs the following steps:

i. Node i finds a KM node in its own cluster and sends an encrypted message to that KM node using the procedure described in the Section 4.4.1 (*key agreement with nodes among same cluster*).

ii. The KM node then sends the message to another KM node belonging to the cluster of node j using the procedure described in the Section 4.4.2 (*key agreement among KM nodes*).

iii. Finally, the KM node of node j sends the encrypted message to node j using the same procedure described in the Section 4.4.1 (*key agreement with nodes among same cluster*).

Figure 4.4 illustrates this process. It is easy to see that the above scheme can be converted so that two nodes from different clusters can generate a temporary secret session key via the above procedure and then can establish a secure communication directly with each other. One such process is described in Section 4.8.1.

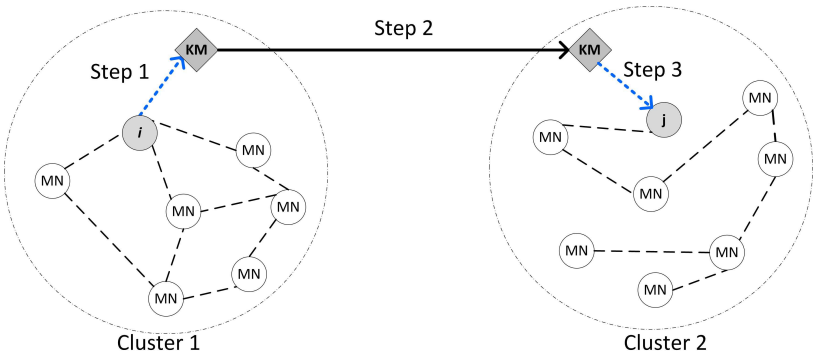

Figure 4.4: Key agreement with nodes from different clusters. Legend: MN = Member Node; KM: Key Manager

Figure 4.5 illustrates all the scenarios of key agreement procedures.

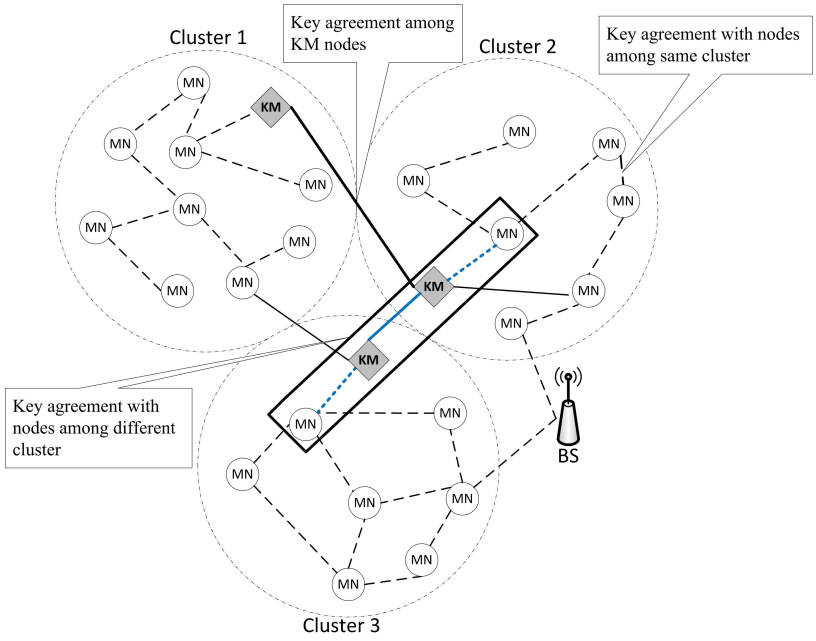

Figure 4.5: Different phases of key agreement. Legend: MN = Member Node; KM: Key Manager

4.5 Key Refresh/Revocation

A key management protocol is not complete if it does not provide mechanisms for revocation and key refresh. Given indefinite time, any network can be compromised if it does not refresh the key periodically. In addition, it is necessary to revoke the key of the compromised node once identified to continue with normal network operation. In our scheme, the base station is responsible for revoking the keys of the compromised nodes. We assume that with the help of intrusion detection systems, such as [93, 94], the compromised nodes can be detected and a list, R composed of the IDs of compromised nodes is sent to the base station. Let us say, R = $(n_1, n_2, \cdots, n_r)$ are the IDs of the compromised nodes in the cluster. The list of compromised node may contain the KM nodes as well as NM nodes. For each cluster the BS performs the key revocation operation separately. For each cluster, the BS initiates the key revocation process as follows:

i. It generates a new random $(\lambda + 1) \times (\lambda + 1)$ symmetric matrix $\bar{D}$ over $GF(q)$ for that particular cluster, and computes an $C_n \times (\lambda + 1)$ matrix $\bar{A} = (\bar{D} \cdot G)^T$, where $(\bar{D} \cdot G)^T$ is the transpose of $\bar{D} \cdot G$ for that cluster.

ii. It then generates a bit vector X of length C_n, where C_n is the number of nodes in that cluster, such that $X[i] = 0$ if the node ID i is in R, 1 otherwise.

iii. Then it computes the $C_n \times (\lambda + 1)$ matrix $H = \bar{A} + A$ for that cluster, and performs $r_i(H) \cdot X[i]$, where $r_i(H)$ are the elements of the i^{th} row of H.

iv. To revoke the additional key of the KM node, BS selects a new master secret key $\bar{S}$. If the node i is a KM node then it computes a new private secret $\bar{S}_i = [\bar{S}]P_i$ for that node. Then it computes a mask $S_h = S_i + X[i] \cdot \bar{S}_i$.

v. For each non-revoked MN node i in that cluster, the base station computes:

$$M_i = i \| E_{k_{ii}}(r_i(H)) \| MAC(K_{ii}, i \| E_{k_{ii}}(r_i(H)))$$

where $\|$ is the message concatenation and K_{ii} is computed as $c_i(G) \cdot r_i(A)$. $E_{k_{ii}}(\cdot)$ is any suitable symmetric encryption algorithm for WSNs using k_{ii} as key. Whereas, for each non-revoked KM node, the BS computes M_i such as:

$$i \| E_{k_{ii}}(r_i(H)) \| S_h \| MAC(K_{ii}, i \| E_{k_{ii}}(r_i(H)) \| S_h)$$

vi. Finally, the BS computes $M = M \| M_i$, where $i \notin R$ and broadcasts $R \| M$. The base station then updates D as $\bar{D}$ and S as $\bar{S}$ for future revocation.

vii. When a non-revoked MN node t in a cluster receives the revocation broadcast, it extracts its own share, $t \| E_{k_{tt}}(r_t(H)) \| MAC(K_{tt}, t \| E_{k_{tt}}(r_t(H)))$ from the broadcast message. It then computes:

$$V = MAC(K_{tt}, t \| E_{k_{tt}}(r_t(H)))$$

where K_{tt} is determined using the node's private share as: $c_t(G) \cdot r_t(A)$. If V matches with the received MAC, then it means the broadcast have come from the base station node. After this the node decrypts $E_{k_{tt}}(r_t(H))$ using k_{tt} and updates its private share (KM_λ) as $r_t(A) = r_t(H) - r_t(A)$. Otherwise, it discards the broadcast message. Whereas, when a non-revoked KM node receives the revocation broadcast, it performs the same above operation, in addition to this, it extracts the S_h value from the message and updates its private secret by $\bar{S}_t = S_h - S_t$. It is easy to see that after the update is done, the non-revoked

node stores $r_t(\bar{A})$ and/or $\bar{S}_t$ whereas the revoked node will not be able to update its key share and therefore will be unable to take part in future communications with non-revoked nodes.

The key refresh is a special case of key revocation with R being an empty list.

4.6 New Node Addition

To add a new MN node in a cluster, the BS node follows the new node addition process as follows:

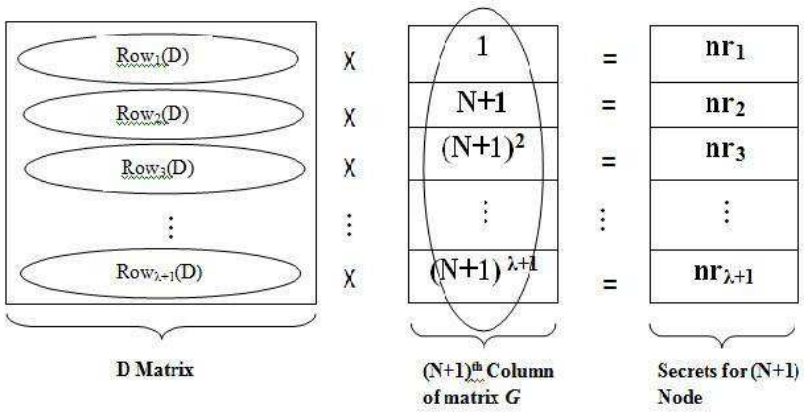

Figure 4.6: Adding new node.

The BS pre-loads the seed value (i.e., node ID) and private secret key shares to the new node's memory. For the sake of simplicity, let us assume that the new node ID is $N+1$, then private secret shares for node $N+1$ can be computed as shown in Algorithm 8.

Figure 4.6 illustrates the new node addition process. However, if a new KM node needs to be added in a cluster, then the BS follows the same algorithm described in Algorithm 8 as well as computes an additional private secret for KM node. The additional new private secret is created as follows, if the new KM node ID is ID_{new} then BS first maps the new node's ID to a point on the elliptic curve by $P_{ID_{new}} = \Phi(ID_{new})$. Following this, a new private secret for the new node will be generated as $S_{ID_{new}} = [s]P_{ID_{new}}$, where s is the current master secret key for the network.

Algorithm 8 Computation of symmetric key for new node.

Input: New node ID $N+1$.

Output: K_λ is the private secret shares of node $N+1$.

1: Extend the matrix G by adding a new column using $N+1$ as the seed.
2: **for** $i = 1$ to $(\lambda + 1)$ **do**
3: **for** $j = 1$ to $(\lambda + 1)$ **do**
4: $KM_\lambda[i] = KM_\lambda[i] + D[i][j] \cdot G[i][N+1]$
5: **end for**
6: **end for**

4.7 Application Scenarios

Our proposed scheme is flexible to support many application requirements. In this section, we provide some example applications where our protocols can be easily used without modifications.

4.7.1 Adaptive Cluster based Network Applications

Most common WSNs applications require periodic data reporting with thousands of static nodes spread randomly over a large area. Examples include micro-climate, structural health, volcano monitoring, disaster recovery, pipelines, etc [14, 26, 16]. These and related applications exhibit several common characteristics:

- They consist of many unattended energy-restricted fixed or static nodes, which may fail unexpectedly. Further, new nodes may be added to the network.
- The nodes communicate over lossy wireless channels.
- Reasonable delivery delays are acceptable, e.g., in agricultural monitoring.
- The data gathered at individual nodes can be pre-processed in the network, e.g., correlating related data, compressing, or filtering.
- The destinations can be multiple and mobile, e.g., consider scientist with a laptop walking along agricultural fields with sensor network.

- The potential clusters are often geographically specified and the nodes have some position or orientation information, e.g., room or floor of the building they are installed in, geographic area, etc.

To facilitate such applications, the network is usually divided into clusters. Clustering a large network provide two fold benefits; (i) optimized resource utilization (ii) easy resource management. Every cluster usually has a cluster head node for central management of the cluster and many member nodes under it. Beside management, the CH can also optimize the sensing data by aggregating redundant reading by similar member nodes. However, since WSNs nodes are energy-restricted and can fail unexpectedly, a fixed cluster strategy with fixed CH node is not suitable. Thus the network has to be adaptive to dynamic nature clusters. Many studies have focused on this issue and proposed promising network layer protocols for adaptive clustering in WSNs [83, 74] (refer to Section 3.3 for an overview of adaptive clustering protocols in WSNs).

