TWO-TIER, LOCATION-AWARE AND HIGHLY RESILIENT KEY PREDISTRIBUTION SCHEME FOR WIRELESS SENSOR NETWORKS

by

ABDÜLHAKİM ÜNLÜ

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
August 2006

TWO-TIER, LOCATION-AWARE AND HIGHLY RESILIENT KEY PREDISTRIBUTION SCHEME FOR WIRELESS SENSOR NETWORKS

APPROVED BY:

Asst. Prof. Albert Levi …….
(Thesis Supervisor)

Asst. Prof. Selim Balcısoy …….

Asst. Prof. Özgür Erçetin …….

Asst. Prof. Özgür Gürbüz …….

Asst. Prof. Erkay Savaş …….

DATE OF APPROVAL: …….

TWO-TIER, LOCATION-AWARE AND HIGHLY RESILIENT KEY PREDISTRIBUTION SCHEME FOR WIRELESS SENSOR NETWORKS

Abdülhakim Ünlü

Computer Science and Engineering, MS Thesis, 2006

Thesis Supervisor: Asst. Prof. Albert Levi

Keywords: Sensor Networks, Key Predistribution

Abstract

Sensor nodes are low power, tiny, and computationally restricted microelectromechanical devices that usually run on battery. They are capable of communicating over short distances and of sensing information for specific purposes. In sensor networks, large amount of sensor nodes are deployed over a wide region. For secure communication among sensor nodes, secure links must be established via key agreement. Due to resource constraints, achieving such key agreement in wireless sensor networks is non-trivial. Many key establishment schemes, like Diffie-Hellman and public-key cryptography based protocols, proposed for general networks are not so suitable for sensor networks due to resource constraints. Since one cannot generally assume a trusted infrastructure, keys and/or keying materials must be distributed to sensor nodes before deployment of them. Such key distribution schemes are called *key predistribution* schemes. After deployment, sensor nodes use predistributed keys and/or keying materials to establish secure links using various techniques.

In this thesis, we propose a probabilistic key predistribution scheme, in which we assume that certain deployment knowledge is available prior to deployment of sensor nodes. We use a two-tier approach in which there are two types of nodes: regular nodes and agent nodes. Agent nodes, which constitute a small percentage of all nodes, are more

capable than regular nodes. Most of the regular nodes can establish shared keys among themselves without the help of agent nodes, whereas some other regular nodes make use of agent nodes as intermediaries for key establishment. We give a comparative analysis of our scheme through simulations and show that our scheme provides good connectivity for the sensor network. Moreover, our scheme exhibits substantially strong node-capture resiliency against small-scale attacks, while the resiliency of the network degrades gracefully as the number of captured nodes increases. In addition, the proposed scheme is scalable such that increasing the number of nodes in the network does not degrade the performance and does not increase the complexity. Another good characteristic of our scheme is that it is resistant against node fabrication and partially resistant against wormhole attacks.

DUYARGA AĞLARI İÇİN İKİ SEVİYELİ, KONUM BİLGİSİ KULLANAN VE YÜKSEK DAYANIKLILIĞA SAHİP ÖN YÜKLEMELİ ANAHTAR DAĞITIM MEKANİZMASI

Abdülhakim Ünlü

Bilgisayar Bilimi ve Mühendisliği, Yüksek Lisans Tezi, 2006

Tez Danışmanı: Yrd. Doç. Dr. Albert Levi

Anahtar Kelimeler: Duyarga Ağları, Anahtar Dağıtımı

ÖZET

Duyarga düğümleri kısa mesafelerde iletişim kurma yetisine sahip, belirli amaçlara yönelik bilgi toplayabilen elektromekanik cihazlardır. Duyarga düğümleri genellikle küçük, düşük enerji tüketen, pil gücü zayıf ve kısıtlı hesaplama yapmaya uygun bir yapıya sahiptirler. Bir duyarga ağında, geniş bir alana dağılmış çok miktarda duyarga cihazı vardır. Herhangi iki duyarga düğümü arasında güvenli bir iletişim için, güvenli ve şifrelenmiş bir hat oluşturmak gerekir. Güvenli bir hat oluşturmak için gerekli olan ortak anahtar türetmek işi, duyarga düğümlerinin kısıtlı kaynaklara sahip olmaları yüzünden basit bir şekilde yapılamaz. Genel anlamda ağlar için önerilen açık anahtarlı şifreleme yöntemi, kısıtlı kaynakları sebebiyle duyarga ağları için uygun değildir. Ayrıca, duyarga ağları güvenilir bir altyapıya sahip olmadıkları için, anahtarların ve diğer güvenlik bilgilerinin duyarga düğümlerine konuşlandırma öncesi yüklenmesi gereklidir. Bu tip şemalara ön-yüklemeli anahtar dağıtım şemaları denir. Konuşlandırma sonrası duyarga düğümleri, önceden yüklenmiş olan anahtarları ve diğer güvenlik bilgilerini değişik metotlarda kullanarak güvenli hat oluştururlar.

Bu tezde, rastlantısal ön yüklemeli bir anahtar dağıtım mekanizması önerilmektedir. Önerilen yöntemde, duyarga cihazlarının konuşlandırma sonrası konumlarına ait bazı bilgilere kısmen sahip olunabileceği kabul edilmektedir. Kullanılan şemada, duyarga düğümleri arasında iki sıralı bir yapı mevcuttur. Duyarga ağını iki tip düğüm oluşturur: sıradan ve aracı düğümler. Aracı duyarga düğümleri duyarga ağının az bir kısmını oluşturur ve sıradan duyarga düğümlerine göre daha gelişmiş özelliklere sahiptir. Önerilen yöntemin performans analizi simülasyonlar ile yapılmıştır ve analiz sonuçları göstermektedir ki, önerilen anahtar dağıtım yöntemi yüksek bağlanabilirlik özelliğine sahiptir. Ayrıca, önerilen anahtar dağıtım yöntemi ufak çaplı saldırılara karşı güçlü dayanıklılığa sahiptir. Tezde önerilen yöntemin bir başka özelliği de kolay bir şekilde ölçeklenebilir olmasıdır. Bununla birlikte, önerilen yöntem duyarga düğümü kopyalanması ve wormhole saldırılarına karşı dayanıklıdır.

To my family

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Albert Levi for his guidance and especially for his patience during this work.

Special thanks are due to Dr. Selim Balcısoy, Dr. Özgür Erçetin, Dr. Özgür Gürbüz, and Dr. Erkay Savaş for their kindness to join my jury.

Also, many thanks to Dr. Erkay Savaş and Dr. Özgür Erçetin for their support and valuable comments during CS 680.

I must thank The Scientific And Technological Research Council Of Turkey (TÜBİTAK) for funding this research under Grant 104E071.

I specially thank to my family for supporting me with every decision I make. Also, sincere thanks to Ali İnan, Selim Volkan Kaya, Önsel Armağan, Sinan Emre Taşcı and other friends at the office.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Sensor nodes are small and battery powered devices with limited computational power, memory capacity and radio range. They are low-cost devices and large number of sensor nodes can be deployed over a target area to form a sensor network. Sensor networks can be utilized for environment monitoring, investigation of hazardous environments, health services or military services as detailed in [3].

Confidentiality, privacy and authenticity of communication among the sensor nodes should be provided, whether communication is for transfer of sensed data or some other operational messages, when nodes are deployed in an environment where there are adversaries. In order to fulfill these security requirements, cryptographic techniques are employed. Although there has been some recent studies to use public key cryptography (PKC) in sensor networks [4, 5, 6], it is still not so practical to use PKC in all sensor nodes. Thus, symmetric cryptography is used to provide security in sensor networks. In order to use symmetric key cryptography, communicating sensor nodes must share the same cryptographic key. The problem of distribution of keys to large number of sensor nodes is an active research area. Key predistribution schemes [1, 7-12], where they keys are stored in sensor nodes before the deployment, are proven to be practical and efficient solutions.

A naïve way of key predistribution is to generate a master key and install this master key to all nodes before the deployment. After deployment, all the sensor nodes can encrypt, decrypt and authenticate their communication with this master key. However, in this scheme, when a node is captured, the master key is also captured and all secure links in the sensor network are compromised.

One possible way to protect keys inside a sensor node is to tamper-proof the device. However, this approach increases the cost of sensor nodes [49]. Furthermore, tamper-proofing may not be always safe as discussed in [13]. In this thesis, we assume that sensor

devices are not tamper-resistant, so when a node is captured, all the cryptographic information in the node can be seized by the attacker.

Another way of key predistribution is to assign unique link keys for each node. In this method, compromise of one node leads to compromise of only that node's links. However, this method is not scalable since the total number keys to be predistributed per node should be as much as the number of nodes in the network in order to guarantee that after deployment each neighboring node pair has a common key. As the size of the network grows, it would be hard to realize this scheme since sensor nodes have only limited amount of memory.

In order to overcome this scalability problem and effectively use the node's memory, Eschenauer and Gligor proposed a probabilistic key predistribution scheme [1]. In this scheme, before sensor deployment, a key server creates a key ring for each node, by picking a limited number of random keys from a large key pool. Then the key server loads the key ring to memory of each node. After deployment, sensor nodes let their neighbors know which keys they have. If two neighboring nodes share one or more keys, then they can establish a secure link by using the shared key. After this shared key discovery with direct neighbors, neighboring node pairs that do not share keys can establish secure links in multiple hops. If the local connectivity (in terms of secure links) is above a certain threshold, then random graph theory [14] states that overall sensor network will be cryptographically connected with high probability.

Du et al. [9] utilized Blom's key management scheme [2] in a key predistribution scheme for sensor networks. Du et al.'s scheme shows a threshold property; until λ nodes are captured, the network is perfectly secure, but if $\lambda+1$ or more nodes are captured all secure links are compromised.

Some recent papers on random key predistribution [8, 11, 15, 16, 17] utilized expected location information of sensor nodes in their sensor node deployment models. In all these location-aware approaches, it is assumed that nodes are prepared in small groups

and deployed as bundles, e.g. groups of nodes can be dropped from a plane, similar to parachuting troops or dropping cargo. The nodes in the same group have a large chance of being in the radio communication range of each other. Similarly, the node groups that are dropped next to each other also have a chance to be close to each other on the ground. Using this deployment location knowledge, key pools and key rings are arranged and analysis show that performance of key predistribution schemes is improved substantially. In location aware schemes, the node deployment model is one of the most important design criteria that directly affect the performance of the scheme. There is still room to further improve the performance, in terms of connectivity, resiliency and memory usage, of key distribution schemes with better deployment models and key distribution methods.

1.1 Contribution of the Thesis

We propose a scalable, two-tier approach for key predistribution problem in sensor networks, where there are two types of sensor nodes with different capabilities: regular nodes and agent nodes. Keys are predistributed according to nodes' capabilities, such that more keys and keying material are stored in agent nodes that constitute a small part of the network and are more capable than regular nodes. For example, in a military setting, many simple sensor nodes may be deployed in a field of operation along with a small set of more powerful, more secure nodes, perhaps in attended vehicles.

In our scheme, we make use of deployment knowledge. Our node deployment method is based on the observation that if a group of sensor nodes is deployed at a deployment point, they will likely reside in close proximity with each other. Using such deployment knowledge, we can predict probable set of neighbors of a node. We divide the deployment area into zones and create separate key spaces for each zone. Bundles of sensor nodes are prepared and keys and/or keying material from corresponding key spaces are distributed to nodes. Then, bundles of nodes are deployed at different zones. By employing deployment knowledge in our key predistribution scheme, we achieve efficient memory usage and high connectivity.