Key management protocols for WSNs should support the underlying network layer by providing easy mechanisms to support this dynamic nature. Our proposed protocol provides complete support for adaptive cluster-based networks. Any kind of adaptive clustering approach, such as LEACH [79] or CLIQUE [78] can leverage our scheme to provide security on top of clustering. Adaptive multiple-cluster setup mode of our scheme can be used to support adaptive network clustering mechanism. In this setup, the CH nodes can be considered the KM nodes, and member nodes would be other nodes in the cluster. Whenever the cluster size is changed or a new cluster head is selected the following steps are performed:

i. The new CH or KM node will contact the BS with list of member nodes under it. This message can be encrypted by the currently held key of that node.

ii. BS calculates the partial shares for all the nodes in that cluster and initiates key refresh process for that cluster. This will refresh the key of the KM node as well.

4.7.2 Secure Group Communication

Many WSNs applications such as network and environmental monitoring, air quality monitoring, forest fire detection, etc., require that the data gathered at individual nodes can be pre-processed in the network, e.g., correlating related data, compressing, or filtering. These kinds of applications may require temporary group of nodes to cooperate with each other by forming a dynamic group. To facilitate such requirement our protocol can be used to form a dynamic secure group.

Assume a node n_m wants to initiate a group G_m with the group members $\{n_1, n_2 \ldots n_m\}$, where $n_i : i = 1 \ldots m$, is the node ID. Node n_m then performs the following steps to create and distribute the group key G_k for group G_m:

i. It computes $G_k = K_{n_1,n_m} \oplus K_{n_2,n_m} \oplus \ldots \oplus K_{n_m,n_m}$. Where $\oplus$ represents bit-wise XOR and K_{n_i,n_m} is the pair-wise key between node n_i and n_m.

ii. Then it computes $Y_i = G_k \oplus K_{n_i,n_m}$, `for i=1 ...m-1`

iii. The node n_m then broadcast $M_g = (n_1, Y_1) \parallel (n_2, Y_2) \parallel \ldots \parallel (n_{m-1}, Y_{m-1})$, where $\parallel$ is message concatenation.

iv. Receiving the broadcast message M_g, the group member node t extracts its share Y_t and computes the group key $G_k = Y_t \oplus K_{n_m,n_t}$, here K_{n_m,n_t} is the pair-wise key between node t and n_m.

As an example of the group creation, let us consider that node i wants to initiate a group with nodes (1, 3, 5, 8) and it shares pair-wise key with each of these nodes $(K_{i_1}, K_{i_3}, K_{i_5}, K_{i_8})$, respectively. Now node i generates the group key

$$G_k = (K_{i_1} \oplus K_{i_3} \oplus K_{i_5} \oplus K_{i_8} \oplus K_{i_i})$$

and broadcasts

$$\begin{aligned} M_g = \{&1, (K_{i_3} \oplus K_{i_5} \oplus K_{i_8} \oplus K_{i_i}) \\ &\parallel 3, (K_{i_1} \oplus K_{i_5} \oplus K_{i_8} \oplus K_{i_i}) \\ &\parallel 5, (K_{i_1} \oplus K_{i_3} \oplus K_{i_8} \oplus K_{i_i}) \\ &\parallel 8, (K_{i_1} \oplus K_{i_3} \oplus K_{i_5} \oplus K_{i_i})\} \end{aligned} \tag{4.4}$$

When any group member node, say node 5, receives the group creation broadcast M_g, it extracts its part P from M_g and computes the $G_k = K_{5_i} \oplus (P)$, which is in turn $(K_{i_5} \oplus K_{i_1} \oplus K_{i_3} \oplus K_{i_8} \oplus K_{i_i}) = G_k$. On the other hand, a non-group member node cannot compute the group key G_k using its pair-wise key and received broadcast message M. Let us now consider a non-group member node j receives the broadcast M from node i. In our scheme, each pair of nodes uses different pair-wise keys for communication. This implies that the pair-wise keys $K_{j_1}, K_{j_3}, K_{j_5}, K_{j_8}$, and K_{j_j} between node j and nodes 1, 3, 5, 8 are completely different from the pair-wise keys between node i and 1, 3, 5, 8 which are $K_{i_1}, K_{i_3}, K_{i_5}, K_{i_8}$, and K_{i_i} respectively. Therefore, it is not possible for node j to compute the G_k from its pair-wise keys between 1, 3, 5, 8 nodes and received broadcast message M_g.

4.7.3 Dynamic Network Applications

Sensor networks are often deployed in hostile environments, such as military operations, monitoring marine habitats, tracking targets, floating sensors in rivers, etc. In such applications the size of the network may not be very large, but requires mobility and time sensitive data in a highly secure fashion. A long distance peer-to-peer secure communication between sensor nodes with end-to-end encryption and authentication may be required. Thus, one primary goal of secure communication is to provide authentication and/or encryption between any sensor nodes. Moreover, in such applications, it is very difficult to physically protect the sensor nodes individually. A few sensor nodes could be captured by adversaries or may destroyed after deployment. Once a sensor node is captured, it can give away its secrets very quickly. Therefore, a dynamic and robust key management is required to satisfy these requirements. Our scheme can be used in a single cluster mode to accommodate these types of applications. It can quickly compute separate pair-wise keys for each node without any extra communication. The pair-wise keys can be used to provide authentication, integrity, and privacy between any pair of nodes in the network. Moreover, our scheme provides a fair amount of resilience against compromised sensor nodes.

4.8 Optimization

In this section, we describe two optimization techniques for performance improvement that can be applied on the proposed scheme. Both optimizations can be applied in the *key agreement between nodes of different clusters* phase described in Section 4.4.3.

4.8.1 Creation of Temporary Keys for Different Clusters

Consider a node ID_a in a cluster under key manager node ID_i. Let us assume that ID_a wants to generate a common key with node ID_b of another cluster with KM node ID_j. Node ID_a will then send a request message to its KM node (ID_i) to generate a key K_{ab}. This request can be sent in plain text. To generate the key K_{ab}, KM node ID_i follows the steps described in Algorithm 9:

Algorithm 9 Shared key generation process for nodes from different clusters

Input: Common secret key of two KM nodes K_{ij}, node IDs ID_a, ID_b.

Output: New secret key K_{ab}, for node ID_a and node ID_b.

1: The KM node, ID_i uses a keyed-hash function to generate a common key as follows:

$$K_{ab} = Hash(K_{ij}, ID_a, ID_b)$$

2: and computes:

$$CT = E_{K_{ia}}(K_{ab})$$

where E is any symmetric key-based encryption algorithm and K_{ia} is the shared secret key of the KM node ID_i and node ID_a. Note that the shared key, K_{ia} is computed following the steps described in Algorithm 7.

3: After this, the KM node, sends the encrypted message, CT to node ID_a.

Node ID_a will then decrypt this message using K_{ai} and use the key K_{ab} to send a secure message to node ID_b. Notice that, one component of K_{ab} is K_{ij}, since $K_{ij} = K_{ji}$, node ID_b can also follow the same Algorithm 9 with its KM node ID_j and get K_{ba}. Figure 4.7 depicts such a scenario.

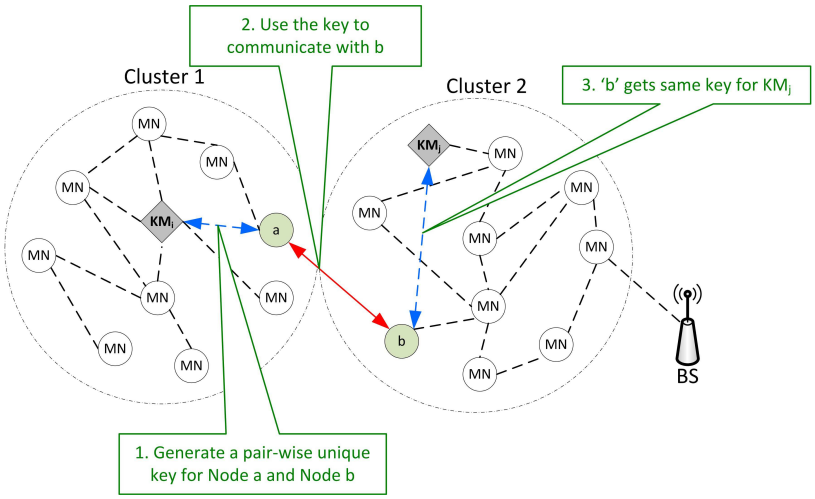

Figure 4.7: Computation of common key for the nodes from different clusters. Legend: MN = Member Node; KM = Key Manager; BS = Base Station

4.8.2 Communication Optimization

Depending on the application, the KM nodes may be more powerful in terms of power and processing than the MN nodes. Since radio communication consumes most of the energy in WSNs [95], some applications might want to make a trade-off between computational time and energy consumption. Here, we extend the process of the key agreement of nodes from different clusters (described in Section 4.4.3) by removing a communication step.

Let us assume node i from one cluster wants to communicate with node j in another cluster via a KM node KM_j then it has to perform the following steps:

i. First, node i will send an encrypted message to the KM node (KM_j) of j's cluster. To send an encrypted message to KM_j, node i first uses the public *hashing-and-mapping* function Φ and generates the public key of KM_j as $P_{KM_j} = \Phi(KM_j)$. Then, it uses Algorithm 4 with public key P_{KM_j} to encrypt the message.

ii. When KM_j node receives encrypted message from other cluster, it follows Algorithm 5 to decrypt the message using its private key S_k. Finally, KM_j forwards the message to node j using the procedure described in *key agreement with nodes among same cluster*.

This optimization is shown is Figure 4.8.

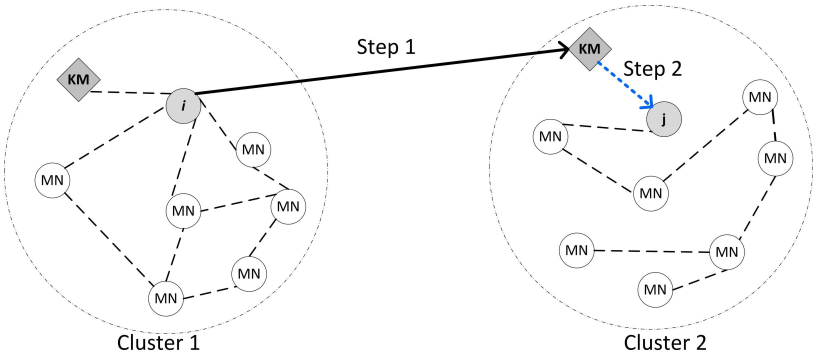

Figure 4.8: Optimized key agreement phase with nodes from different clusters. Legend: MN = Member Node; KM = Key Manager

Although this optimization removes a communication step from the proposed scheme, it comes at a price. Now, the sensor nodes need to use the IBE based encryption (Algorithm 4) and decryption algorithms (Algorithm 5) in order to send the data packet to the KM node. It is well known that asymmetric cryptosystems are computationally expensive than symmetric based cryptosystem [34]. Thus this optimization will increase computational cost over communication cost.

4.9 Summary of Contributions and Advantages

The main objective of our proposed scheme is to provide efficient key management framework for resource constraint networks such as WSNs. The main contributions are given below:

1. We have improved Blom's scheme as a result without communication and only knowing each other's ID, any pair of nodes can establish pair-wise keys.

2. We have combined pairing based cryptography with symmetric cryptography to make the key management protocol scalable without increasing storage requirements.

3. We have provided efficient mechanism for key revocation, key material refresh, and new node addition without compromising security.

4. Without modification, our scheme supports multiple type of networks, such as clustered, group, or flat networks.

5. Authentication and key distribution in one set of protocols. Only nodes that share common key materials will be able to establish a secure channel.

6. Improved network resiliency and lower memory utilization. Compromise of one node compromises only a small portion of network. This provides little value without the additional compromise of more than λ nodes, as specified by Blom's scheme and supported in our algorithm.

7. Our scheme is scalable to large networks and suits many different types of applications.

Chapter 5

Performance Analysis and Experimental Results

In this chapter, we provide a detailed description of experimental setups, implementations and results of our proposed protocol. We then compare our scheme with contemporary schemes to measure performance.

5.1 Metrics for Performance Evaluation

5.1.1 Energy Overhead

Since nodes are battery powered and may be deployed in hostile environments, it is not feasible to frequently change the power source in such networks. Hence, energy is the most valuable and scarce resource in WSNs.

5.1.2 Computational Cost

Our scheme comprises many cryptographic operations. We will evaluate the performance for each operation. Then, the performance costs for each operation are accumulated to attain the total cost. Each cryptographic operation needs to be processed within a certain period of time, which can be viewed roughly as performance cost. Therefore, like other research works, this thesis assumes that the performances of these cryptographic operations can be measured by time. In the case of resource constraint WSNs, computational cost is very crucial because it has a direct impact on time and energy consumption.

5.1.3 Communication Overhead

The communication overhead consists of the number of packets exchanged. It also depends on length of the message. Since WSNs are battery operated, energy is the most valuable resource. Radio communication consumes most of the energy in WSNs [95].

Therefore, communication overhead is a very important metric for measuring performance of a key management protocol.

5.1.4 Memory Overhead

Memory is very limited in sensor motes. Therefore, the protocol has to make optimal use of memory. For the sake of fairness, we used same length keys for every nodes in the network. We assume the key length is standard on the contemporary security protocols for WSNs.

5.2 Implementation Software and Tools

5.2.1 TinyOS

TinyOS [96] is an open source, BSD-licensed operating system designed for low-power wireless devices, such as those used in sensor networks, ubiquitous computing, personal area networks, smart buildings, and smart meters. Industry and academia use it worldwide for developing, testing, and running WSNs applications. TinyOS uses an interface to build components of an application. However, unlike traditional programming interfaces, TinyOS interfaces are bi-directional. In addition to the ability for the consumer of an interface to issue a command to its provider, providers have the ability to signal event callbacks to consumers. This relationship allows for a split-phase, non-blocking paradigm to emerge. The same paradigm that is typically seen in hardware [96].

In the split-phase model, the consumer of an interface signals a request for a particular service. The provider of the interface is then responsible for performing the service, and signalling an event to the consumer when the service has completed. Programmatically, in order for these operations to be non-blocking, TinyOS introduces the concept of tasks, which are code segments that are scheduled to run at a later time. Tasks are put in a run-to-completion, first-come, first-serve queue. In essence, one component calls a command on another component. The receiving component creates a task to be completed, and when this task is eventually run to completion, it signals an event back to the original calling component.

5.2.2 NesC

NesC (network embedded systems C) [97] is the programming language of TinyOS. It is a programming language with a set of cooperating tasks and processes. NesC is built as an extension to the C programming language with components wired together to run applications on TinyOS. TinyOS and nesC are the de-facto standard for developing industrial standard and research applications for WSNs. NesC complies its raw code into a form that the gcc compiler can use, after which gcc builds the TinyOS application into bytecode that can run on a mote or in a simulator. Since nesC is a specialized version of the C language designed for use on wireless sensor networks, it has some fundamental differences from that of traditional C. One of the largest differences is that nesC is a static language and does not allow for the dynamic allocation of memory. This prevents memory fragmentation and run-time memory errors, such as null pointer exceptions. In addition to this, nesC does not allow function pointers. This allows the compiler to know the call path of an application at compile time and perform more rigorous optimizations on the code, resulting in crucial memory savings.

5.3 Hardware Selection

Choosing the right sensor mote is an important consideration for any WSNs deployment. A number of "off-the-shelf" motes are available, including the Mica2 [98], MicaZ [92], IMote2 [99], and TelosB [100]. Table 5.1 compares the features of these motes. As we will use these motes for research, not a real-world application, the physical size of the mote would not be a concern. The ideal mote for our research would be simple to program and have enough memories to store readings from experiments, with a low price.

5.4 Methodology and Experimental Setup

To test the proposed key management schemes, we have implemented the key agreement phases on TelosB (TPR2420) motes, the latest research oriented mote developed by UC Berkeley [100]. TelosB has MSP430 microcontroller. MSP430 incorporates an 8MHz, 16-bit RISC CPU, 48K bytes flash memory (ROM) and 10K RAM. The RF

Table 5.1: Comparison of mote specifications

Mote	**TelosB**	**Mica2**	**MicaZ**	**IMote2**
Processor Speed	8 Mhz	16 MHz	16 Mhz	13-416 Mhz
Processor Current Draw	1.8 mA	8 mA	8 mA	13-66 mA
Memory Size	10 KB	512 KB	512 KB	32 MB
Radio	CC4220	CC1000	CC4220	CC4220
Radio Receive Current Draw	23 mA	10 mA	19.7 mA	44 mA
USB Interface	Yes	No	No	Yes
Integrated Sensor Board	Yes	No	No	No

transceiver on TelosB is IEEE 802.15.4/ZigBee compliant, and can have 250kbps data rate. Figure 5.1 shows an image of Crossbow TelosB mote. We have used NesC programming language to implement our protocol on TinyOS [96]. The key size $GF(q)$ is kept to 64 bits, where q is a large prime number that requires 64 bits to store. We choose 64 bits key size because TinySec [35], which is a fully implemented protocol for link-layer cryptography in sensor networks, currently uses 64 bits or 8 bytes key for cryptographic operation. To improve the accuracy of our results, we repeated the simulations at least 20 times.

Figure 5.1: Crossbow TelosB sensor mote

5.4.1 Energy Consumption

The energy consumption E can be calculated by

$$E = U \cdot I \cdot t \tag{5.1}$$

where U is the voltage, I is the current and t is the time duration. TelosB motes are powered by two AA batteries, so U is approximated equal to three volts. The

current-supply value varies in different operations as shown in Table 5.2 (abstracted from [100]).

Table 5.2: Current drawn for different operations on TelosB

Operation	**Normal**	**Max**
MCU On, Radio Off	$1.8mA$	$2.4mA$
MCU On, Radio Receiving	$21.8mA$	$23mA$
MCU On, Radio Transmitting	$19.5mA$	$21mA$

To calculate energy of different phases of our protocol, we first compute the processing time of each phase then use Equation 5.1 to calculate energy consumption. Unless a phase in our protocol specifically requires data transmission, we use the current draw metric as normal "Micro Controller Unit (MCU) on, Radio off" mode of data processing. Besides, communication time can be estimated in the following way. Given 250kbps radio transmission rate, and 29 bytes in each packet (maximum payload size of TinyOS 2.0.2), the time it takes one sensor node to send or receive a data packet can be calculated as:

$$\frac{29 \times 8\, bits}{250\, kbps} = 0.928\, ms \tag{5.2}$$

Without considering message loss and retransmission, the total transmission time for pkt packets can be calculated as:

$$\frac{29 \times 8\, bits}{250\, kbps} \times pkt \tag{5.3}$$

5.5 Performance Analysis

Our proposed scheme has a set of setup tasks that BS node performs at the network setup phase. To simplify the experiments, we have simulated the setup phase that a BS node would do in our protocol. In real deployment scenarios, the BS would compute the key materials or private key shares for each node and preload it into node's memory before the deployment of the network. The key agreement is the most important phase of our protocol since two sensor nodes, in that phase, will compute the common secret keys. Our proposed scheme has three different phases of key agreement. We have implemented each phase separately and measured the time

and energy cost individually. Finally, we have accumulated these costs to analyze the computational and energy consumption of entire system.

In our experiments, the code was kept to the bare minimum for key computation and turning on the LEDs at the start and end of process. When the program runs, it lights up an LED, computes the pairwise key, and lights up another LED on completion.

5.5.1 Key Agreement between Nodes of the Same Cluster

One essential part of our proposed scheme is the key agreement of nodes in the same cluster. In this phase, we used modified Blom's scheme. Each node stores $(\lambda + 1)$ private key materials. To compute the pair-wise key, a node first computes $x^2, \cdots, x^\lambda$, where x is the ID of the other node. The node then evaluates a λ-degree polynomials. In general, we are interested in measuring the cost of the λ-degree polynomial computation in sensor mote. Given a share of the polynomial $f(x) = a_0 + a_1 x + \cdots + a_t x^\lambda$ over $GF(q)$, the computation of $f(x)$ requires λ modular multiplications and λ modular additions, plus the computation of values $x^2, \cdots, x^\lambda$. In our implementation, we have chosen q to be 64 bits. Therefore, λ (64-bit $\times$ 64-bit) modular multiplications are required to compute the polynomial. On the 16-bit CPU of TelosB, each 64-bit $\times$ 64-bit multiplication costs 16 word multiplications. To further reduce the computational cost, we adopt the simplification proposed in [57]. The simplification is based on the fact that variable x is the sensor ID, which is normally a 16-bit integer. We can use another finite field $GF(q')$ for $x, x^2, \cdots, x^\lambda$. Therefore, the modular multiplication in polynomial $f(x)$ is always performed between a 64-bit integer and 16-bit integer. As the result, the cost of multiplication is reduced by four times. The modular reduction operation is as important as modular multiplication. Each multiplication must be followed by a reduction operation. To further reduce the computational cost, we pick a pseudo-Mersenne prime as q because modular reduction cost on field of a pseudo-Mersenne prime can be optimized to a negligible amount. A pseudo-Mersenne prime can be represented as $q = 2^m - \omega$, where $\omega << 2^m$. Given a $2m$-bit multiplication result $B = (b_1, b_0)$, (b_1, b_0 are two m-bit halves), the reduction can be computed based on the congruence $2^m \equiv \omega$:

$$\begin{aligned} &\text{while } (b_1 \neq 0) \\ &\quad (b_1, b_0) = b_1 * \omega + b_0 \\ &B = b_0 \bmod q. \end{aligned} \tag{5.4}$$

In our experiment, we chose $q = 2^{64} - 2^8 - 1$, $q' = 2^{16} - 2^4 - 1$. We tested the average time delay and power consumption for computing the polynomial with different λ values. In each test, we randomly generate $\lambda + 1$ 64-bit coefficients and a 16-bit variable x, we repeat 20 times to get the average time delay. We also calculated average memory consumption of our protocol using protocol data structure described in Section 4.1.2 which is directly proportional to λ, since every node needs to store $(\lambda + 1)$ 64-bit key materials in its RAM. The test results are shown in Table 5.3 (The complete experimental results from all runs are given in Appendix A).

Table 5.3: Computation cost of λ-degree polynomial evaluation in sensor node

λ	16	32	64	128
average time consumption	7.9ms	16.8ms	34.94ms	74.07ms
average power consumption	$42.66\mu J$	$90.72\mu J$	$188.94\mu J$	$399.98\mu J$
average memory consumption	154 bytes	282 bytes	538 bytes	1050 bytes

The test results show that the key agreement is efficient even in low-power sensor nodes. With λ value 128, we could imagine a sensor network of at least 385 motes, where the network would be secure even with one-third compromised nodes. Results show that in this setting any pair of nodes can establish a pair-wise key in 74.07 ms and consumes 399.98 μJ.

5.5.2 Key Agreement among KM Nodes

In an adaptive multiple-cluster setup, a KM node establishes keys with other KM nodes using pairing-based operations. The most time-consuming part of any PBC is computing the pairing itself. We have used TinyPBC [47], an efficient PBC implementation of the Tate pairing (realized in the form of the η_T [101] pairing over supersingular binary curves and a variant of the Tate pairing over prime fields [101]) for resource-constrained nodes. It is an open source library written in both NesC and C language, compatible with TinyOS. The source code is available at URL http://sites.google.com/site/tinypbc/. It is by far the most optimized PBC based

library in TinyOS environment. Once again, we simulated the BS operation and pre-loaded the private key into node's RAM. In our protocol, before calculating common keys using PBC, it is necessary to map the nodes ID to an elliptic curve point so that the PBC operation can be applied on it. We can use any hash-and-map algorithm for this. However, in our implementation we have used the process described in Algorithm 10. We have used a binary field $\mathbb{F}_{2^{271}}$, a square root friendly polynomial $f(x) = x^{271} + x^{207} + x^{175} + x^{111} + 1$, given in [47]. We have used this polynomial because TinyPBC library uses the same polynomial for its calculations.

Algorithm 10 Hash-and-map-to-point Algorithm in elliptic curves over binary field $\mathbb{F}_{2^{271}}$

Input: Node ID, $N_i \in \{0,1\}^*$.

Output: Point on a elliptic curve *f*.