The proposed key predistribution scheme is highly scalable, such that our scheme works well in networks of all sizes. Since we use a zone-based approach and key spaces for each zone are separate, addition of new zones does not increase the memory and communication costs of other nodes. Agent nodes are given keys from their zones' key space(s) and they share keys with other agent nodes from neighboring zones. On the other hand, regular nodes are given keys such that they can establish secure links only with same-zone neighbors without intervention of agent nodes. Another benefit of this approach is that our scheme provides partial resistance against wormhole attacks.

Our key predistribution scheme has node-to-node authentication property. In our scheme, cryptographic keys and IDs of nodes that the keys are stored are linked, so that nodes can verify the identity of each other. Authentication property prevents attacks like node fabrication and malicious node insertion, which will be described in detail in the following sections.

The rest of this thesis is organized as follows: in Section 2, we provide background information on sensor networks and sensor devices, and we briefly describe some of the previous work done on key predistribution for sensor networks. In Section 3, we describe our two-tier location-aware key predistribution scheme. In Section 4, we provide a comparative analysis of our scheme. We give evaluation of our scheme in terms of connectivity, resiliency and communication cost. In addition, we provide comparisons of our scheme with existing key predistribution schemes. Finally, we provide some concluding remarks in Section 5.

2 BACKGROUND AND RELATED WORK

We can define a sensor node as a low power, tiny and computationally restricted microelectromechanical device that usually runs on battery and is capable of communicating over short distances and sensing information for specific purposes. A sensor node typically contains a power unit, a sensing unit, a processing unit, a storage unit, and a wireless transmitter / receiver. A wireless sensor network (WSN) is composed of large number of sensor nodes with limited power, computation, storage and communication capabilities [22]. Commercially available sensors, such as the Berkeley MICA2 mote [38] and μAMPS wireless sensor node [39], are characterized by their limited processing capability, tiny memory, and small size.

SmartDust [25] and WINS [37] are some of practical sensor network projects. An important feature of a sensor network is its cooperative effort [23]. Sensor network protocols and algorithms must possess self-organizing capabilities and exhibit cooperative higher-level behavior. Sensor devices have on-board processors, which can carry out simple computations. Instead of sending raw sensed data, sensor devices can do some local computation and partially process raw data.

Sensor nodes are deployed near the phenomenon or event that we need data about [21]. The environments that nodes are deployed can be controlled or uncontrolled places. Controlled environments are usually places with limited size, such as home, office, warehouse, factory, etc. Uncontrolled environments are usually dangerous and/or vast places, such as battleground, disaster area, toxic regions, forests, etc. Deployment of nodes can be achieved by hand in controlled areas but as the number of nodes grows, it becomes highly impractical. If the target environment is an uncontrolled one, then nodes have to be deployed by scattering, using either aerial or ground vehicles.

2.1 Sensor Network Applications

Sensor devices have on-board sensing circuits for various purposes. Sensor devices are capable of monitoring wide variety of conditions [24]:

- temperature,
- humidity,
- vehicular movement,
- lightning condition,
- pressure,
- soil makeup,
- noise levels,
- the presence or absence of certain kinds of objects,
- mechanical stress levels on attached objects, and
- characteristics of an object such as speed, direction, and size.

We can categorize sensor network applications as military, environmental, health, home and other commercial areas [24].

- *Military applications*: Sensors are suitable tools for battlefields and other hostile areas because they can be easily deployed and they are low-cost and disposable. Sensor networks can be used for monitoring the status of friendly forces and availability of ammunition and other equipments. Sensor networks can be deployed at strategic areas in order to watch the activities of opposing forces. Sensor networks can be incorporated into guidance systems of the intelligent ammunition for finer targeting. In addition, sensor networks can be used to determine the battle damage, detect nuclear, biological and chemical attack and make nuclear reconnaissance.

- *Environmental applications*: Sensor networks can be deployed in forests and used to detect fires before they spread uncontrollable. Sensor devices may be equipped with effective power scavenging methods [26], such as solar cells, because the sensors may be left unattended for months and even years. Biocomplexity mapping of the environment can be achieved via sensor networks. Ground level deployment of sensor networks can be especially useful for observing small size biodiversity in an ecosystem [27, 28]. Another environmental application of sensor networks is flood detection [29]. An example of real life deployment of sensor networks for flood detection is the ALERT system [30] deployed in the USA.

- *Health applications:* Sensor networks can be used to monitor human physiological data and this data can be stored for a long period [31, 32]. Real-time health data can help doctors to identify predefined symptoms earlier. In addition, sensor networks can be used in hospitals to track doctors and patients.

- *Home applications:* Smart sensor nodes can be integrated with home appliances and they can interact with each other and other entities via networks, such as the Internet [33]. Sensor devices can help users to control their domestic devices remotely or sensor devices can automate management of home devices.

- *Other commercial applications*: Sensor networks are suitable for commercial applications such as building virtual keyboards; managing inventory; monitoring product quality; constructing smart office spaces; environmental control in office buildings; robot control and guidance in automatic manufacturing environments; interactive toys; interactive museums; factory process control and automation; monitoring disaster area; smart structures with sensor nodes embedded inside; machine diagnosis; transportation; factory instrumentation; local control of actuators; detecting and monitoring car thefts; vehicle tracking and detection; and instrumentation of semiconductor processing chambers, rotating machinery, wind tunnels, and anechoic chambers [34-36].

2.2 Sensor Network Deployment

Deployment and management of large number of unattended and inaccessible sensor nodes is a challenging task. Hundreds to several hundred thousands of nodes are deployed throughout the sensor field. They are deployed within tens of feet of each other [40] and the node densities may be as high as 20 nodes/m^3 [39].

Sensor nodes can be deployed in groups or one by one. Deployment groups can be either bundles or lines of nodes; whichever method is used, the objective is to cover the deployment area with nodes evenly. Sensor nodes can be deployed by [24]

- dropping from a plane,
- delivering in an artillery shell, rocket or missile,
- throwing by a catapult (from a ship board, etc.),
- placing in factory, and
- placing one by one either by a human or a robot.

After deployment, sensor network topology may change due to the following changes in sensor nodes' [24]

- position,
- reachability (due to jamming, noise, moving obstacles, etc.),
- available energy,
- malfunctioning, and
- task details.

2.3 Hardware in Sensor Networks

A sensor device is made of four main parts: a sensing unit, a processing & memory unit, a transceiver unit and a power unit. In addition, a sensor device may be attached with an optional location finding system, a mobilizer or a power generator [24]. A typical sensor node is depicted in Figure 2.1.

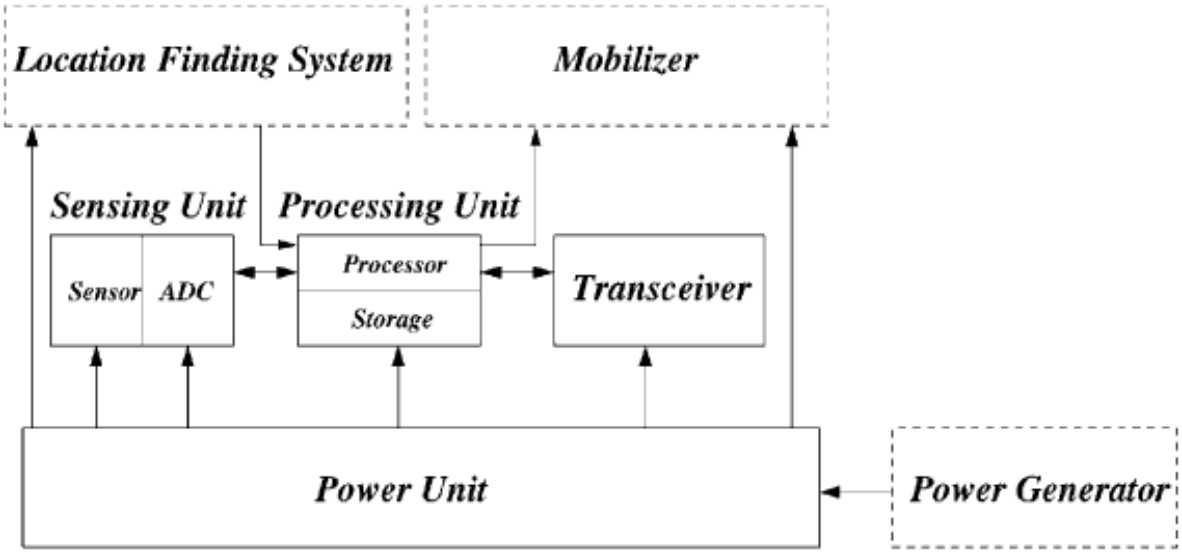

Figure 2-1 A typical sensor device

Smart dust mote and *μAMPS* are two prototype sensor devices. The processing unit of a smart dust mote prototype is a 4 MHz Atmel AVR8535 micro-controller with 8 KB instruction flash memory, 512 bytes RAM and 512 bytes EEPROM [41]. TinyOS operating system runs on this sensor, which has 3500 bytes OS code space and 4500 bytes available code space. μAMPS, another sensor node prototype, utilizes a 59–206 MHz SA-1110 micro-processor. A multithreaded μ-OS operating system runs on μAMPS wireless sensor nodes [39].

It is crucial for sensor nodes to have low power and energy efficient processing units, since sensors have limited energy sources. As pointed in [44], low power is a quality of a device that indicates low energy consumption per clock cycle and energy-efficiency is a quality of a device that indicates low energy consumption per instruction. For example, ATMega128L @ 4MHz consumes 16.5 mW and ARM Thumb @ 40 MHz consumes 75 mW. However, the energy efficiency of ATMega128L @ 4MHz is 242 MIPS/W, spending

4nJ/Instruction and the efficiency of ARM Thumb @ 40 MHz is 480 MIPS/W, spending only 2.1 nJ/Instruction [45]. In Table 2.1, we show energy efficiencies of several microprocessors taken from [45].

Table 2-1 Energy efficiency comparison

Processor Unit	nJ/Instruction
Cygnal C8051F300 @ 32 KHz, 3.3V	0.2 nJ/Instruction
IBM 405LP @ 152 MHz, 1.0V	0.35 nJ/Instruction
Cygnal C8051F300 @ 25MHz, 3.3V	0.5 nJ/Instruction
TMS320VC5510 @ 200 MHz, 1.5V	0.8 nJ/Instruction
Xscale PXA250 @ 400 MHz, 1.3V	1.1 nJ/Instruction
IBM 405LP @ 380 MHz, 1.8V	1.3 nJ/Instruction
Xscale PXA250 @ 130 MHz, .85V	1.9 nJ/Instruction

Sensor nodes have a limited functional life. Because of sensor devices' small size, their power sources are very scarce and because it is not possible to recharge nodes' batteries, when a node runs out of power, it is considered dead. Possible battery types for sensor nodes include NiCd, NiZn, AgZn, NiMh, and Lithium-Ion. They can be further divided as rechargeable and non- rechargeable batteries. It is possible to extend the battery life by using energy scavenging techniques such as solar cells or extracting electrical energy from vibrations [44]. Transceiver components require complex circuitry and consume most of the energy in a duty cycle of a node.

2.4 Communication in Sensor Networks

Communication between sensor nodes is in multi-hop fashion and nodes are linked by a wireless medium. The most preferred communication media for sensor networks is radio, however infrared and optical media are available options. For RF (radio frequency) communication in sensor nodes, small-sized, low-cost, ultra low power transceivers are

required. Choice of carrier frequency is bounded by the trade-off between antenna efficiency and power consumption limit [42].

A suitable carrier frequency band for sensor networks is the industrial, scientific and medical (ISM) band, which offers license-free communication in most countries. Some frequency bands that may be made available for ISM applications are listed in Table 2.2. The main advantage of using ISM bands is that there is no standards enforced for ISM bands and power saving strategies can be freely incorporated into transceivers. However, there are some rules for using unregulated bands, such as power limitations and interference control. As a sample RF design, we can give μAMPS's radio. It uses a Bluetooth-compatible 2.4 GHz transceiver with an integrated frequency synthesizer [39].