1: Set i = 0 and j = 1
2: $digest = Hash(i||N_i)$ where $Hash : \{0,1\}^* \rightarrow \{0,1\}^l$ is a conventional cryptographic hash function. For example, if $Hash$ is SHA-64 based, then $l = 64$.
3: Encode every two bits of digest to the j^{th} digit of element $x \in \mathbb{F}_{2^{271}}$. Set $j = j + 1$.
4: **if** *digest* is not long enough for generating all the 271 digits of x **then**
5: go back to step 2.
6: **end if**
7: compute $f(x)$ where f is the elliptic curve equation.
8: **if** $f(x)$ has two roots y_0, y_1 **then**
9: set (x, y_0) as the output point and return if i is even.
10: set (x, y_1) as the output point and return if i is odd.
11: **else**
12: set $i = i + 1$ and go back to step 2.
13: **end if**

In Table 5.4, we summarize the time and energy consumed for computation of common key between KM nodes. The complete experimental results from all runs are given in Appendix A.

Our findings align with findings from [47]. Compared with highest time and energy cost (i.e., for $\lambda = 128$) of key agreement between nodes in the same cluster,

Table 5.4: Computation cost to calculate η_T-pairing on TelosB mote.

average time consumption	2.2s
average power consumption	$11.88mJ$
average RAM consumption	732 Bytes

the time and energy consumed in this phase is much higher. However, note that after computing a common key between KM nodes, they can derive a symmetric key from it and use that symmetric key for future communications, thus KM nodes can limit the use of expensive pairing-based operation and improve overall performance.

5.5.3 Secure Communication between Nodes from Different Clusters

In this section, we use Algorithm 9 from Section 4.8.1 to describe the performance analysis of the key agreement process for nodes among different clusters. In Table 5.5 we show the basic operations of each step from Algorithm 9 and their associated computational and energy costs. At the last row we accumulate all costs and show the total cost of computing a common key between nodes from different clusters when a common shared key between KM nodes is already cached on KM nodes.

Table 5.5: Total cost of common key computation for nodes from different clusters without key agreement of KM nodes.

Operations	**Time**	**Energy**
Step 1: *One* SHA-1 hash	4.66ms [35]	$25.168\mu J$
Step 2: *One* TinySec-AE encryption [35]	0.38ms [35]	$2.054\mu J$
Step 3: Send *one* packet	0.928ms (Equation. 5.2)	$54.288\mu J$
Step 4: Receive *One* Packet	0.928ms (Equation. 5.2)	$60.69\mu J$
Step 5: *One* TinySec-AE decryption [35]	0.38ms [35]	$2.054\mu J$
Total:	**7.276ms**	**$144.524\mu J$**

In addition, Table 5.6 shows the total cost of computing a shared key between nodes from different clusters when a common shared key between KM nodes does not exist.

5.5.4 Key Refresh/Revocation Performance

In key refresh phase, the BS broadcasts a list with updated key materials to all nodes in the network. Table 5.7 provides a summary outlining the individual operations

Table 5.6: Cost of common key generation for nodes from different clusters including the key agreement of KM nodes.

Computation of common key between KM nodes	2.2s	$11.88mJ$
Accumulated cost from Table 5.5	7.276ms	$144.524\mu J$
Total:	**2.2072s**	**12.02**mJ

that nodes have to do at this stage and their associated computational and energy consumption costs. The calculation presented in Table 5.7 uses $\lambda = 128$.

Table 5.7: Cost of key refresh.

Operations	**Time**	**Energy**
Step 1: *One* λ-degree polynomial evaluation	74.07ms	$399.98\mu J$
Step 2: *One* SHA-1 hash	4.66ms [35]	$25.178\mu J$
Step 3: *One* TinySec-AE decryption [35]	0.38ms [35]	$2.054\mu J$
Step 4: (λ) 64-bit subtractions	0.128ms	$0.691\mu J$
Total:	**79.238ms**	**427.90**μJ

In the case of KM node, the cost of key refresh is the total cost from Table 5.7 plus one 64-bit subtraction (which is very negligible to consider).

5.5.5 Scalability and Proof-of-concept

It would have been ideal if we could run the protocol on large number of devices and test the overall performance. However, such an implementation requires more than thousand nodes to be deployed in a large geographical area, which is costly and infeasible to conduct in a lab environment. Therefore, we chose to simulate on the computer using simulation tools as viable alternative. We simulated our scheme on a Java platform. We have simulated a network of 5000 sensor nodes. We used multiple thread mechanism where activities of each node are implemented as a separate thread. We have used inter thread communication to simulate communication between nodes. For simplicity, we assumed that all nodes can directly connect with each other and there is no underlying routing protocols. However, we divided the network into five clusters and we distributed 1000 nodes randomly to each cluster. Each node was given a unique ID and a field to represent which cluster it belongs to. We selected a cluster head (KM) node for each cluster and every node knew their cluster head node

ID beforehand. We have used PC with Intel i7 dual core (2.80Ghz/core) CPU and 8GB of RAM. For this simulation, we have selected $\lambda = 128$. We have simulated the BS node, which calculated the key materials for each node in every cluster. Every node maintains a data structure defined in Section 4.1.2.

To test the key agreement, a tester thread randomly generates a cluster:node ID pair, such as c1:ID1 and c2:ID2. Then, it sends the c1:ID1 to node with ID2 and c2:ID2 to node with ID1 through inter-thread communication. Each node then generates key individually and sends the computed key back to the tester thread, the tester thread then verifies whether they are same or not. To improve the accuracy of our results, we repeated each test at least 50 times. Figure 5.2 shows two such instances: first, key agreement of nodes in same cluster, second key agreement of nodes from different clusters.

```
Key agreement in same cluster...
Node 53: 587 received from Node c3:587
Node 53: Calculated key for Node 587 Key53_587:23006883589206829904
Node 587: 53 received from Node c3:53
Node 587: Calculated key for Node 53 Key587_53:23006883589206829904

Key agreement in different clusters...
Node 136: 341 received from Node c1:341
Node 136: Calculated key for Node 341 Key136_341:3410255669121234535
Node 341: 136 received from Node c4:136
Node 341: Calculated key for Node 136 Key341_136:3410255669121234535
```

Figure 5.2: Simulation snapshot: key agreement phase

```
After adding a new node...
Node 1: 1001 received from Node c2:1001
Node 1: Calculated key for Node 1001 Key1_1001:1796160628191206789
Node 1001: 1 received from Node c2:1
Node 1001: Calculated key for Node 1 Key1001_1:1796160628191206789

After Key Refresh...
Node 416: 7 received from Node c3:7
Node 416: Calculated key for Node 7 Key416_7:11317253699760701842
Node 7: 416 received from Node c3:416
Node 7: Calculated key for Node 416 Key7_416:11317253699760701842
```

Figure 5.3: Simulation snapshot: new node addition and key refresh

We have implemented the new node addition process in our simulation and added a new member node in a cluster. The simulation shows that other nodes in the cluster can successfully setup keys with the newly added node. Figure 5.3 shows a snapshot of such a scenario. We have also simulated the key revocation process of our protocol

by randomly generating list of revoked node IDs. After this, the BS node calculates the new key shares for all the nodes in a cluster except the revoked nodes and sends notification to all the nodes in that cluster with key revocation message. Following this, each node updates their private shares after verifying the hash of the message. The BS node then repeats this process for all five clusters. After this phase, we again performed the key agreement phase with the revoked and the non-revoked nodes. The simulation showed that key generated between revoked and non-revoked nodes do not match. However, the key generated between two non-revoked nodes are still the same. Following similar process, we also refreshed keys for all the nodes in every cluster. Figure 5.4 depicts a snapshot of key revocation followed by a key agreement. It shows that revoked nodes cannot establish keys with non-revoked nodes.

```
After revocation...
Revocation List:2,30,4,50,6,70,8,90,10,
Node 261: 182 received from Node c1:182
Node 261: Calculated key for Node 182 Key261_182:4691215544081502639
Node 182: 261 received from Node c1:261
Node 182: Calculated key for Node 261 Key182_261:4691215544081502639

After revocation... (communication with revoked node)
Revocation List:2,30,4,50,6,70,8,90,10,
Node 714: 30 received from Node c1:30
Node 714: Calculated key for Node 30 Key71430:28252348174219015000
Node 30: 714 received from Node c1:714
Node 30: Calculated key for Node 714 Key30714:0
```

Figure 5.4: Simulation snapshot: key revocation/refresh

The simulation gives a proof-of-concept of our proposed scheme and shows possibility of using the system in large-scale networks.

5.5.6 Overall Performance of Proposed Scheme

In this section, we summarize the overall costs of our proposed scheme. The communication and memory overheads at each Phase of our proposed protocol is shown in the Table 5.8. Then, in Table 5.9, we show computational and energy consumption overheads at each phase of our proposed protocol. Finally, Figure 5.5 graphically depicts these overheads. The result shows that the largest overhead of our protocol comes from the Phase 3 of Table 5.9, which uses pairing-based operations to compute common key among nodes from different clusters. However, Phase 3 is only required when KM nodes have not cached the keys of other KM nodes. Furthermore, overheads

of our protocol are significantly reduced when KM nodes can cache the common keys of other KM nodes (Phase 2, Table 5.9).

Table 5.8: Memory and communication overheads of our scheme

Memory requirement of each nodes

MN node	$(\lambda + 1) \cdot q$ bits
KM node	$(\lambda + 1) \cdot q + q$ bits

Number of communications required

Key Agreement of MN nodes (intra-cluster)	0
Key Agreement of KM nodes (inter-cluster)	0
Key Agreement of MN nodes (inter-cluster)	1 $packet(< 29\, bytes)$ exchange
Key refresh/revocation	1 broadcast; Total size: $N \cdot (\lambda + 2) \cdot q$ bits
New node addition	0

Table 5.9: Summary of computation and energy cost of our protocol

Protocol phases	**Computational costs**	**Energy costs**
Phase 1: Key agreement, MN nodes (intra-cluster)	74.07ms	$399.98\mu J$
Phase 2: Key agreement, MN nodes (inter-cluster, without KM key calculations)	7.276ms	$144.52\mu J$
Phase 3: Key agreement, MN nodes (inter-cluster, with KM key calculations)	2.207s	$12.02mJ$
Phase 4: Key refresh/revocation	79.23ms	$427.90\mu J$

5.6 Comparison of Proposed Scheme

Computation, *communication* and *storage* complexity are used to measure the efficiency of security schemes in WSNs [62]. Whereas *scalability*, *connectivity* and *resilience* are used for measuring performance. In the following subsections, we compare our scheme with other schemes in terms of *efficiency* and *performance*.

5.6.1 Efficiency Comparison

It is important to quantify and compare the efficiency of our scheme with other schemes in the same category. We compare storage, communication, and computation overheads of our scheme with the key pre-distribution scheme proposed by Eschenauer

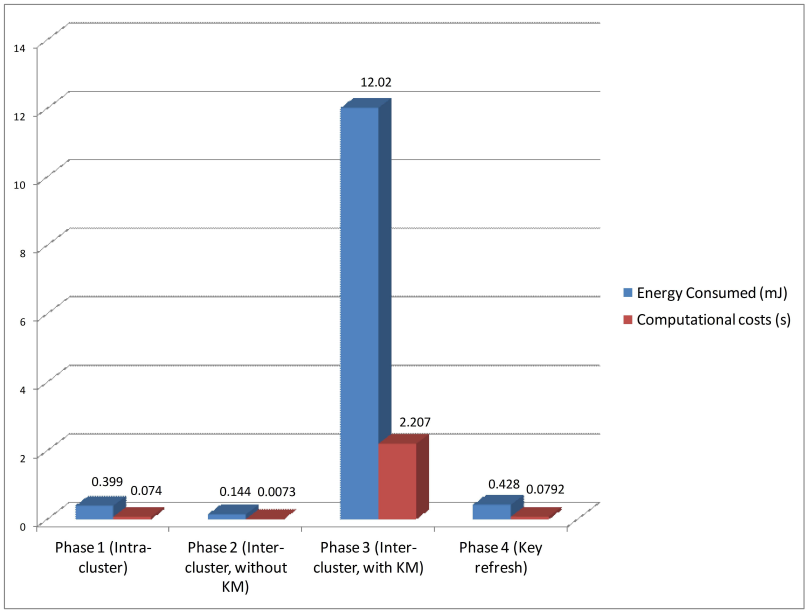

Figure 5.5: Summary of computation and energy cost of our protocol

and Gligor [38] refereed as EG protocol, centralized ECC based scheme [102] refereed as C-ECC, and TinyIBE [48] scheme.

Storage Cost

Let us assume a network composed of M KM nodes and C_n MN nodes, λ is the security threshold for each cluster. Typically, we have $M << C_n$ and $\lambda \leq \lfloor \frac{(C_n/M)}{2} \rfloor$. In our proposed key management scheme, we used identity-based schemes for generating both inter and intra cluster keys. Therefore, neither MN nor KM nodes need to preload private or public keys of other sensor nodes on the network. Each node only needs to store its secret key or key materials assigned to it by the BS. Hence, after deployment of the network and without any communication, each node can compute its shared secret with each of its neighbouring nodes by computing either pairing-based or polynomial-based operations. Thus total storage requirement (in terms of key-length unit) for a network with $M + C_n$ sensor nodes and λ security parameter can be derived as:

$$((M + C_n) \times (\lambda + 1)) + M \tag{5.5}$$

In the case of EG scheme, the probability of connectivity of a network with $(M +$

C_n) nodes is calculated by the following equation:

$$P[Match] = 1 - \frac{((P-k)!\,)^2}{(P-2k)!\,P!} \tag{5.6}$$

where P is the key pool size, k is the size of the key ring size. With a given probability of connectivity the total number of preloaded keys in the network can be calculated as:

$$(C_n + M) * k \tag{5.7}$$

Similarly, for TinyIBE scheme, each KM node is pre-loaded with *three* keys. Thus total number of pre-loaded keys in the network is:

$$3 * M \tag{5.8}$$

In the C-ECC scheme each MN node stores a private key and public keys of other KM nodes. Besides, KM node is preloaded with a pair of private/public key for itself and with the public keys of other MN nodes. In addition, it also stores a key K_h for newly deployed nodes. Thus each KM sensor node is preloaded with $C_n + 3$ keys. In that case the total number of preloaded keys in the network is:

$$M * (C_n + 3) + 2 * C_n = (M + 2) \times C_n + 3M \tag{5.9}$$

As an example, let us consider a WSN with $M = 50$ clusters and $C_n = 500$ member nodes evenly distributed among all the clusters. With this network and using $\lambda = \lfloor \frac{(500/50)}{2} \rfloor = 5$, the total memory requirements in our proposed scheme is 3350 keys. Please note that with the $\lambda = 5$, an adversary has to compromise at least half of the member nodes in every cluster to compromise the entire network. The EG scheme is normally applied in a flat sensor network. Therefore, considering a flat network setup, let us assume a key pool of $10,000$ keys and each $M + C_n = 550$ sensor nodes are pre-loaded with 200 keys from the key pool. For a network with a connection probability not exceeding 85%, the total storage requirement would be $550 * 200 = 11000$ keys. This is about 228% more than the total storage requirement of our proposed scheme. To increase the connection probability in the EG scheme, we must also increase the size of the key rings, as a result increasing the total key storage of the network. In contrast, the storage requirement of our proposed scheme is independent of node connectivity. For the C-ECC scheme, which is an asymmetric

key-based scheme, the total storage requirement is 26150 keys, which is almost 680% more than the total storage requirement of our proposed scheme. On the other side, TinyIBE scheme has a storage cost of only 150 keys, which is much less than the total network storage required in our proposed scheme. Although TinyIBE has less memory cost, the operations in TinyIBE is only based on expensive pairing-based calculations. As a result it incurs high computation and energy overheads.

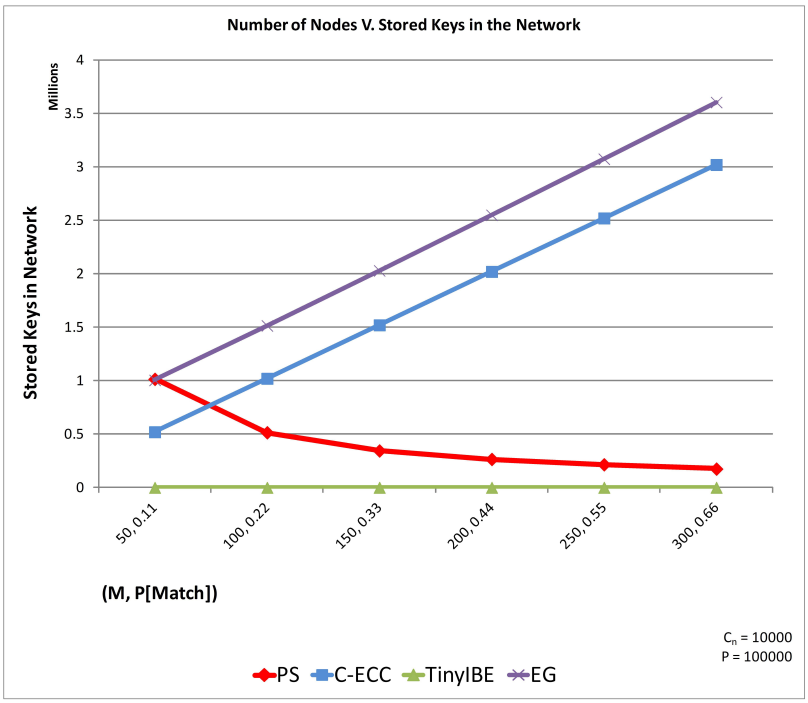

Figure 5.6: Storage requirements of the proposed and other schemes for large networks.

In Figure 5.6 we plot the total storage requirements of our proposed scheme and other schemes by changing the size of networks. To plot the graph we used $C_n = 10,000$ member nodes in each cluster and we have increased the number of clusters (KM nodes, i.e., M) from 50 to 300 every time increasing by 50. We also used $p = 100,000$ for calculating storage requirements of the EG protocol and varied the connectivity probability from 33% to 66%. The graph indicates that as the number of clusters (i.e. M) increases, our scheme performs well compared to other schemes except TinyIBE. It also indicates that the proposed protocol is scalable in terms of memory, which means it can be feasibly used in very large sensor networks without increasing network memory overhead.

Communication Cost

Energy is precious in WSNs. The most energy-consuming operation in WSNs is wireless communication. It consumes 80% of the total energy of a sensor mote [103]. Therefore, it is important that transmissions be used carefully. In our proposed scheme, to establish a secret key with any KM or MN node in the same cluster, it does not require any prior message exchange. Only knowing each other IDs any pair of nodes can establish a pair-wise key. However, if nodes from different clusters need a pair-wise key, it must activate its transmitter twice: once for sending a request to the KM node and another for receiving an encrypted message of $< 29\,bytes$ from the KM node. Thus only one *packet* is exchanged by a member node to set up a key with another member node from different cluster.

In the case of other comparative schemes, each node must exchange at least one message to establish a secret key. However, the size of the message varies from one scheme to another. For example, the message broadcast by a node in the EG scheme contains the list of key identifiers on its key ring. If the key pool contains $P = 10,000$ keys and each node is preloaded with $k = 100$ keys and with a connection probability of 65%, then each key identifier requires 14-bit. Which can be calculated as: $\lceil log_2 P \rceil$ and the broadcast message have a size of 14-bits $*k = 14 * 100 = 1400$-bits or 175 bytes.

In the C-ECC scheme, components include a public key which is a point on the elliptic curve, its hash value, and a signature comprising two integers provided by the trusted authority. Using 160-bits, an authenticated ECDH scheme would require exchange of at least 768-bits or 96 byte.

Now we evaluate the message length of TinyIBE. Using this protocol, the exchanged message contains two values C_1 and C_2. C_1 is a point on the elliptic curve $E(F_2^{271})$, which can be compressed to 34 bytes and C_2 has the size of the session key $(128 - bits)$. The resulting message has 52 bytes. For our proposed pair-wise key establishment, typically, nodes in a cluster will need pair-wise keys with its neighbor in the same cluster thus most of the time we do not require any message exchange. Therefore, our proposed scheme is much more efficient compared to EG scheme and TinyIBE. We summarize the comparison of communication costs in Table 5.10.

Computational Cost

In this section we compare the computational overhead of our proposed scheme with EG,C-ECC, and TinyIBE schemes. The EG scheme introduces a high storage and communication overheads, but it has very minimal computational overhead. In this scheme, a node performs a search in its key ring to find a matched shared key with other node. Whereas the TinyIBE makes two steps (encrypt and decrypt) on-line to establish a pairwise session key. The encryption steps involve two hashing operations, two point multiplications, one exponentiation, one addition, and one XOR operation. The decryption steps require one η_T pairing calculation and one hashing to retrieve the session key. On a TelosB mote, result shows it takes 2.62s to perform key computation in their protocol. On the C-ECC scheme, it uses Elliptic Curve Diffie-Hellman (ECDH) protocol for key computation and Elliptic Curve Digital Signature Algorithm (ECDSA) for signature verification in a MICAz mote. The results showed that with all optimizations enabled, the execution times were: ECDSA initialisation 3,393 ms, verification 2,436 ms, and for ECDH initialisation 1,839 ms, and key computation 2,117 ms. Hence, key computation and signature verification can take 4.5 seconds, after initialisation of about 5.2 seconds.

Typically communications in WSNa are limited in neighboring nodes in the same cluster. In our scheme, key agreement among same cluster nodes take a negligible time, $74.07ms$ compared with C-ECC and TinyIBE schemes. Besides, key generation of nodes from different clusters also shows better result, 2.207s, in our scheme compared with other schemes. If KM node cache the common key of the other cluster node, then the time for generating the common keys for nodes from different clusters reduces to 7.276ms which is almost 3470 % less than TinyIBE. In Table 5.10 we summarize the communication, storage and computation costs of all the schemes. It clearly shows when considered scalability (storage), number of communications and computational cost of our scheme is better than other compared schemes.

Energy Cost

The total energy consumed in the proposed scheme is $12.99mJ$, which is calculated by adding energy consumption cost of every phase from our proposed scheme shown in Table 5.9. This is less than 1% of the total available energy in a 3V battery. The

Table 5.10: Comparison of efficiency of our scheme with different schemes

Scheme	**Storage**	**Communication**	**Computation**
EG scheme	$(M + C_n) * t$	175 Bytes	*search*
C-ECC Scheme	$M * (C_n + 3) + 2 * C_n = (M + 2) \times C_n + 3M$	96 Bytes	4.5 Sec
TinyIBE	$3 * M$	52 Bytes	2.62 Sec
Our scheme	$((M + C_n) \times (\lambda + 1)) + M$	**0, 29 Bytes**	**74.07 ms, 2.209 Sec**

M: # of KM nodes, C_n: # of MN nodes, λ: security parameter

most important task of a sensor mote is sensing. We also compare the total energy consumed by different phases of our protocol with energy consumption of a typical sensing operation [95]. The graph in Figure 5.7 shows total energy consumption of different phases of our proposed scheme versus energy consumption of a single sensing operation on a sensor mote. It indicates that the total energy consumed by our protocol is lesser than that by a typical sensing operation, even with Phase 3 (Table 5.9) considered. This implies that our protocol is feasible for deployment in real-time applications.