A very desirable property of a radio system for sensor devices is the *wake-up property*. A wake-up radio can receive a very simple signal and detect whether a communication with its own node is desired. In this case, it can power up the main radio that will then receive the actual communication. Sensors periodically turn off their components, primarily the transceiver, in order to sustain a long battery life. For example, Berkley's mica mote has to run at 1% duty cycle in order to last for a year [48]. However, turning of the radio means that some nodes may not be able receive critical messages. Keeping the radio in listening state is impractical because radio consumes considerable amount of energy while waiting in receiving mode. For example, Chipcon's CC1000 radio consumes 16.5 mA in transmit mode and 9.6 mA in receive mode [45]. Hence, it is desirable to have an ultra low power communication channel to wake up neighboring nodes on demand. A radio system with wake-up property is the PicoRadio developed by Berkeley Wireless Research Center [46].

Another communication media for sensors is infrared. Infrared communication is license-free and effective against interferences from other electrical appliances. Because infrared communication technology is well developed and used in many electronical devices, infrared-based transceivers are cheap and easy to build. The main drawback of infrared is that it requires a line of sight between sender and receiver. Another alternative

of RF communication is optical systems. In [43], two optical communication systems for sensor networks are discussed. The first one uses corner-cube retroreflector (CCR) and the system does not require an onboard light source. A configuration of three mirrors is used to communicate a digital high or low. The second one uses laser diodes and active-steered mirrors.

Table 2-2 Freely available ISM frequency bands

Frequency Bands	Center Frequency
6765–6795 kHz	6780 kHz
13,553–13,567 kHz	13,560 kHz
26,957–27,283 kHz	27,120 kHz
40.66–40.70 MHz	40.68 MHz
433.05–434.79 MHz	433.92 MHz
902–928 MHz	915 MHz
2400–2500 MHz	2450 MHz
5725–5875 MHz	5800 MHz
24–24.25 GHz	24.125 GHz
61–61.5 GHz	61.25 GHz
122–123 GHz	122.5 GHz
244–246 GHz	245 GHz

2.5 Security Issues in Sensor Networks

When sensor nodes are deployed at hostile environments, security becomes necessary. Adversaries may listen to communication channels, add nodes to the sensor network or physically capture nodes. Hence, sensor networks require security measures like secure communication, intrusion detection, key revocation and node capture detection. However, there are limitations for security in WSN [22]:

1) wireless nature of communication,

2) resource limitation on sensor nodes,
3) very large and dense WSN,
4) lack of fixed infrastructure,
5) unknown network topology prior to deployment,
6) high risk of physical attacks to unattended sensors.

While sensor nodes have limited capabilities, adversaries can have powerful computers and extensive communication devices. They can easily move among sensor nodes using laptops with high capacity batteries. Adversaries have the capability to physically capture, damage or replace sensor nodes. Wireless nature of communication makes it easy for adversaries to eavesdrop on radio messages. Content of radio messages can be classified into four categories: (i) sensor readings, (ii) mobile code, (iii) key management, and (iv) location information [22].

2.5.1 Security Requirements

Wireless and multi-hop nature of communication makes security requirements of sensor networks similar to those of ad-hoc networks [47]. Sensor networks have the following general security requirements [22]:

- *Availability*: ensuring that services offered by the sensor network are available whenever demanded. Denial-of-Service (DOS) attacks can deteriorate the availability of a sensor network. When considering availability in sensor networks, it is important to achieve graceful degradation in the presence of node compromise or node failures.

- *Authentication*: verifying identity of other nodes, base stations or any other type of nodes before granting a limited resource, or sending information. Authentication prevents outsiders from inserting or spoofing messages.

- *Integrity*: ensuring that exchanged messages are not altered or corrupted; receiver gets exactly what the sender sends.

- *Secrecy*: providing privacy of the wireless communication channels to prevent any kind of passive attacks. Any appropriate encryption function and a shared key between communicating parties can be used to achieve secrecy.

- *Non-repudiation*: preventing malicious nodes from hiding their activities.

Sensor networks differ from ad-hoc networks in properties like limited and scarce resources and unattended operation. Thus, sensor networks have the following specific requirements [22]:

- *Survivability:* ability to provide a minimum level of service in the presence of power loss, failures or attacks.

- *Degradation of security services:* ability to change security level as resource availability changes.

2.5.2 Attacks against Sensor Networks

Attacks against sensor networks can be broadly divided into two categories: insider and outsider attacks.

- *Outsider attacks*: An outsider adversary can be defined as unauthorized participant of the sensor network. An outsider adversary can launch passive attacks, like eavesdropping on the network's radio channels, in order to steal sensitive information. In addition, the adversary can alter or spoof messages or inject interfering signals to the network's radio channel in order to jam the network. A form of active attack, which an outsider can perform, is to disable sensor nodes. This can be achieved by sending junk packets to a node and

draining its energy. Furthermore, adversary can physically capture or destroy sensors [49].

- *Insider attacks*: A significant threat to sensor networks is node compromise. Using compromised nodes, an adversary can perform insider attacks. Insider attacks are more critical than outsider attacks because in an insider attack, original nodes of a network are used and authentication techniques fail to detect compromised nodes. The adversary can reprogram and deploy back compromised nodes to disrupt the services of the sensor network or steal private data. In addition, the adversary can use authentic information compromised from a node to fake a more powerful device, like a laptop, as an original member of the sensor network. Using compromised nodes in coordination, an adversary can perform more harmful attacks to the sensor networks, as we will describe in the following section.

Some of the major attacks against sensor networks are as follow [48]:

- Spoofed, altered, or replayed messages
- Selective forwarding
- Sinkhole attacks
- Sybil attacks
- Wormholes
- HELLO flood attacks
- Acknowledgement spoofing

2.5.3 Spoofed, Altered, or Replayed Messages

An outsider adversary can attack routing mechanism of the network by spoofing, altering or replaying routing messages. Without proper countermeasures, the adversary can harm the sensor network easily by creating routing loops, attracting or repelling network

traffic, extending or shortening source routes, generating false error messages, partitioning the network, increasing end-to-end latency, etc. [48].

An outsider attacker can be stopped by employing encryption and authentication on communication. An adversary cannot spoof or alter any messages unless he knows the shared key. However, an insider adversary can maliciously spoof, alter or replay messages since he has the required cryptographic materials.

2.5.4 Selective Forwarding

In sensor networks, there may be one or more sinks and sensors forward sensing data to the sinks hop-by-hop. In such a multi-hop network, it is assumed that sensors faithfully forward the received messages toward the nearest sink. However, in a selective forwarding attack, the adversary drops some of the messages it receives. In order to perform a selective forwarding attack, first the adversary must include itself into a path of data. After doing that, the adversary can simply drop all the packets it receives, but neighbors can easily assume that the malicious node is a failed node and find another route. The adversary can harm the sensor network for a long time by selectively dropping some of the messages and forwarding the rest of the messages. An adversary can include itself to path of data by sinkhole and Sybil attacks, which we will discuss in the next two sections.

2.5.5 Sybil Attack

In [50], the Sybil attack is defined as the action of a malicious device illegitimately taking on multiple identities. We refer to a malicious device's additional identities as Sybil nodes. An adversary can effectively disturb and harm fault-tolerant schemes such as distributed storage [51], dispersity [52] and multipath [53] routing and data aggregation, using the Sybil attack.

A Sybil node can get new identities by either fabricating random identities or stealing identities of existing legitimate nodes. If the sensor network has a mechanism to check

identities, the adversary has to steal IDs from compromised nodes. In a multipath or dispersity routing protocol, seemingly disjoint paths could in fact go through a single malicious node presenting several Sybil identities. The Sybil attack can be used in coordination with other attacks; attackers can use the Sybil attack to evade misbehavior detection mechanisms. The Sybil node can "split the blame" [50] by acting maliciously using multiple identities and not having any of the Sybil identities to misbehave enough for the system to take action. In order to prevent a Sybil attack, the keys must be generated and distributed such that nodes can validate each other's identities.

2.5.6 Sinkhole Attack

An adversary can use the sinkhole attack to attract some part or all of the communication to a particular node or group of nodes. After successfully creating a sinkhole, the adversary can launch a selective forwarding attack.

Sinkholes can be created by making a node look attractive to other nodes in terms of routing. A malicious node can advertise a high quality link to the sink by either spoofing or replaying routing messages. Some of the routing protocols may actually verify the speed and reliability of the advertised link by end-to-end acknowledgements containing latency information. In this case, the adversary can actually establish a high quality link to the sink using a device with a powerful transceiver, e.g. laptop, or use wormhole attack, as described in the next section.

Due to the real or imaginary high quality link to the sink, surrounding nodes send their messages, destined to the sink, to the malicious node. The sinkhole's area of influence may cover several hops away from the malicious node because routing advertisements propagate through the sensor network. An attacker can easily perform selective forwarding attack using sinkholes and drop any of the packets send from its area of influence.

2.5.7 Wormhole Attack

In the wormhole attack, an adversary tunnels messages received in one part of the network over a low latency link and replays them in a different part [48]. The adversary can easily establish a fast link by employing powerful transceivers or optical links. It is very easy to build a basic wormhole; even no node compromise is required. The adversary can replay packets received from one particular node at another part of the network. It is more effective if one end of the wormhole is near the sink and the other end is away from sink. The attacker can convince nodes away from the sink that they are only a few hops away from the sink using the wormhole. If the adversary's wormhole creates an out-of-band route to the sink, which is significantly better than alternative routes, all the surrounding nodes will prefer the wormhole. Attacker can use the wormhole in order to create a sinkhole, see Figure 2.2.

Wormholes are difficult to cope with because a wormhole can be created even if sensor messages are encrypted and authentication methods are used. Detection of wormholes is even more difficult when it used in conjunction with Sybil attack. Compromised nodes at the ends of the wormhole can pretend to be a cluster of nodes. An adversary can harm the sensor network by using the wormhole attack in combination with selective forwarding.

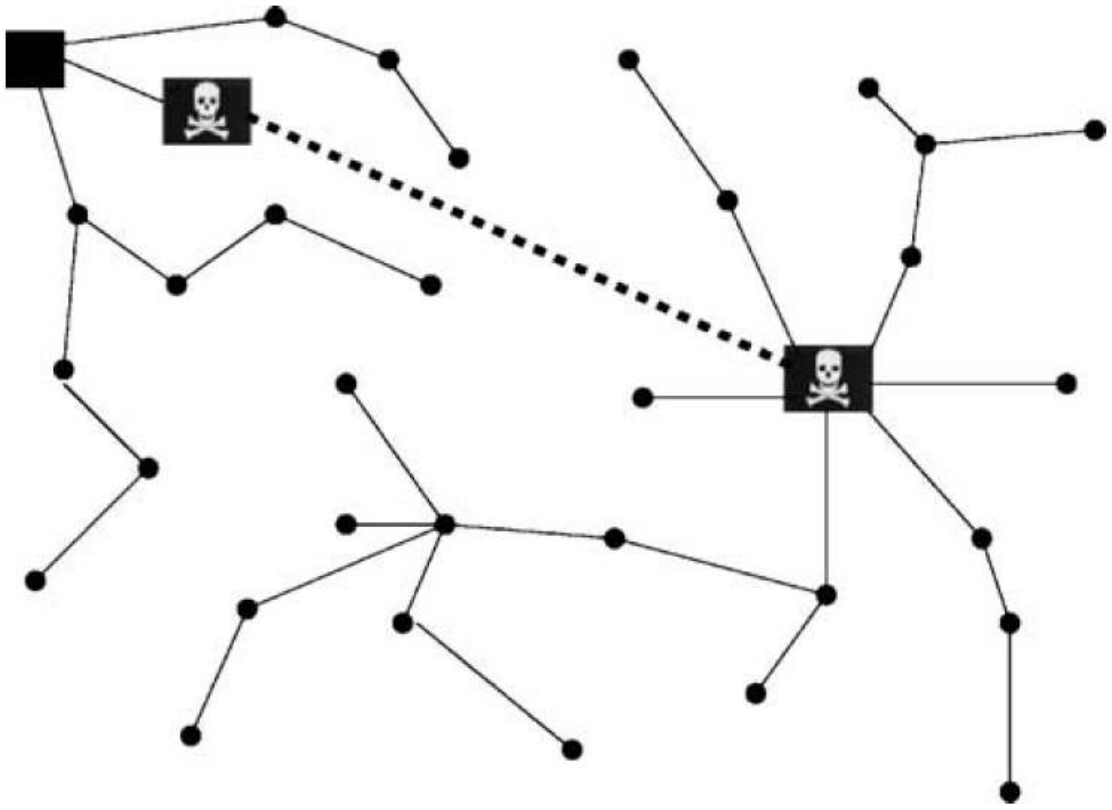

Figure 2-2 An adversary using a wormhole to create a sinkhole.