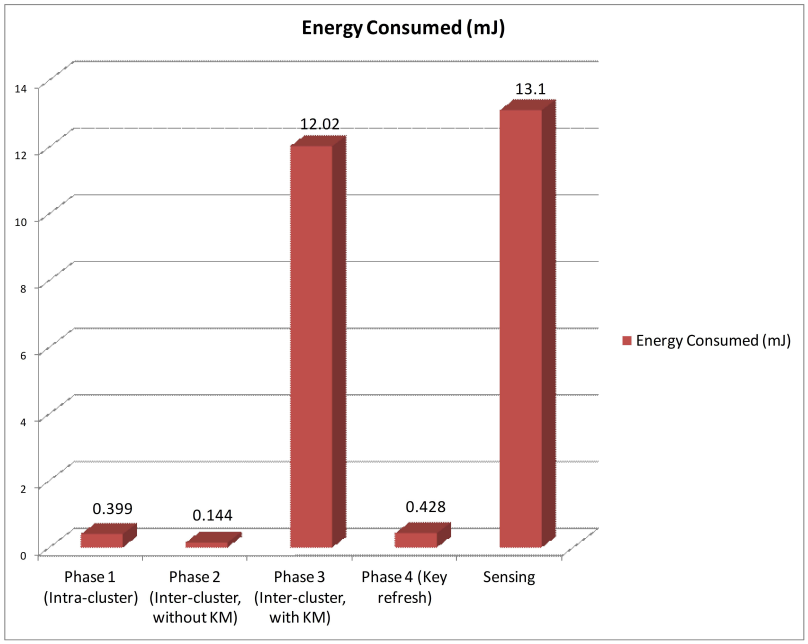

Figure 5.7: Energy overheads of proposed protocol vs. sensing

5.6.2 Performance Comparison

Apart from only being able to provide basic protection, i.e., help in maintaining confidentiality and integrity of information and authenticating the users through secret keys, a key management scheme in WSNs should be able to provide *scalability*, *connectivity* and *resilience*. These criteria can be used to measure performance of key management scheme in WSNs [62]. Connectivity means the connection probability for two nodes have the same predistributed key or establishing a key path between them. Scalability means whether a scheme support sensor node revocation/addition for large wireless sensor network. Resilience means the probability that a link is compromised when an adversary captures a node or the number of sensors required for adversary to compromise the whole wireless sensor network. In Table 5.11, we compare the performance of our proposed scheme with key management schemes discussed in literature survey (Chapter 2). Note that we only show the basic schemes of each category. The variation schemes may have many differences with the basic scheme.

Table 5.11: Performance comparison of proposed scheme with other schemes

Scheme	**Scalability**	**Connectivity**	**Resilience**
Single-shared key	High	1	1
Pair-wise key pre-distribution	Low	1	0
Trusted third-party based scheme	High	0	
Probabilistic key pre-distribution	Medium	$\frac{((KP-k)!)^2}{((KP-2k)!\,KP!)}$	k/KP
Matrix-based key pre-distribution	High	1	λ
Polynomial-based key pre-distribution	Medium	1	λ
Location-aware key management	Medium	1	λ
C-ECC	High	1	0
TinyIBE	High	1	0
Our scheme	**High**	**1**	λ

KP: Key pool; k: Key ring

Chapter 6

Security Analysis

In this chapter we present the security analysis of our scheme. Depending on the network size and security requirements of the application one might choose to use only pair-wise distributed symmetric based key management (i.e., for flat network type). To facilitate this option, our proposed protocol can easily be degenerated to a pair-wise distributed symmetric key protocol as shown in Section 4.3, if it is deployed in a single cluster mode, the security of the protocol then follows from Section 3.1.3 and would depend on the value of λ, which states that how many nodes the attacker needs to compromise before he/she can compromise the entire network.

However, if the network size is very large, then one can choose to distribute the deployment of nodes among different clusters. In that case the security of each individual cluster is the same as the security of one cluster scenario. With such setup, the overall security of the network is improved since, an attacker now has to compromise $M \times \lambda$ nodes, where M is the number of clusters, to compromise the whole network. Additionally, to communicate between two clusters the KM nodes use IBC to generate the keys. The security of IBC is based on the hard Bilinear Diffie-Hellman Problem (BDHP), which is believed to be difficult to reverse in real-time [67]. A security analysis of IBC is presented in Section 3.2.4.

Apart from general security discussion, in the following sections we present how our protocol can withstand some specific security threats.

6.1 Spoofed or Replay Attack

In this type of attack, an attacker may spoof the wireless communication channel and capture data communication between nodes. Later the attacker replays the captured message to gain unauthorized access or make the system de-synchronized. In *key agreement* phase of our proposed system, only knowing each others ID and without any message exchange, communicating parties can generate secret symmetric key.

Therefore, an attacker cannot capture and replay messages. However, in the *key refresh* phase, the base station node sends key update materials to all the nodes in the network. A malicious attacker may capture this update message and replay it later to launch a de-synchronization attack. To prevent such an attack, note that the BS encrypts such broadcast messages by each node's private key (K_{ii} where i is the node ID), as well as a hash of the message is computed and attached to verify integrity. After receiving and extracting its part from the broadcast message, each node first checks the integrity of the message by calculating a hash of the received message using its currently held private key and comparing it with received MAC. If the computed MAC matches with the received MAC then, it updates its key. If an attacker replays a captured broadcast message from BS, the integrity check would fail because the node would have already updated its private key. Hence, the receiving node will discard the replayed update/refresh broadcast as bogus message and prevent replay attack.

6.2 De-synchronization Attack

A de-synchronization attack is launched with an intention to make communicating parties reach at a different state in their protocol, where they cannot find common credentials for the communication. In the proposed system, an attacker may prevent key refresh/update broadcast message to be reached to a particular node. As a result, all nodes will update their key materials except that particular node. Consequently, the other nodes will not be able to establish a shared key with it. This would lead to a de-synchronized state in the system.

To overcome such a situation, when a non-revoked node is unable to establish a shared key with other non-revoked node, it sends a request to the BS node to resend the key update/refresh materials for both nodes. For instance, assume node i wants to send an encrypted message to node j. Node i uses its currently held key K_{ij} for the encryption and computes MAC of the message. Then, it sends the encrypted message and the MAC to node j. When node j receives this message it first decrypts the message and computes MAC of the received message. To verify the integrity of the message, node j then compares the computed MAC with the received MAC. If they do not match, then there might by two cases; (i) the key materials used for computing

K_{ij} or K_{ji} are different on either side, or (ii) some intruder may have modified the message in transit. In both cases, the receiving node, (i.e., node j) sends a request to the BS node to resend the key material update/refresh for both nodes.

Upon receiving such request, the BS node, first keeps a record of the request and queries both nodes for their current key material values. Once the current key material values are identified, BS node re-sends the key update/refresh materials to both nodes separately. If the BS node receives two such requests within a certain period from the same set of nodes, then it initiates intrusion detection for that part of the network to determine if the network is under attack. Thus, even though we cannot completely prevent the de-synchronization attacks, we can detect and recover from it.

6.3 Denial-of-service Attack

Denial-of-Service (DoS) attack is very difficult to prevent in any system. It is done with the intention of disrupting the availability of legitimate services. Many DoS attacks require man-in-the-middle (MiM), where an intruder can actively capture and modify/drop the data packets. In our scheme, all the broadcast and unicast message transfers are sent encrypted and a MAC is attached to identify any modifications. If multiple phenomena are detected, where integrity check fails, nodes can request BS node to trigger intrusion detection system to identify any potential threats. Another simple way to launch a DoS attack is by jamming the radio signal. An intruder may jam the radio signal and disrupt or prevent any message exchange between the BS node and other nodes. Such attacks are difficult to prevent. However, our protocol should still perform normally with the non-jammed part of the network.

6.4 Physical Node-capture Attack

As WSNs are deployed in unattended mode, capturing and reading memory contents of a node is very easy for an attacker. It is challenging to prevent such attacks. In our scheme, we only store partial key materials in the memory of a node. If an intruder captures and reads the memory contents of a node, it can only compromise messages that are transferred between that node with other nodes. However, the attacker will

not be able to compromise communication of any other set of nodes. To compromise the entire network, in each cluster, the attacker needs to capture and read the memory of λ nodes within a short period of time, which makes it hard in the context of WSNs.

6.5 Forward and Backward Secrecy

Forward secrecy means that the sender's private key is compromised, but the attacker still cannot recover any previous message M from $E_k(M)$ which is an encrypted text that the sender sent to somebody before. In contrast, the *backward secrecy* means that after compromising the sender's private key, an attacker cannot compute keys that would enable him to decrypt future messages that it would send. In our scheme, if an attacker captures a node and gets hold of its private key materials, then it can only decrypt messages that were exchanged between that node and other nodes until the next key update/refresh phase. Once the key materials are refreshed, the new keys are completely unrelated to old keys. As a result, the attacker cannot infer anything from the current key materials, and thus, it preserves *forward* and *backward* secrecy.

6.6 Traffic Analysis

In this type of attack, an attacker captures a significant amount of data from the network and then performs cryptanalysis on the captured data, to gain knowledge of the encryption and keys used for the encryption. In our scheme, performing *traffic analysis* attack would be difficult because as soon as the keys are refreshed, it gets updated to completely unrelated new key. Moreover, the attacker has to perform the cryptanalysis and break the security within two key refresh/update phases.

Chapter 7

Conclusion

Key management is a pre-requisite to provide security in WSNs. Because of unique constraints of WSNs, key management schemes that are used in existing wireless networks, such as WiFi, WiMAX, MANET, etc, are not applicable in WSNs. Moreover, large network sizes and the lack of physical protection of sensor nodes make key management harder in WSNs. Symmetric key cryptosystems are lighter in terms of computation, hence many research works have used such systems to address the key management problem in WSNs. However, in a very large sensor networks, symmetric key-based cryptosystems require each node to store a large number of keys. Due to the constraint on memory in WSNs mote, many such schemes thus become unfeasible. In contrast, PKI requires fewer number of keys, but expensive in processing and energy consumption. Signature verification is the most computationally expensive operation in PKI system. Recent improvements on pairing based encryption techniques have made it possible to use in WSNs. In PBC, Node's unique ID is used for key generation, thus keys are self-authenticated and trusted third-party is not necessary. Large scale WSNs are typically divided into clusters to optimize resource utilization. A key management protocol should support the clustered as well as flat networks.

In this thesis, we have presented a hybrid, robust, and non-interactive key management protocol for WSNs. It combines the pairing based scheme with matrix based key pre-distribution scheme to propose a novel key management protocol for WSNs. We provide a scalable key management solution by combining the two schemes that has better resilience against node capture attack and less memory requirement. It supports both flat and clustered network type without modification. The proposed scheme is very flexible and can be adjusted to support the need for many applications. Because of the computational overhead of PBC, in our scheme, we use the PBC based scheme at the bootstrapping phase only. After the bootstrapping, nodes can cache keys for future use, thus we can limit the use of PBC computation very

occasionally, hence improving the overall performance of our protocol. Our protocol is non-interactive and ID based, as only IDs are required, in particular no extra public key data is needed, as a result it uses very less message exchanges and thus less energy consumption. We do not claim that our scheme is sufficient for providing security in WSNs and that our scheme is immune from attacks. We do, however, claim that our protocol is the most feasible solution for many different types of WSNs applications where it requires scalability, and resilience against node capture, given that there is no secure storage in WSNs node and memory is scarce. In addition, our scheme can adopt dynamic network changes without impacting any existing nodes.

The proposed scheme provides mechanisms for new node addition and eviction of compromised nodes. Additionally, it provides key refresh mechanism without compromising security. We have implemented and tested the result of our proposed scheme in TinyOS using TelosB sensor motes. We then used simulation with result from real devices to measure performance of our scheme in large-scale network. We measured efficiency of our scheme in terms of computation, memory, communication, and energy consumption. To validate our scheme we have compared the results with other promising schemes in WSNs. Result shows feasibility of using our scheme in very large networks without increasing memory overhead. Our scheme requires less computation and provides better security than other schemes. The communication overhead in our scheme is much less, as a result it consumes less energy. We have also conducted exhaustive security analysis of our scheme which shows resilience against many types of attacks. To the best of our knowledge, this is the first protocol in WSNs which combines the matrix-based key pre-distribution solution with pairing based solution. One important aspect of the proposed scheme is that since it is light-weight, it can be easily extended to other similar resource constraint emerging networks, such as MANET, Radio Frequency IDentification (RFID), and Near Field Communication (NFC).

This thesis provides an effective key management technique that will be of significant value to designers of future WSNs and help ensure that their deployment provides a safe and reliable environment for many important applications. As future work, our proposed scheme could be extended to provide access control services in a network. The keys or key materials stored on a node can be used as required knowledge for

authorizing access to a particular zone or service. When a node requests access to a specific resource in a network, legitimate nodes in the network can be configured to verify that the requesting node has a knowledge of the keys / key materials required for accessing the said resource. Furthermore, our proposed key management scheme can be extended to heterogeneous wireless sensor networks where cluster head nodes are powerful in terms of processing, memory and energy. A secure routing structure can also be devised as an application of our proposed key management scheme (when used in a flat network setup) so that the network throughput can be increased, data packet delay can be improved, and routing overhead can be reduced.