2.5.8 HELLO Flood Attack

HELLO messages are required in most of the protocols' bootstrapping phase; nodes announce themselves to their neighbors via HELLO messages. A node receiving such a HELLO message thinks that it has a one-hop neighbor. However, it may be wrong; a laptop-class attacker with enough transmission power can trick sensor nodes away from the attacker to believe that the attacker is their one-hop neighbor [48]. Similar to a wormhole attack, if the attacker resides near the sink, it can convince other nodes that they can send packets in several hops using the malicious node of the attacker. However, when sensor nodes, sufficiently far away from the adversary, try to send packets to attacker, their packets will be lost.

HELLO flood attack is especially effective against protocols that depend on localized information exchange between neighboring nodes for topology maintenance or flow control. The adversary does not necessarily need to compromise nodes and create legitimate broadcast messages. It can simply replay overheard HELLO messages with enough power to be heard over a large area. HELLO floods can also be thought of as one-way, broadcast wormholes [48].

2.5.9 Acknowledgement Spoofing

Some of the protocols for sensor networks rely on implicit or explicit acknowledgements. For routing protocols, acknowledgments can determine the quality and reliability of link. In a sensor network, an adversary can easily spoof acknowledgements of overheard messages addressed to neighboring nodes.

An attacker can use acknowledgement-spoofing method to convince sensor nodes that a weak link is strong and a depleted or dead node is still functional [48]. Sensor nodes may think that their messages are received at the other end of the link; however, their messages are actually lost. With acknowledgement spoofing, an attacker can mount a selective forwarding attack by encouraging the target node to transmit packets on those links.

2.6 Previous Work on Key Distribution Mechanisms

In this section, we give detailed description of some of the major key predistribution schemes. In sensor networks, nodes use predistributed keys directly or generate pairwise keys using given cryptographic material. The challenge is to distribute keys or keying materials efficiently. Key distribution mechanisms are essentially trade-offs between resiliency and resource-consumption; the more resources, in terms of memory, computational complexity and communication cost, are utilized for security, the harder for the attackers to compromise nodes and harm the sensor network. Different key distribution mechanisms propose trade-offs of varying nature. At one end of the trade-off, we can

achieve minimum cost with very poor security by using one global key, such that every node shares the same key with all other nodes. Resource allocation is minimal because, each node has to store only one key. On the other hand, even if only one node is compromised, all the secure links are compromised. In the following sections, we describe some of the major key predistribution schemes and we classify them according to their proposed keying styles, similar to the classification in [22].

2.6.1 Pair-wise Key Predistribution Solutions

Chan et al proposes *random pairwise key scheme* in [7], which trades off high resiliency for inefficient memory usage. According to Erdös and Renyi's work [19], we can calculate the smallest probability p that any two nodes are connected such that the entire graph is connected with high probability c. To achieve this probability p in a network with N nodes, each node needs to store only a random set of Np pairwise keys instead of exhaustively storing all $N - 1$ [7]. In this scheme, maximum supportable network size, N, depends on p and m, number of keys a node can store.

$$N = \frac{m}{p}$$

At the *key setup* phase, identity of each node is matched up with randomly selected m other node identities. A pairwise key is generated for each pair of nodes and then, the key is stored in that pair of nodes' memory, along with the ID of the other node. At the *shared key discovery phase*, each node broadcasts its ID; therefore, each node sends one message, and receives one message from each node within its radio range. A node can determine if it shares pairwise keys with its neighbors by searching their IDs in its key ring.

Random pairwise keys scheme has node-to-node authentication property. Because node IDs and matching pairwise keys are pre-deployed, a node can easily determine the identity of its neighbor by searching its own key ring. In addition, because each pairwise key is uniquely created for each pair, nodes can be sure of their neighbors' identities.

Furthermore, random pairwise keys scheme has perfect resiliency against node capture. Since each pairwise key is unique, capture of any node does not allow the adversary to decrypt any additional communications in the network besides the ones that the compromised node is directly involved.

Liu and Ning proposes *closest (location-based) pair-wise keys pre-distribution scheme* in [11]. Their scheme is an improvement on Chan et al's random pairwise key scheme, that takes advantage of location information. Sensor nodes are deployed in a rectangular area. Each node has an *expected location*, which can be predicted and predetermined. After deployment, sensor nodes reside in their *actual location*. The difference between expected location and actual location of is the *deployment error*, which can be modeled by a *probability density function*. The nearer two nodes have their expected locations, the more probable they are in the communication range of each other. The idea is to make each sensor share pair-wise keys with its *c* closest neighbors.

In [11], Liu and Ning present a basic and an extended version of their key predistribution scheme. In the extended version, they use a technique based on a pseudo random function (PRF) and a master key shared between each sensor and the setup server. Their scheme achieves a small and fixed storage overhead in each sensor no matter how the sensors are deployed, and no extra communication overhead is introduced during the addition of new sensors. At the *key setup* phase, for each sensor *u*, the setup server randomly generates a master key K_u and randomly selects a set *S*, with size *c*, of other sensor whose expected locations are closest to that of *u*. Then, for each node $v \in S$, the setup server generates a pairwise key shared between *v* and *u*, $k_{u,v}$, using master key of *v* and ID of *u*: $k_{u,v} = \mathrm{PRF}_{\mathrm{Kv}}(u)$. Node *u* stores the pairwise key, while node *v* can generate $k_{u,v}$ using the PRF, its master key and ID of *u*. In this scheme, sensor addition is easy, a new node *a* can be preloaded with pairwise keys generated with master keys of *c* nodes, nearest to node *a*'s expected location.

Liu and Ning's key predistribution scheme decreases memory usage, and performs a good key connectivity if deployment errors are low. Similar to Chan et al's random

pairwise key scheme, this scheme has very good resiliency. However, it introduces a computational overhead such that nodes have to compute pairwise keys using a PRF for some cases.

2.6.2 Random Key-Chain Based Key Predistribution Solutions

Eschenauer and Gligor proposed a *basic probabilistic key pre-distribution scheme* in [1]. Their scheme uses a randomized approach in order to achieve a well-connected sensor network with reasonable security. Theoretically, the basic scheme is based on Erdös and Renyi's work [19]. According to Erdös and Renyi's work, for monotone properties, there exists a value of p, which is the probability that there exists a shared key between any two nodes, such that the property that the random graph is connected moves from "nonexistent" to "certainly true" in a very large random graph. A uniformly distributed sensor network with large number of nodes forms a random graph, which Erdös and Renyi's theory applies.

At the key predistribution phase, a setup server randomly generates a large pool of P keys with their identities. For each sensor node, setup server draws k keys and stores these keys and their IDs in that node. These k keys form a node's key ring. At the shared key discovery phase, nodes broadcast the key IDs in their key rings. If two neighboring nodes have a shared key, they can directly use that key to secure their communication. The third step is the path key establishment phase. In this phase, *path-keys* are generated for pairs of neighboring sensor nodes that do not share any keys but are connected by two or more links at the end of the shared-key discovery phase. One of the nodes sends a randomly generated key to the other node over a path of secure links. All the sensor nodes must finish the shared key discovery phase in order to begin path-key establishment phase.

In an attack against the sensor network, an adversary can physically capture nodes and have complete control over captured nodes. When an adversary captures a node, he obtains only k keys of a single key ring, which means he has a probability of approximately k/P to attack successfully any link in the sensor network.

It is desirable to have a sensor network, whose secure links form a highly connected random graph. If p is defined as the probability that a shared key exists between any two sensor nodes, and N is the number of network nodes, then $d = p\ (N - 1)$ is the expected degree of a node, i.e. the average number of edges connecting that node with its neighbors. When p is zero, there is no edge in the network and when p is one, the network is fully connected. The connectivity of the sensor network depends on the key ring size, k, global key pool size, P, and the network of size, N. Using the Erdös and Renyi's work on random graph theory, it can be found that desired probability P_c for graph connectivity is:

$P_c = e^{e^{-c}}$ and $p = \frac{\ln(n)}{N} + \frac{c}{N}$, where c is any real constant. Therefore, for given N, we can calculate the required p and d, node degree, for a desired P_c value. Also, note that, because of wireless radio communication limits, the number of neighbors of a node, n', is much less than N. Therefore, the probability of two neighboring nodes sharing a key, p', becomes $p' = \frac{d}{(n'-1)} >> p$. The parameters k, size of key ring, and P, size of global key pool, also determines p' as follows:

$$p' = 1 - \frac{((P-k)!)^2}{(P-2k)!P!}$$

For example, when size of the key pool, P, is 10000 keys, only 75 keys are needed to be stored into key rings in order to have p equal to 0.5. If key pool enlarged 10 times, P=100000, the required key ring size becomes 250, which is only 3.3 times larger than the previous case.

There have been proposals to improve the security of links and resiliency in Eschenauer and Gligor's basic scheme. One approach is to increase the minimum number of shared keys required in the shared key discovery phase. This approach is presented by Chan et al in [7] as *q-composite random key predistribution scheme*. In the basic scheme,

any two neighboring nodes need to find a single common key from their key rings to establish a secure link. In the q-composite scheme, q common keys are required, where $q>1$. At the shared key discovery phase, a secure link, $K_{a,b}$, key between node a and node b is set as the hash of all common keys $K_{a,b} = hash\ (K_1 \| K_2 \| K_3 \| \ldots \| K_{q'})$, where q' is the number of shared keys between node a and b, $q' > q$.

As the amount of key overlap, q, increases, it becomes exponentially harder for an attacker with a given key set to break a link. However, in order to keep the probability p that any two nodes can establish a secure link the same, either key pool size, P, has to be decreased or number of keys in a key ring, k, has to be increased. Either way, the attacker gains a larger portion of the key pool with each node capture. For small-scale attacks, q-composite scheme provides improved node capture resiliency. However, as the number of captured nodes increases, it becomes easier for the attacker to compromise new links.

One way to improve the performance of key predistribution schemes is to prevent nodes that are far away from carrying common keys in their key rings. This can be achieving by using location information. Du et al propose a location-aware scheme, *key predistribution using deployment knowledge*, in [8]. Du et al, present a group-based deployment model and a key predistribution scheme in their paper.

The group-based deployment model is described in [8] as follows:

1. N sensor nodes to be deployed are divided into $t \times n$ equal size groups so that each group, $G_{i,j}$, for $i = 1, \ldots, t$ and $j = 1, \ldots, n$, is deployed from the deployment point with index (i, j). Let (x_i, y_j) represent the deployment point for group $G_{i,j}$.

2. The deployment points are arranged in a grid. The key predistribution scheme for grid-based deployment can be easily extended to different deployment strategies.

3. During deployment, the resident points of the node k in group $G_{i,j}$, follow the probability distribution function $f^{ij}_k(x, y \mid k \in G_{i,j}) = f(x - x_i, y - y_j)$. The pdf f(x,y) used in this paper is the two dimensional Gaussian distribution.

At the key predistribution phase, a global key pool S is randomly created with size |S|. Then S is divided into $t \times n$ key pools, $S_{i,j}$ (for i = 1, . . . , t and j = 1, . . . , n). Each vertically or horizontally or diagonally neighboring key pool, $S_{i,j}$, shares a certain number of keys. The scheme defines overlapping factors a and b such that two horizontally or vertically neighboring key pools share exactly $a|\ S_c\ |$ keys, where $0 \leq a \leq 0.25$. Two diagonally neighboring key pools share exactly $b|\ S_c\ |$ keys, where $0 \leq b \leq 0.25$ and $4a + 4b = 1$. Two non-neighboring key pools share no keys. The key pool setup is summarized in Figure 2.3. After key pools are setup, for each node, m keys are selected from the corresponding key pool and loaded into that node's key ring.

The shared key discovery phase is similar to that of Eschenauer and Gligor's [1]. Sensor nodes broadcast their key ring and if two neighboring nodes share one or more keys, then they have a secure link. Two nodes from different zones can also establish a direct link because key pools are setup such that they have common keys. As overlapping factors a and b increases, the probability that any two nodes from neighboring zones share a key increases. Du et al's scheme has very good connectivity and node capture resiliency performances.

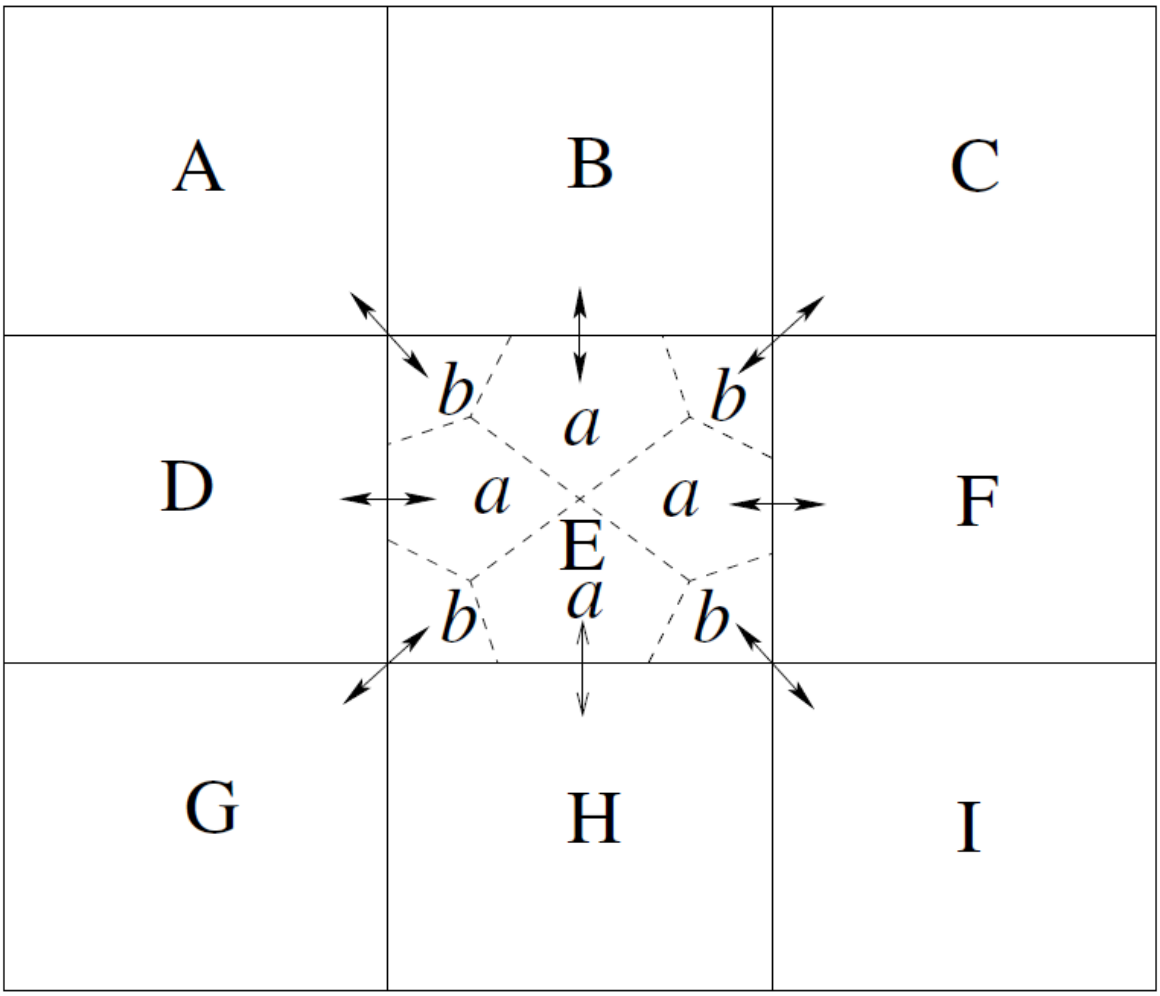

Figure 2-3 Shared keys between neighboring key pools.

2.6.3 Key Matrix Based Dynamic Key Generation Solutions

.

Instead of storing keys into sensor nodes' key ring and letting nodes to directly use them for encrypting messages, it is possible to store a small amount of information to each sensor node so that every pair of nodes can calculate a pairwise key, and use it as the link key.

Blom proposed a key pre-distribution scheme in [2] that allows any pair of nodes to find a secret pairwise key between them. Compared to the (N−1)-pairwise-key predistribution scheme, where N is the number of nodes in the sensor network and each node has pairwise keys with every other node, Blom's scheme only uses λ+1 memory

spaces with λ much smaller than N. The tradeoff is that, unlike the (N−1)-pairwise-key scheme, Blom's scheme is not perfectly resilient against node capture. Instead it has *λ-secure property*, which is explained in [9] as: as long as an adversary compromises less than or equal to λ nodes, uncompromised nodes are perfectly secure; when an adversary compromises more than λ nodes, all pairwise keys of the entire network are compromised. Apparently, a larger λ leads to a more secure network. However, there is a tradeoff between λ and memory usage. The parameter λ determines the amount of information stored into sensor nodes' memory. As λ increases, memory usage grows, too.

Blom's scheme uses a public $(\lambda + 1) \times N$ matrix G and a private $(\lambda + 1) \times (\lambda + 1)$ symmetric matrix D which is generated over GF(q). Matrix G must have $(\lambda + 1)$ linearly independent columns to provide λ-secure property. $N \times (\lambda + 1)$ matrix A is generated as $A = (D \cdot G)^T$, where $(D \cdot G)^T$ is the transpose of $(D \cdot G)$. Matrix D is completely secret, setup server does not share with any other entity, whereas, one row of matrix A will be disclosed to each node in the sensor network. The key matrix is defined as a symmetric $N \times N$ matrix $K = A \cdot G = (D \cdot G)^T \cdot G$. For the matrix K, $K_{ij} = K_{ji}$, where K_{ij} is the element located in the i^{th} row and j^{th} column of matrix K. Figure 2.4 illustrates how matrix K is generated.

For each node s_i, where $i = 1, \ldots, N$, Blom's scheme distributes

1. i^{th} column of matrix G, G_i, which is a vector of size $\lambda + 1$,

2. i^{th} row of matrix A, A_i, which is a vector of size $\lambda + 1$.

When two neighboring nodes, s_i and s_j, wants to setup a secure link, they first exchange their public columns of matrix G. Then, their shared key is generated as $K_{ij} = A_i \times G_j$ and $K_{ji} = A_j \times G_i$. The shared secret key generated is shown in Figure 2.4.

As a result, two nodes compute the same secret key by exchanging only columns of matrix G, which is public information. λ-secure property guarantees that no nodes other than s_i and s_j can compute K_{ij} or K_{ji} if no more than λ nodes are compromised. The cost of a

single shared key generation for each node is 1) multiplication of two vectors of size λ + 1 where the elements of the vectors are as large as the corresponding cryptographic key size, 2) transmission of a λ + 1 element vector.

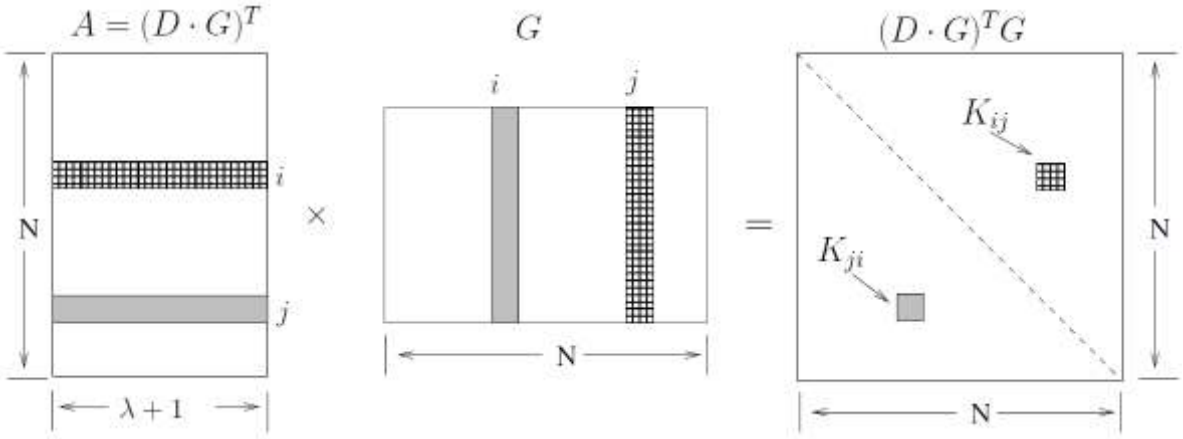

Figure 2-4 Matrix A, G and K in Blom's scheme.

Du et al *proposes multiple-space key predistribution scheme* [9] that uses Blom's scheme as a building block and improves its resiliency. A *key space* is defined as a tuple (D, G), where matrices D and G are as defined in Blom's scheme.

At the key predistribution phase, Du et al propose generating ω key spaces, where ω>2. A single public G matrix and a set of ω private D matrices are generated. These matrices form ω key spaces $S_i = (G, D_i)$, where $i = 1, \ldots, \omega$. Then, for each key space, matrix A is computed, $A_i = (D_i \cdot G)^T$, where $i = 1, \ldots, \omega$. For each sensor node, a set of τ spaces are randomly selected among these ω spaces, $2 \leq \tau < \omega$. Matrix shares for each selected space are stored to the sensor node as in Blom's scheme.

At the key agreement phase, sensor nodes exchange IDs of key spaces they store and discover whether they share any spaces with their neighbors. If any two nodes have a key space in common, they can generate a pairwise secret key using the Blom's scheme. It is possible that two neighboring nodes do not share a common space, in that case, they have to apply path-key establishment phase to establish a key through intermediate nodes.

Connectivity of the sensor network depends on ω and τ, as more key spaces are stored in a node, the possibility of having common key spaces with neighboring nodes increases. Note that, parameters ω and τ do not alone determine the memory cost. By keeping ω and τ fixed, and by changing only λ, we can change the memory cost for each node and the resiliency of the sensor network, while keeping the connectivity of the network the same. A node carries τ +1 vectors of size λ+1.