Advances in wireless sensor mote hardware in terms of processing, storage, and energy consumption indicate that it will not take very long time when the sensor motes would be capable of performing asymmetric operations within acceptable time limits. If that happens our proposed scheme can take full advantage of the improved hardware and can be deployed solely using pairing-based scheme (that is without requiring any member nodes) to provide high security. Additionally, with minimum modifications the proposed scheme can be applied to other resource-constrained networks such as mobile ad hoc networks, vehicular ad hoc networks, wireless smart phone networks, etc. As a final thought, a large scale deployment can be carried out of the proposed scheme for critical applications to unveil other deficiencies which cannot be dealt with under simulation.

Bibliography

[1] X. Chen, K. Makki, K. Yen, and N. Pissinou, "Sensor network security: a survey," *Communications Surveys Tutorials, IEEE*, vol. 11, pp. 52 –73, quarter 2009.

[2] J. Lee, V. Leung, K. Wong, J. Cao, and H. Chan, "Key management issues in wireless sensor networks: current proposals and future developments," *Wireless Communications, IEEE*, vol. 14, pp. 76 –84, october 2007.

[3] J. Zhang and V. Varadharajan, "Wireless sensor network key management survey and taxonomy," *Journal of Network and Computer Applications*, vol. 33, no. 2, pp. 63 – 75, 2010.

[4] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *Communications Magazine, IEEE*, vol. 40, pp. 102 – 114, aug 2002.

[5] T. Arampatzis, J. Lygeros, S. Member, and S. Manesis, "A survey of applications of wireless sensors and wireless sensor networks," in *Proc. 13 th Mediterranean Conference on Control and Automation, Limassol*, pp. 719–724, 2005.

[6] Z. Li and G. Gong, "A survey on security in wireless sensor networks," Tech. Rep. CACR 2008-20, University of Waterloo, 2008.

[7] Y.-M. Huang, M.-Y. Hsieh, and F. Sandnes, "Wireless sensor networks and applications," in *Sensors* (S. Mukhopadhyay and R. Huang, eds.), vol. 21 of *Lecture Notes Electrical Engineering*, pp. 199–219, Springer Berlin Heidelberg, 2008.

[8] T. Bokareva, W. Hu, S. Kanhere, B. Ristic, N. Gordon, T. Bessell, M. Rutten, and S. Jha, "Wireless Sensor Networks for Battlefield Surveillance," *roceedings of The Land Warfare Conference (LWC)*, 2006.

[9] D. Li, K. Wong, Y. H. Hu, and A. Sayeed, "Detection, classification, and tracking of targets," *Signal Processing Magazine, IEEE*, vol. 19, pp. 17 –29, mar 2002.

[10] C. Meesookho, S. Narayanan, and C. Raghavendra, "Collaborative classification applications in sensor networks," in *Sensor Array and Multichannel Signal Processing Workshop Proceedings, 2002*, pp. 370 – 374, aug. 2002.

[11] B. Sinopoli, C. Sharp, L. Schenato, S. Schaffert, and S. Sastry, "Distributed control applications within sensor networks," *Proceedings of the IEEE*, vol. 91, pp. 1235 – 1246, aug. 2003.

[12] J. Byun, I. Hong, B. Kang, and S. Park, "A smart energy distribution and management system for renewable energy distribution and context-aware services based on user patterns and load forecasting," *Consumer Electronics, IEEE Transactions on*, vol. 57, pp. 436 –444, may 2011.

[13] A. D. S. B. Clemente, J. R. Martnez-de Dios, and A. O. Baturone, "A wsn-based tool for urban and industrial fire-fighting," *Sensors*, vol. 12, no. 11, pp. 15009–15035, 2012.

[14] A. Rytter, "Vibration based inspection of civil engineering structures," in *Ph.d disertation, department of building technology and structural enginnering, Aalborg University*, 1993.

[15] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, WSNA '02, (New York, NY, USA), pp. 88–97, ACM, 2002.

[16] E. S. Biagioni and K. W. Bridges, "The application of remote sensor technology to assist the recovery of rare and endangered species," *International Journal of High Performance Computing Applications*, vol. 16, p. 2002, 2002.

[17] J. Burrell, T. Brooke, and R. Beckwith, "Vineyard computing: Sensor networks in agricultural production," *IEEE Pervasive Computing*, vol. 3, pp. 38–45, Jan. 2004.

[18] T. Brooke and J. Burrell, "From ethnography to design in a vineyard, in," in *Proceedings of the Conference on Designing for User Experiences*, 2003.

[19] H. Mareca, G. Darryl, and C. Charlie, "Industrial wsn global 2012 survey." `http://onworld.com/smartindustries/2012survey/index.html`, Feb 2013. [Online; accessed 20-February-2013].

[20] T. Knot, "Smart surrogates," *BP Frontiers Magazine*, pp. 6–10, April 2004.

[21] M. Akhondi, A. Talevski, S. Carlsen, and S. Petersen, "Applications of wireless sensor networks in the oil, gas and resources industries," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pp. 941 –948, april 2010.

[22] CITRIS, "Smart energy distribution and consumption." `http://www.intel.com/content/www/us/en/research/intel-research.html`, 2013. [Online; accessed 20-February-2013].

[23] G. Yang, *Body Sensor Networks*. Springer, 2011.

[24] I. Research, "Proactive health research initiative." `http://www.intel.com/content/www/us/en/research/intel-research.html`, 2013. [Online; accessed 20-February-2013].

[25] T. Gao, C. Pesto, L. Selavo, Y. Chen, J. G. Ko, J. H. Lim, A. Terzis, A. Watt, J. Jeng, B. rong Chen, K. Lorincz, and M. Welsh, "Wireless medical sensor networks in emergency response: Implementation and pilot results," in *Technologies for Homeland Security, 2008 IEEE Conference on*, pp. 187 –192, may 2008.

[26] D. Malan, T. Fulford-jones, M. Welsh, and S. Moulton, "Codeblue: An ad hoc sensor network infrastructure for emergency medical care," in *In International Workshop on Wearable and Implantable Body Sensor Networks*, 2004.

[27] G. Super, S. Groth, and R. Hook, "START: Simple triage and rapid treatment plan," *Hoag Memorial Hospital Presbyterian*, 1994.

[28] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensor networks," *COMMUNICATIONS OF THE ACM*, vol. 47, no. 6, pp. 53–57, 2004.

[29] G. Padmavathi and D. Shanmugapriya, "A survey of attacks, security mechanisms and challenges in wireless sensor networks," *CoRR*, vol. abs/0909.0576, 2009.

[30] A. Pathan, H.-W. Lee, and C. S. Hong, "Security in wireless sensor networks: issues and challenges," in *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*, vol. 2, pp. 6 pp. –1048, feb. 2006.

[31] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures," in *Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on*, pp. 113 – 127, may 2003.

[32] X. Chen, K. Makki, K. Yen, and N. Pissinou, "Sensor network security: a survey," *Communications Surveys Tutorials, IEEE*, vol. 11, pp. 52 –73, quarter 2009.

[33] J. Sen, "A survey on wireless sensor network security," *International Journal of Vommunication Networks and Information Security (IJCNIS)*, vol. 52, no. 12, p. 24, 2010.

[34] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, Inc., 1st ed., 1996.

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

[36] G. Padmavathi, Dr. and M. D. Shanmugapriya, "A Survey of Attacks, Security Mechanisms and Challenges in Wireless Sensor Networks," *ArXiv e-prints*, Sept. 2009.

[37] T. Bonaci, L. Bushnell, and R. Poovendran, "Node capture attacks in wireless sensor networks: A system theoretic approach," in *CDC*, pp. 6765–6772, 2010.

[38] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, (New York, NY, USA), pp. 41–47, ACM, 2002.

[39] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pp. 197 – 213, 11-14 2003.

[40] R. Blom, "An optimal class of symmetric key generation systems," in *Proc. of the EUROCRYPT 84 workshop on Advances in cryptology: theory and application of cryptographic techniques*, (New York, NY, USA), pp. 335–338, Springer-Verlag New York, Inc., 1985.

[41] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-secure key distribution for dynamic conferences," in *CRYPTO '92: Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, (London, UK), pp. 471–486, Springer-Verlag, 1993.

[42] T.-H. Shan and C.-M. Liu, "Enhancing the key pre-distribution scheme on wireless sensor networks," in *Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference*, (Washington, DC, USA), pp. 1127–1131, IEEE Computer Society, 2008.

[43] W. Du, R. Wang, and P. Ning, "An efficient scheme for authenticating public keys in sensor networks," in *In 6th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc 05*, pp. 58–67, ACM Press, 2005.

[44] X. Du, Y. Xiao, M. Guizani, and H.-H. Chen, "An effective key management scheme for heterogeneous sensor networks," *Ad Hoc Networks*, vol. 5, no. 1, pp. 24 – 34, 2007.

[45] B. Lai, S. Kim, and I. Verbauwhede, "Scalable session key construction protocol for wireless sensor networks," in *In IEEE Workshop on Large Scale RealTime and Embedded Systems (LARTES*, p. 7, 2002.

[46] H. Chan and A. Perrig, "PIKE: peer intermediaries for key establishment in sensor networks," in *INFOCOM*, pp. 524–535, 2005.

[47] L. B. Oliveira, D. F. Aranha, C. P. L. Gouvêa, M. Scott, D. F. Câmara, J. López, and R. Dahab, "TinyPBC: Pairings for Authenticated Identity-Based Non-Interactive Key Distribution in Sensor Networks," *Computer Communications*, vol. 34, no. 3, pp. 485–493, 2011.

[48] P. Szczechowiak and M. Collier, "TinyIBE: Identity-based encryption for heterogeneous sensor networks," in *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2009 5th International Conference on*, pp. 319 –354, dec. 2009.

[49] Wikipedia, "Key management — wikipedia, the free encyclopedia." `http://en.wikipedia.org/w/index.php?title=Key_management&oldid=527556690`, 2012. [Online; accessed 22-January-2013].

[50] Z. Yu and Y. Guan, "A key management scheme using deployment knowledge for wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, pp. 1411–1425, Oct. 2008.

[51] R. M. S. Silva, N. S. A. Pereira, and M. S. Nunes, "Probabilistic key management practical concerns in wireless sensor networks," *JNW*, vol. 3, no. 2, pp. 29–37, 2008.

[52] J. Spencer, *The Strange Logic of Random Graphs.* Algorithms and Combinatorics Series, Springer, 2010.

[53] M. Rahman, S. Sampalli, and S. Hussain, "A robust pair-wise and group key management protocol for wireless sensor network," in *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pp. 1528 –1532, dec. 2010.

[54] H. Y. Chien, R.-C. Chen, and A. Shen, "Efficient key pre-distribution for sensor nodes with strong connectivity and low storage space," in *Advanced Information Networking and Applications, 2008. AINA 2008. 22nd International Conference on*, pp. 327 –333, march 2008.

[55] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," in *Proceedings of the 10th ACM conference on Computer and communications security*, CCS '03, (New York, NY, USA), pp. 42–51, ACM, 2003.

[56] D. Liu, P. Ning, and R. Li, "Establishing pairwise keys in distributed sensor networks," *ACM Trans. Inf. Syst. Secur.*, vol. 8, no. 1, pp. 41–77, 2005.

[57] D. Liu and P. Ning, "Location-based pairwise key establishments for static sensor networks," in *In 2003 ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN 03*, pp. 72–82, ACM Press, 2003.

[58] N. T. Canh, Y.-K. Lee, and S. Lee, "HGKM: A group-based key management scheme for sensor networks using deployment knowledge," in *Communication Networks and Services Research Conference, 2008. CNSR 2008. 6th Annual*, pp. 544 –551, may 2008.

[59] D. Malan, M. Welsh, and M. Smith, "A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography," in *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, pp. 71 – 80, oct. 2004.

[60] R. Watro, D. Kong, S. fen Cuti, C. Gardiner, C. Lynn, and P. Kruus, "Tinypk: securing sensor networks with public key technology," in *In SASN 04: Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks*, pp. 59–64, ACM Press, 2004.