3 TWO-TIER LOCATION AWARE KEY PREDISTRIBUTION SCHEME

In this section, we give a detailed description of our key predistribution scheme. In our scheme, we exploit the predeployment knowledge of sensor nodes in order to improve the performance of key predistribution. In an ideal key distribution scenario, a node should store shared keys with only its neighbors in its memory; there should not be any unused keys. Therefore, the memory of a sensor device, which is a scarce resource in sensor devices, would not be wasted. However, it is not possible to predict the precise location of sensor nodes and their neighbors before deployment, but we can keep the nodes near to a target location by deploying them in bundles. We arrange target locations in a grid fashion and we determine which bundle will be deployed at which target location. We call each cell of the grid a *zone* and center of each zone is the target location of deployment. Before deployment, separate key spaces are created for each zone according to our key predistribution scheme, which will be described in detail in the following sections. Using this method, we can have probabilistic information on the neighbors of a node. By utilizing this probabilistic location information about sensor nodes in our key predistribution scheme, we increase the average number of shared keys between nodes and lower the required memory size.

We employ a two-tier approach in our key predistribution scheme; there are special sensor nodes in each zone with larger memory and better power source. We call these special nodes, *agent nodes*. Agent nodes are required to establish *secure links* between sensor nodes from neighboring zones. A secure link exists between two nodes if they both own at least one key in common and if they are neighbors.

Let us define the parameters and symbols used in this scheme:

N	number of nodes in each zone
Z	number of zones in the sensor network

ω	number of key spaces for each zone
τ	number of key spaces installed in a regular node
λ+1	number of columns (rows) in matrix A (G) (see Section 2.6.3 for further details on matrix A and G)
R	communication range for sensor nodes (in meters)
A_z	number of agent nodes in each zone
s_{ij}	ID of j^{th} sensor node in zone i
r_{ij}	resident point of node s_{ij}
m	size of each element in G matrix and size of a pairwise key in bits
k_{ij}	ID of j^{th} key space in zone i
Z_{ij}	ID of zone at i^{th} row and j^{th} column
d_{ij}	deployment point at zone Z_{ij}
G_{ij}	ID of group deployed at d_{ij}

Our location-aware key predistribution scheme consists of four phases:

- First phase is the *predistribution* phase, where keys are stored in nodes according to two different methods. The first method is the *intra-zone key predistribution* method and it is applied to all sensor nodes, both regular and agent nodes. The second method is the *inter-zone key predistribution* method; this method is applied only to agent nodes.

- Second phase is the *direct key establishment* phase, where nodes discover their neighbors and find out if they share common key spaces with their neighbors to form secure links. If they share a common key space, they generate a common pairwise key using Blom's scheme [2].

- In the third phase, nodes use the *hybrid key establishment method.* In hybrid key establishment method, secure links between regular nodes and agent nodes are established. In this hybrid method, nodes use keying material distributed at the intra-zone key predistribution phase.

- Fourth phase is the *path key establishment phase*, where nodes try to find secure paths to their neighbors, with which they do not share common key spaces, in order to establish secure link. In the path key establishment phase, sensor nodes initiate *intra-zone path key establishment process* to establish path keys with same zone neighbors and *inter-zone path key establishment process* to establish path keys with nodes from neighboring zones.

3.1 Zone-Based Deployment Model

In sensor networks, it may be possible to predict the locations of sensor nodes to a certain degree. If we can determine the locations of sensor nodes precisely, we can store cryptographic material in sensor devices such that they can establish pairwise shared keys with their neighbors and avoid using unnecessary memory for nodes away from their communication range. This way, we can achieve optimal key predistribution and have high connectivity and strong resiliency results. However, determining locations of large number of sensor nodes in advance is very difficult. One way to deploy sensors at precise locations is placing them by hand, but it is not a practical approach. Nevertheless, we can have a probabilistic knowledge on the location of sensor nodes.

In our deployment model, we divide sensor nodes into $z_x \times z_y$ equal sized groups and the target deployment area is a rectangular region with length X and width Y. Then, we arrange the deployment area into a grid with $z_x \times z_y$ equal sized zones. Center point of zone Z_{ij} is the deployment point, d_{ij} of group G_{ij}, where $i = 1, \ldots, z_x$ and $j = 1, \ldots, z_y$. Using a plane, we drop group G_{ij} over deployment point d_{ij}. The actual location of sensor nodes after deployment is named as their *resident points*, r_{ij} where $i = 1, \ldots, Z$ and $j = 1, \ldots, N$. Note that each sensor node has a resident point, whereas sensor node groups have deployment points. Resident points of sensor nodes in the same group follow the same probability distribution function.

If the probability distribution function is uniform distribution, any node can be located anywhere, regardless of its deployment point. However, uniform distribution does not provide a realistic deployment model. Thus, in our deployment model we employ two-dimensional Gaussian distribution. Using a Gaussian distribution, sensor nodes dropped at the same deployment point tend to be closer to each other and we can deduce a probability on the locations of sensor nodes.

We can determine probability of a node being at a resident point based on its deployment point using the Gaussian distribution function. When mean of the Gaussian distribution (μ) equals to the deployment point d_{ij} = (x, y) and standard deviation is σ, the probability of a node to have a resident point at r_{ij} = (x', y') is given as follows.

$$P(r_{ij} = (x', y') \mid d_{ij} = (x, y)) = f(x'-x, y'-y) = \frac{1}{2\pi\sigma^2} e^{-\left|(x'-x)^2+(y'-y)^2\right|/2\sigma^2} \quad (3.1)$$

It is desirable in our deployment model to distribute sensor nodes over the deployment area as evenly as possible. In Gaussian distribution, the distance between mean, which is the deployment point, and the resident point for a node is less than 3σ with probability 0.9987. Therefore, it is important to pick proper parameters for distance between neighboring deployment points and σ to create an evenly distributed sensor network. Because we use Gaussian distribution, we know that nodes in the same group will most probably be near to each other and using this information, we can increase connectivity by storing cryptographic material from same key spaces into same-group nodes.

A sample sensor network created using the above distribution formulation where σ = 10 is plotted in Figure 3.1. The sample sensor network is a 4×4 grid, each zone is a $30m \times 30m$ square and center of each zone is the deployment point of a sensor node group. There are three agent nodes in each zone; A_z = 3. In order to achieve a homogenous distribution of sensors, the distance between deployment points is set to 3σ. As shown the

figure below, the deployment and corresponding resident points are deviated according to the distribution.

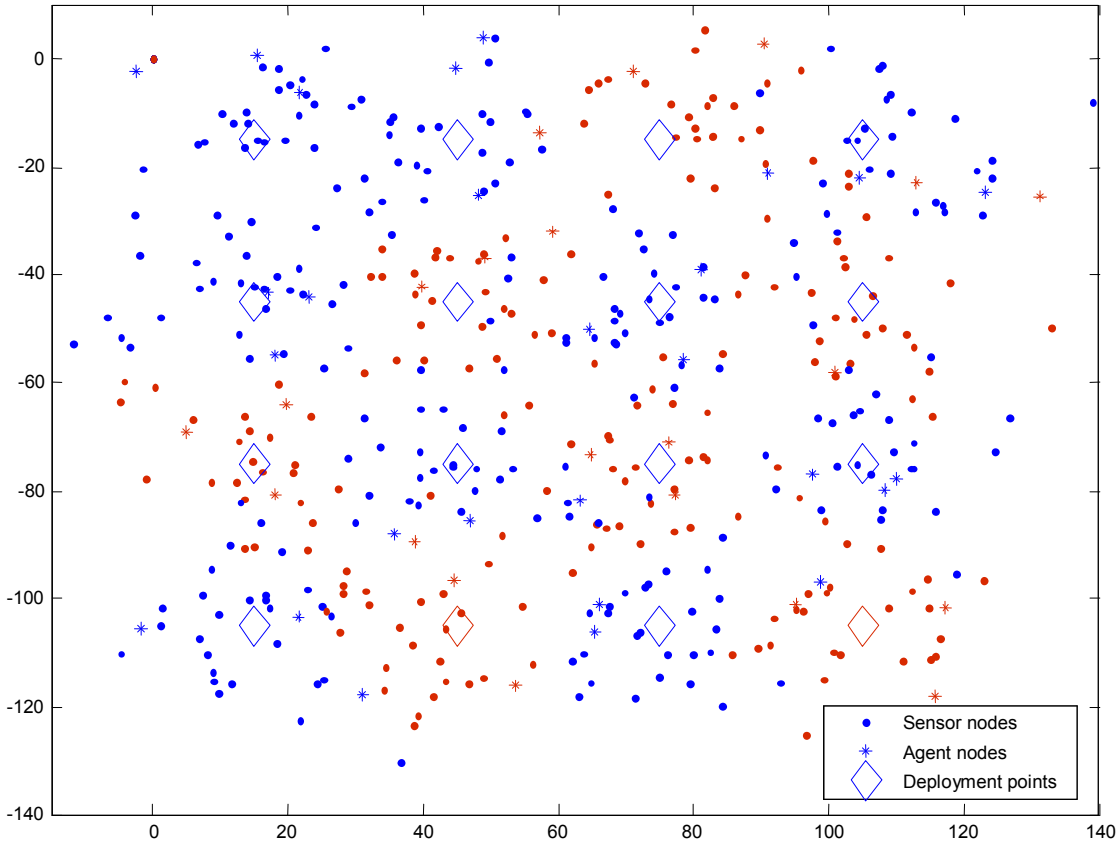

Figure 3-1 A sample sensor network, σ=10m, N=30 and distance between adjacent deployment points is 3σ

3.2 Predistribution Phase

In key predistribution phase, we describe the method of how keys are distributed to nodes. We define two methods in this phase: *intra-zone key predistribution* method and *inter-zone key predistribution* method. In intra-zone key predistribution method, setup server distributes keys required for establishing secure links between nodes from the same zone and intra-zone step applies for both regular nodes and agent nodes. In inter-zone key predistribution method, setup server distributes keys to agent nodes so that agent nodes from neighboring zones can securely send messages to each other. Note that agent nodes

from the same zone can use their cryptographic material installed in the intra-zone step to establish secure links between each other.

3.2.1 Intra-zone key predistribution method

When any two nodes, regular-regular, or regular-agent, or agent-agent, from the same zone want to establish secure links with each other, they use cryptographic material distributed in this step. In the intra-zone key predistribution method, we use a method that is build on Blom's key predistribution scheme [2], which is explained in Section 2.6.3, and Eschenauer and Gligor's probabilistic and random key predistribution scheme [1]. Using this scheme, any two nodes having shares from the same matrix can compute a secret pairwise key and use this key to create a secure link.

Setup server generates a single public matrix G, as described in Section 2.6.3, all λ+1 columns of this matrix are linearly independent. For each zone, setup server generates ω random and symmetric D matrices and uses these matrices to compute ω A matrices, each A matrix is a $N \times (\lambda + 1)$ matrix, and each D and A matrix pair makes up a key space. Each key space has a unique ID, k_{ij}, where $1 \leq i \leq Z$ and $1 \leq j \leq \omega$. Sensors can use these key space IDs to find out if they have common key spaces with their neighbors. Then, for each node, s_{ij}, setup server picks τ key spaces and stores corresponding row of A matrix and corresponding column of G matrix to node s_{ij}. For node s_{ij}, setup server picks j^{th} row of matrix A and j^{th} column of matrix G.

In order for two neighboring nodes to compute a common key, they need to know each other's public columns in G matrix. Note that each element in G matrix is in finite field *GF(q)*, where q is a large prime number. In addition, as described in Section 2.6.3, λ+1 columns of matrix G must be linearly independent to have the λ-*secure* property [2]. Setup server can generate all columns of matrix G by using a primitive element in *GF(q)*, namely p. It is known that each non-zero element in *GF(q)* can be computed by some power of p; p^i, $0 < i \leq q-1$, and when $i \neq j$, $p^i \neq p^j$. Using these facts, it can be seen that it is feasible to generate a public G matrix by using a single primitive element:

$$G = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ p & p^2 & p^3 & \dots & p^N \\ p^2 & (p^2)^2 & (p^3)^2 & \dots & (p^N)^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ p^\lambda & (p^2)^\lambda & (p^3)^\lambda & \dots & (p^N)^\lambda \end{bmatrix}$$

Above G matrix is a form of Vandermonde matrix [18], it can be shown that any λ+1 columns of the above matrix is linearly independent as long as p, p^2, p^3, ..., p^N are all distinct. Therefore, setup server can pick a primitive root, p, of GF(q) and use it to generate all columns of G and store a copy of p in all sensor nodes.

At the end of this phase, all nodes will have τ rows and one primitive element stored in their memory. These cryptographic materials will occupy $(\tau(\lambda+1)+1)m$ bits in sensor devices' memory.

3.2.2 Inter-zone key predistribution method

In our key distribution scheme, agent nodes play a crucial role, such that regular nodes depend on agent nodes to establish secure links over different zones. We use random-pairwise keys to establish common keys between agent nodes.

Before sensor deployment, setup server generates unique random pairwise keys for each agent node pair; there are only two copies of a pairwise key. For an agent node s_{ij} in zone i, setup server generates pairwise keys that s_{ij} shares with all agent nodes in neighboring zones of zone i. Then, these pairwise keys are stored in s_{ij} along with IDs of corresponding agent nodes. In Figure 3.2, we show an agent node sharing pairwise keys with agent nodes from neighboring zones.

As explained in [7], random pairwise keys have node-to-node authentication property, so that agent nodes can verify the identities of other agent nodes they are communicating.

Adversaries cannot impersonate an agent node as long as that node is not captured by an attacker. Another attractive feature of random pairwise keys is that they have perfect node capture resiliency, meaning that when a pairwise key is compromised by adversaries, only the secure link, that compromised key is used, is affected. Adversaries cannot compromise any other secure link since pairwise keys have only two copies.

At the end of this step, setup server will store $A_z \times N_{neigh} \times m$ bits in each agent node, where N_{neigh} is number of neighboring zones of corresponding zone, $3 \le N_{neigh} \le 8$. As explained in the intra-zone key predistribution step, agent nodes have also $(\tau(\lambda+1)+1)m$ bits stored in their memory. Thus, agent nodes must have larger memory than regular nodes. Considering that there will be limited number of agent nodes in each zone, this is a practical approach. Although random pairwise keys scheme has useful features, like node-to-node authentication and perfect node capture resiliency, it has the disadvantage of larger memory consumption than other key distribution scheme and it is not scalable as the number of agent nodes in each zone grows. Again, since agent nodes will be limited in number, it is still practical to employ random-pairwise keys. As we will show in the following sections, with only eight agent nodes in each zone, our scheme can have very high connectivity and "node capture resiliency" performances. In our scheme, expanding the sensor network by deploying new zones does not increase the memory cost for agent nodes because agent nodes contact with only neighboring zones and number of neighboring zones can be at most eight. Thus, scalability problem of random pairwise keys is not applicable to our key predistribution scheme.

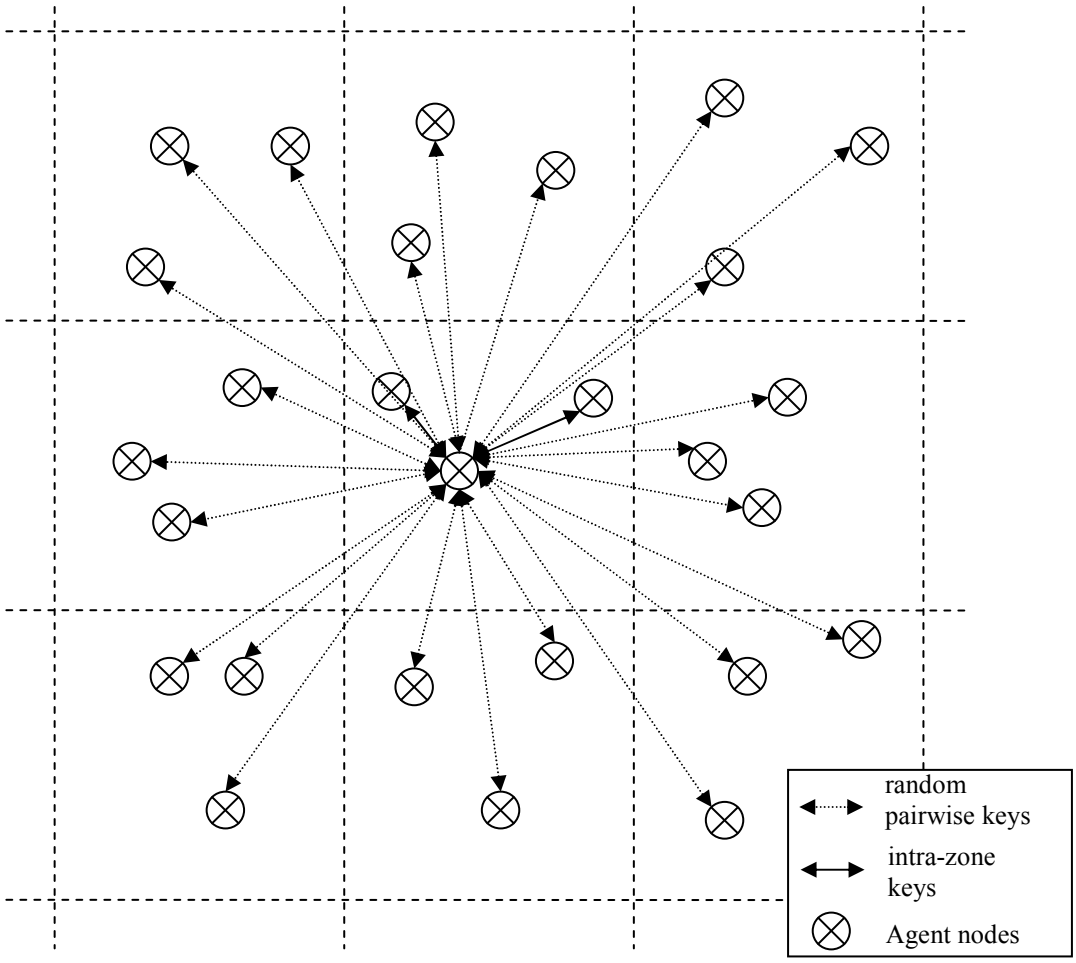

Figure 3-2 Agent nodes share pairwise keys with other agent nodes from neighboring zones

3.3 Direct Key Establishment Phase

After deployment, sensor devices try to establish secure links with all of their neighbors. Direct key establishment phase explains how sensors can share pairwise keys with their neighbors and use these keys to establish secure links. Note that, if two nodes are within each other's communication range, we define these two sensor nodes as *neighbors*. If two neighboring nodes have secure link between them, we define these nodes as *secure neighbors*.

Table 3-1 Direct key establishment methods

	Regular Node (Same zone)	Agent Node (Same zone)	Regular Node (Different zone)	Agent Node (Different zone)
Regular Node	*Method 1*	*Method 1*	*Method 2*	*Method 4*
Agent Node	*Method 1*	*Method 1*	*Method 4*	*Method 3*

When two sensor nodes are neighbors and want to establish a secure link, depending on node types and zone relationship, one of four methods is used, as shown in Table 3.1. We show the possible situations and corresponding methods used in Figure 3.3.

- *Method 1:* Both sensor devices can be regular nodes or agent nodes and they can be from the same zone. If one of the nodes is an agent node, then the agent node uses its cryptographic material it received at the intra-zone key predistribution step and acts like a regular node. In this case, they have to find out if they share any key spaces. To do this, each node broadcasts a message containing the following information: (1) the node's id, (2) the indices of the key spaces it carries. When neighboring sensor nodes, s_{in} and s_{im}, receive this message, they can find out if they share a common key space. If they do not have a common key space, they can establish a secure link with the help of other nodes, which they have secure links with, as we will describe in Section 3.5. If s_{in} and s_{im} share a common key space, they can compute a pairwise key as described in Section 2.6.3. s_{in} can compute the pairwise key by using its private row from matrix A and s_{im}'s column of public matrix G, which s_{in} can generate by using s_{im}'s ID and the primitive root, that is already stored in every node. This shared key is called the *direct key*. After computing the direct key, neighboring nodes can use this key to establish a secure link and encrypt their communication.

- *Method 2:* Both of the devices can be regular nodes but they may belong to different zones. In this case, they cannot directly establish a secure link because they do not have any common key spaces. Only agent nodes from different zones can share a pairwise key and communicate securely without help of other nodes. When regular nodes, s_{im} and s_{jn} want to establish a secure link, they have to reach their nearest agent.

In Section 3.5.2, inter-zone path key establishment process, we describe how two regular nodes from different zones can establish a secure link.

- *Method 3:* Both of the devices can be agent nodes from neighboring zones. In this case, they can easily establish a secure link by exchanging IDs. They do not need to do any computation; each node can find the pairwise key shared with the other agent node just by looking at other node's ID. They do not need any intermediaries like regular nodes from different zones.

- *Method 4:* One of the sensor nodes can be a regular node, the other node can be an agent node, and they may belong to neighboring zones. In this case, they follow the inter-zone path key establishment process and during this process, the agent node acts like both a regular node and its 0-hop neighbor agent zone.

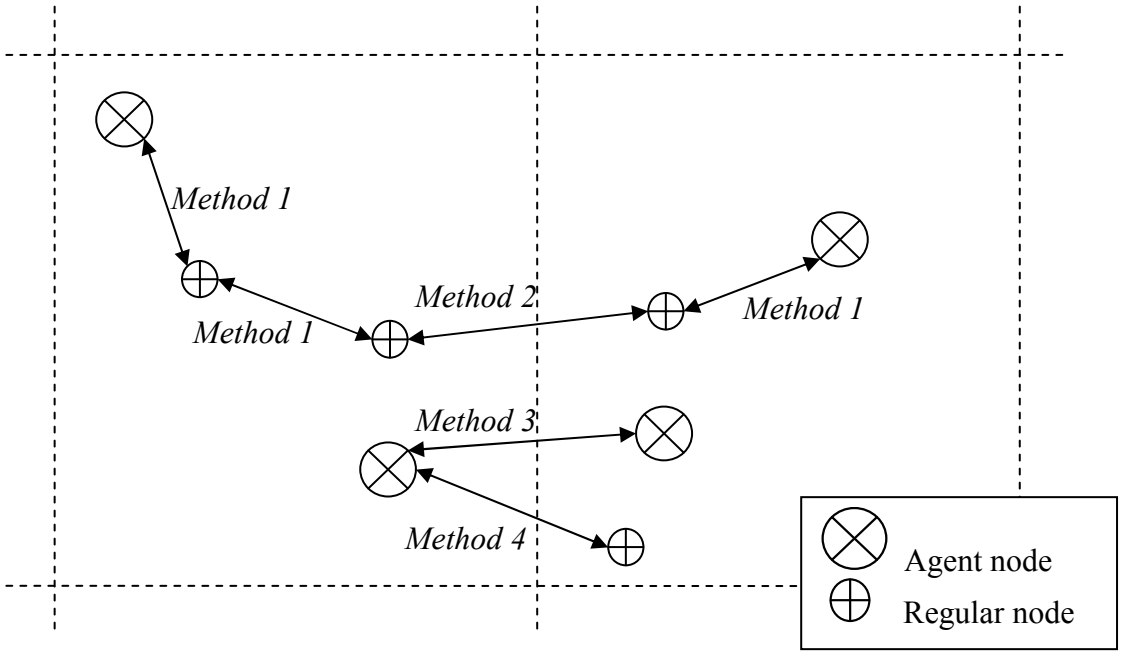

Figure 3-3 Key establishment methods for neighboring nodes

After the direct key establishment phase, the entire sensor network forms a sec*ure link graph*. In [9], secure link graph is defined as follows:

Let V represent all sensor nodes in the sensor network. In a secure link graph G(V, E) two nodes can have a edge between them only if they are neighbors and they share a secret key.