[61] K. Ren, W. Lou, and Y. Zhang, "Multi-user broadcast authentication in wireless sensor networks," in *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON '07. 4th Annual IEEE Communications Society Conference on*, pp. 223 –232, june 2007.

[62] M. A. Simplício, Jr., P. S. L. M. Barreto, C. B. Margi, and T. C. M. B. Carvalho, "A survey on key management mechanisms for distributed wireless sensor networks," *Comput. Netw.*, vol. 54, pp. 2591–2612, Oct. 2010.

[63] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proceedings of CRYPTO 84 on Advances in cryptology*, (New York, NY, USA), pp. 47–53, Springer-Verlag New York, Inc., 1985.

[64] R. Sakai, K. Ohgishi, and M. Kasahara, "Cryptosystems based on pairing," in *The 2000 Symposium on Cryptography and Information Security*, 2000.

[65] A. Joux, "A one round protocol for tripartite diffie-hellman," *Journal of Cryptology*, vol. 17, pp. 263–276, 2004. 10.1007/s00145-004-0312-y.

[66] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," *SIAM J. of Computing*, vol. 32, no. 3, pp. 586–615, 2003. extended abstract in Crypto'01.

[67] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," in *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '01, (London, UK), pp. 514–532, Springer-Verlag, 2001.

[68] C.-K. Chu, J. K. Liu, J. Zhou, F. Bao, and R. H. Deng, "Practical id-based encryption for wireless sensor network," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, (New York, NY, USA), pp. 337–340, ACM, 2010.

[69] L. B. Oliveira, R. Dahab, J. Lopez, F. Daguano, and A. A. Loureiro, "Identity-based encryption for sensor networks," *Pervasive Computing and Communications Workshops, IEEE International Conference on*, vol. 0, pp. 290–294, 2007.

[70] Y. Yussoff and H. Hashim, "IBE-Trust: A security framework for wireless sensor networks," in *Internet Security (WorldCIS), 2011 World Congress on*, pp. 171 –176, feb. 2011.

[71] G. YANG, C. ming RONG, C. VEIGNER, J. tao WANG, and H. bing CHENG, "Identity-based key agreement and encryption for wireless sensor networks," *The Journal of China Universities of Posts and Telecommunications*, vol. 13, no. 4, pp. 54 – 60, 2006.

[72] L. Martin, *Introduction to Identity-Based Encryption (Information Security and Privacy Series)*. Norwood, MA, USA: Artech House, Inc., 1 ed., 2008.

[73] Wikipedia, "Vandermonde matrix — wikipedia, the free encyclopedia," 2013. [Online; accessed 31-January-2013].

[74] X. Liu, "A survey on clustering routing protocols in wireless sensor networks," *Sensors*, vol. 12, no. 8, pp. 11113–11153, 2012.

[75] Z. J. Haas, J. Y. Halpern, and L. Li, "Gossip-based ad hoc routing," *IEEE/ACM Trans. Netw.*, vol. 14, pp. 479–491, June 2006.

[76] J. Kulik, W. Heinzelman, and H. Balakrishnan, "Negotiation-based protocols for disseminating information in wireless sensor networks," *Wirel. Netw.*, vol. 8, pp. 169–185, Mar. 2002.

[77] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. Netw.*, vol. 11, pp. 2–16, Feb. 2003.

[78] A. Forster and A. Murphy, "Clique: Role-free clustering with q-learning for wireless sensor networks," in *Distributed Computing Systems, 2009. ICDCS '09. 29th IEEE International Conference on*, pp. 441 –449, june 2009.

[79] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocols for wireless microsensor networks," *Hawaaian Int'l Conference on Systems Science*, 2000.

[80] K. Hwang, C.-S. Wu, B.-C. Cheng, C.-C. Cheng, J.-B. Lin, and H. Chen, "An improved leach-based power aware routing protocol in wireless sensor networks," in *Communications and Networking in China, 2008. ChinaCom 2008. Third International Conference on*, pp. 726 –731, aug. 2008.

[81] Z. Liu, Z. Liu, and L. Wen, "A modified leach protocol for wireless sensor networks," in *Advanced Computational Intelligence (IWACI), 2011 Fourth International Workshop on*, pp. 766 –769, oct. 2011.

[82] L. Buttyán and P. Schaffer, "Position-based aggregator node election in wireless sensor networks," *IJDSN*, vol. 2010, 2010.

[83] P. Ding, J. Holliday, and A. Celik, "Distributed energy-efficient hierarchical clustering for wireless sensor networks," in *Proceedings of the First IEEE international conference on Distributed Computing in Sensor Systems*, DCOSS'05, (Berlin, Heidelberg), pp. 322–339, Springer-Verlag, 2005.

[84] M. Ye, C. Li, G. Chen, J. Wu, and M. Y. E. Al, "Eecs: An energy efficient clustering scheme in wireless sensor networks," in *In: Proc. of the IEEE Intl Performance Computing and Communications Conf*, pp. 535–540, IEEE Press, 2005.

[85] L. Krishnamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," in *Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on*, pp. 575 – 578, 2002.

[86] L. Wang, J. Liu, and W. Wang, "An improvement and simulation of leach protocol for wireless sensor network," in *Pervasive Computing Signal Processing and Applications (PCSPA), 2010 First International Conference on*, pp. 444 – 447, sept. 2010.

[87] J. Xu, N. Jin, X. Lou, T. Peng, Q. Zhou, and Y. Chen, "Improvement of leach protocol for wsn," in *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, pp. 2174 –2177, may 2012.

[88] O. Younis and S. Fahmy, "Heed: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, pp. 366–379, 2004.

[89] G. Chen, J. W. Branch, and B. K. Szymanski, "Local leader election, signal strength aware flooding, and routeless routing," in *In 5th IEEE Intern. Workshop Algorithms for Wireless, Mobile, Ad Hoc Networks and Sensor Networks*, pp. 4–8, IEEE CS Press, 2005.

[90] C. Fetzer and F. Cristian, "A highly available local leader election service," *IEEE Transactions on Software Engineering*, vol. 25, pp. 603–618, 1999.

[91] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes (North-Holland Mathematical Library)*. North Holland, June 1988.

[92] D. Crossbow, "Micaz — crossbow datasheet," January 2013.

[93] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "Establishing pairwise keys for secure communication in ad hoc networks: A probabilistic approach," in *Proceedings of the 11th IEEE International Conference on Network Protocols*, ICNP '03, (Washington, DC, USA), pp. 326–, IEEE Computer Society, 2003.

[94] M. Boujelben, H. Youssef, and M. Abid, "An efficient scheme for key predistribution in wireless sensor networks," *Wireless and Mobile Computing, Networking and Communication, IEEE International Conference on*, vol. 0, pp. 532–537, 2008.

[95] W. S, O. Rosemary, W. Ninging, H. Michael, O. Brendan, and M. S. Cian, "Practical wireless sensor networks power consumption metrics for building energy management applications," *23rd European Conference Forum Bauinformatik 2011*, 2011.

[96] P. Levis, S. Madden, J. Polastre, R. Szewczyk, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "Tinyos: An operating system for sensor networks," in *in Ambient Intelligence*, Springer Verlag, 2004.

[97] D. Gay, M. Welsh, P. Levis, E. Brewer, R. V. Behren, and D. Culler, "The nesc language: A holistic approach to networked embedded systems," in *In Proceedings of Programming Language Design and Implementation (PLDI*, pp. 1–11, 2003.

[98] Mica2 data sheet, "Crossbow Technology Inc.." `http://www.xbow.com/`. Accessed April 20, 2012.

[99] W. Page, "Intel mote 2," December 2012.

[100] TelosB data sheet, "Crossbow Technology Inc.." `http://www.xbow.com/`. Accessed April 20, 2012.

[101] P. S. L. M. Barreto, S. Galbraith, C. O. Heigeartaigh, and M. Scott, "Efficient pairing computation on supersingular abelian varieties," in *Designs, Codes and Cryptography*, pp. 239–271, 2004.

[102] X. Du, Y. Xiao, S. Ci, M. Guizani, and H.-H. Chen, "A routing-driven key management scheme for heterogeneous sensor networks," in *Communications, 2007. ICC '07. IEEE International Conference on*, pp. 3407 –3412, june 2007.

[103] V. Shnayder, M. Hempstead, B. rong Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *In Sensys*, pp. 188–200, ACM Press, 2004.

Appendix A

Raw Data from Experimental Results

The Table A.1 shows the results of 20 runs from the experiment. For each run, the computational time for evaluating λ-degree polynomial on a TelosB sensor mote is shown. The value of the λ is varied from 16 to 128, doubling its value every time.

Table A.1: Computation time for key agreement using matrix-based polynomial evaluation on TelosB mote

Run#	$\lambda = 16$	$\lambda = 32$	$\lambda = 64$	$\lambda = 128$
				Running Time in Milliseconds
1	7.903	15.901	34.972	73.917
2	7.507	16.423	33.949	74.197
3	8.052	16.894	35.782	74.107
4	7.523	16.762	34.827	74.11
5	7.833	17.202	34.692	73.917
6	8.012	16.524	34.902	74.119
7	7.021	16.782	34.881	74.211
8	7.611	17.132	34.91	73.817
9	8.23	16.921	35.851	74.271
10	8.113	16.238	34.751	73.915
11	7.239	16.921	35.019	74.198
12	8.124	16.412	35.101	74.17
13	7.824	16.782	34.718	74.224
14	7.123	17.101	34.821	74.187
15	8.34	16.921	34.902	73.991
16	8.702	16.8012	34.912	73.819
17	7.902	17.11	34.981	74.217
18	8.231	16.928	34.891	74.082
19	8.034	16.889	34.987	74.115
20	8.201	17.018	34.864	73.889
Avg. Time (ms):	**7.87625**	**16.78311**	**34.93565**	**74.07365**

In Table A.2, we show the raw computational time of 20 runs from the experimental results. In each run, a common key between two nodes is computed on a TelosB mote using pairing-based calculations. We have used the TinyPBC [47] library for computing the common key.

Table A.2: Computation time for key agreement using pairing-based operation on TelosB mote

Run#	**Running Time (sec)**
1	2.421
2	1.982
3	2.2012
4	1.921
5	2.312
6	2.2589
7	2.342
8	2.012
9	2.123
10	2.298
11	2.134
12	2.201
13	2.312
14	2.19
15	2.312
16	2.223
17	2.231
18	2.121
19	2.2012
20	2.314
Avg. Time (sec):	**2.205515**

Appendix B

Publications/Patents from the Thesis

List of patent:

1. Musfiq Rahman, "A Robust Pair-wise and Group Key Management Protocol for Wireless Sensor Network", *US Provisional Patent (Number: 61354867)* Filed on June 15, 2010.

List of publications:

1. Musfiq Rahman and Srinivas Sampalli, "Scalable Key Management Protocol for Wireless Sensor Networks", *manuscript under review, Wireless Personal Communications (Springer), Special Issue on Advances in Trust, Security and Privacy for Wireless Communication Networks*, submitted January 2013.

2. Musfiq Rahman and Srinivas Sampalli, "A Pair-wise and Group-wise Key Management Protocol for Wireless Sensor Network", *manuscript under review, Journal of Communications Software and Systems (JCOMSS) Special issue on RFID Technologies and Internet of Things*, submitted November 2012.

3. Ambica Pawan Khandavilli, Musfiq Rahman and Srinivas Sampalli, "A Mobile Role-Based Access Control System using Identity-Based Encryption with Zero Knowledge Proof", *in Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defence Applications (CISDA 2012)*, July 2012, Ottawa, Canada.

4. Musfiq Rahman and Srinivas Sampalli,"A Hybrid Key Management Protocol for Wireless Sensor Networks", *in Proceedings of the* 11^{th} *IEEE Conference on Trust, Security, and Privacy in Computing and Communications (TrustCom 2012)*, June 2012, Liverpool, UK.

5. Musfiq Rahman, Srinivas Sampalli and Sajid Hussain,"A Robust Pair-wise and Group Key Management Protocol for Wireless Sensor Network", *in Proceedings of the IEEE Globecom 2010 Workshopon Web and Pervasive Security (WPS 2010)*, December, 2010, Florida, USA.