3.4 Hybrid Key Establishment Method

After direct key establishment phase, sensor nodes establish secure links with their neighbors. If a regular node has a direct key with an agent node, i.e. agent node is a secure neighbor of regular node, that node can perform inter-zone key establishment process. In case a regular node has no agent node within its radio communication range, i.e. none of regular node's 1-hop neighbors is an agent node; it has to find an agent node within several hops range and try to establish a pairwise key using *hybrid key establishment method*.

Every regular node needs to have a contact with an agent node in order to perform inter-zone path key establishment. Some of the regular nodes can find an agent node within their communication range. In that case, regular nodes can treat agent nodes like regular nodes and establish secure links with neighboring agent node(s) in the direct key establishment phase. If the regular node and the agent node do not share any key spaces, they can establish a path-key using the method explained in Section 3.5.1.

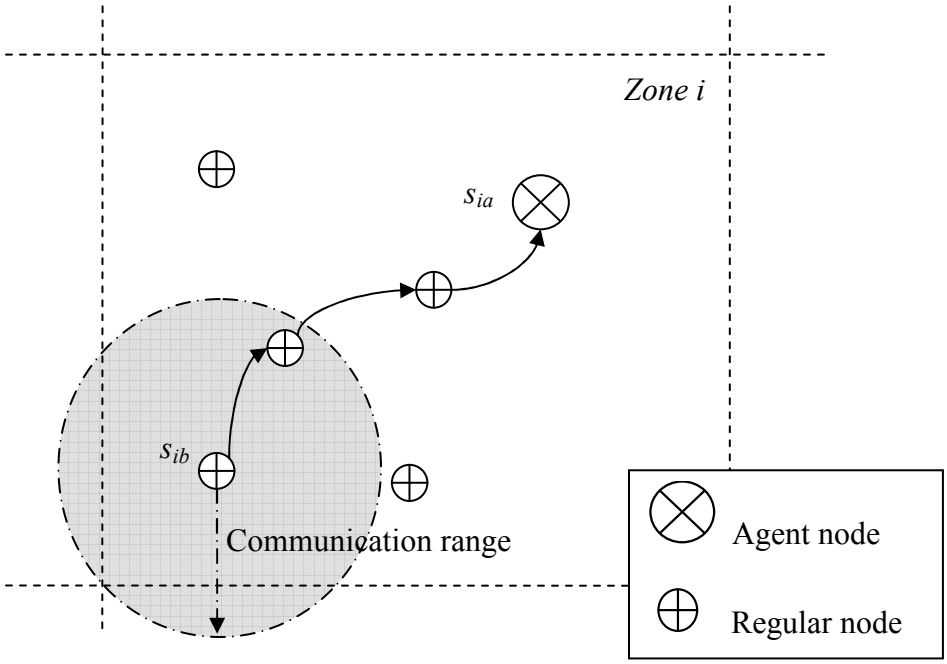

Figure 3-4 Regular node s_{ib} establishes a pairwise key with agent node s_{ia} using *hybrid key establishment method*

Because there is limited number of agent nodes in each zone, not every regular node may be able to find an agent node in 1-hop range. Nevertheless, regular nodes can still share key spaces with an agent node and if they can exchange their key space IDs over a secure path, they can compute their secret shared key even if they are several hops away. As an example, in Figure 3.4, regular node s_{ib} has an agent node 3-hops away and tries to establish a pairwise key using hybrid key establishment method.

The steps of hybrid key establishment method are as follows:

1. If a regular node, s_{ij}, where $1 \leq i \leq Z$ and $1 \leq j \leq N$, does not have an agent node in its neighbor list, it tries to find the nearest agent node by broadcasting a query including its key space IDs. If s_{ij}'s neighbors have an agent node in their neighbor lists, they forward the query to the agent node. If there are no agent nodes in two hops, neighbors of s_{ij} forward the query to their neighbors and this flooding of queries goes on until either a hop-limit is reached or an agent node is reached. If the secure link graph is connected, s_{ij} eventually finds an agent node.

2. If more than one agent node is found at the same hop count, they all get the same query and reply by sending their key space IDs. Node s_{ij} picks one of the agent nodes, let the chosen agent node be s_{ia}, and sends back an acknowledgement message to s_{ia}.

3. Since both s_{ia} and s_{ij} knows each other's key space IDs, they can find out their shared key spaces and compute their pairwise key as explained in Section 3.3.

It is possible that regular node s_{ij} does not share any key spaces with any of the agent nodes in its zone. In this case, a path key can be established between s_{ij} and its nearest agent node using the method explained in Section 3.5.1.

3.5 Path Key Establishment Phase

After direct key establishment phase, a sensor node, s_{ij}, may end up in a case where it cannot find any shared keys to establish secure links with one or more of its neighbors. If s_{ij}'s neighbor is from the same zone, s_{ij} initiates the *intra-zone path key establishment process*. Else, if s_{ij}'s neighbor is from a neighboring zone, s_{ij} initiates the *inter-zone path key establishment process*.

3.5.1 Intra-Zone Path Key Establishment Process

If node s_{ij}'s neighbor, s_{in}, is from the same zone, s_{ij} tries to find a secure path to s_{in} with the help of its secure neighbors. As long as *secure link graph* is a connected graph, s_{ij} can find a path to s_{in} using only secure neighbors. The process of establishing a secure link over a secure path between same zone nodes is called *inter-zone path key establishment.*

The process works as follows. Assume node s_{ij} does not have a secure link with its neighbor node s_{in}. Node s_{ij} asks its 1-hop secure neighbors through broadcasting, to see if

they have secure links with node s_{in}. If any of the neighbors, say s_{im}, has such a secure link, then s_{im} generates a random key and sends the key to both node s_{ij} and s_{in} over secure links. Then, s_{im} removes this random key from its memory and the nodes s_{ij} and s_{in} use this key to establish a secure link. If none of the 1-hop secure neighbors has a secure link with node s_{in}, then the query is forwarded to s_{ij}'s 2-hop secure neighbors. If not found again, s_{ij} asks next hop neighbors until it finds a node that have a secure link with s_{in}. If the graph of secure links is a connected graph, a node eventually finds a secure path to any node in the sensor network.

In the performance analysis section, we show that a node can reach most of its same zone neighbors in only one hop. For a regular node to reach its different zone neighbors, it should perform the following process.

3.5.2 Inter-Zone Path Key Establishment Process

When node s_{ij}'s neighbor, s_{mn}, is from a neighboring zone, s_{ij} needs an agent node to communicate securely with s_{mn}. Thus, every regular node needs a secure path to its nearest agent node before initiating *inter-zone path key establishment process*. In the performance analysis section, we show that when there are eight agent nodes in a zone, most of the regular nodes can reach their nearest agent node in only one hop. If a regular node cannot reach any of the agent nodes in one hop, it can find a secure path to its nearest agent node with the help of secure neighbors, as described in the *hybrid key establishment method*. Assuming both s_{mn} and s_{ij} have direct or hybrid links with an agent node, inter-zone path key establishment process works as follows:

I. They exchange their and their nearest agent node's ID,

II. One of the regular nodes send IDs received from the other node to its nearest agent node over a secure link. They can decide which node will send according to some metric; it can be their energy level or which one has the smallest ID. Let s_{mn} send the message to its agent node.

III. If s_{mn} and s_{ij} are from neighboring zones, their agent nodes must share a pairwise key, as explained in the inter-zone key predistribution step. Agent nodes can easily find out their shared pairwise key, K_p, just by looking at each others ID. Node s_{mn} has either a direct or hybrid key, explained in Section 3.4, K_s, to its agent node. Node s_{mn}'s agent node generates a random key, K_r, and encrypts it with K_p; $E_{Kp}\{K_r\}$. Then s_{mn}'s agent node sends the message $E_{Ks}\{ K_r, E_{Kp}\{K_r\}\}$ to s_{mn} over a secure path or secure link.

IV. Node s_{mn} decrypts the message and retrieves K_r. Then s_{mn} sends $E_{Kp}\{K_r\}$ to its neighbor, s_{ij}.

V. Node s_{ij} sends the message, $E_{Kp}\{K_r\}$, to its nearest agent node. The agent node decrypts $E_{Kp}\{K_r\}$ and sends K_r back to s_{ij} over a secure link or secure path.

VI. Both s_{mn} and s_{ij} share K_r and can use the key to encrypt their communication.

As an example, we provide the inter-zone path key establishment process between two nodes, s_{ib} and s_{jb}. Nodes s_{ib} and s_{jb} are from neighboring zones and they already established hybrid keys with their nearest agent nodes. The example is shown in Figure 3.5:

1. Sensor node s_{jb} initiates the inter-zone path key establishment process by sending message $M_1 = \{s_{jb}, s_{ja}\}$ to node s_{ib}

2. Sensor node s_{ib} receives M_1 and sends back $M_2 = \{ s_{ib}, s_{ia} \}$ to s_{jb}

3. Assume that agent node s_{ia} and regular node s_{ib} has already established a secure link and share a direct key, K_d. Sensor node s_{ib} sends $M_3 = E_{Kd}\{\{ s_{jb}, s_{ja} \}\}$ to its agent node s_{ia}.

4. Agent node s_{ia} finds out its shared pairwise key, K_p, with s_{ja} by looking at s_{ja}'s ID. Then agent node s_{ia} generates a random key, K_r, to be used as s_{ib} and s_{jb}'s shared secret key. After that, agent node s_{ia} sends encrypted message $M_4 = E_{Kd}\{K_r, E_{Kp}\{K_r\}\}$ to s_{ib}.

5. Sensor node s_{ib} receives and decrypts M_4 and retrieves K_r and $E_{Kp}\{K_r\}$. Then, s_{ib} sends $M_5 = E_{Kp}\{K_r\}$ to s_{jb}.

6. Sensor node s_{jb} receives M_5 but cannot decrypt it because s_{ib} and s_{jb} does not yet share any key. In addition, s_{jb} does not have a direct link with agent node s_{ja}; s_{jb} can reach agent node s_{ja} in two hops with the help of s_{jc}. Assume that s_{jb} and s_{jc} has already established secure link using direct key establishment method and share a direct key, K_{d1}. Sensor node s_{jb} sends $M_6 = E_{Kd1}\{E_{Kp}\{K_r\}\}$ to $s_{jc.}$

7. Sensor node s_{jc} receives and decrypts $M_{6,}$ then retrieves $E_{Kp}\{K_r\}$. After that, s_{jc} encrypts $E_{Kp}\{K_r\}$ with K_{d2}, shared direct key with s_{jc} and agent node s_{ja}. s_{jc} sends $M_7 = E_{Kd2}\{E_{Kp}\{K_r\}\}$ to $s_{ja.}$

8. Agent node s_{ja} receives and decrypts M_7, retrieves $E_{Kp}\{K_r\}$. Then using the shared pairwise key between two agent nodes, s_{ja} and s_{ia}, s_{ja} retrieves K_r. Assume that agent node s_{ja} and s_{jb} shares a secret key, E_{Kd3}, generated using the hybrid key establishment method, explained in Section 3.4. Then, s_{ja} encrypts K_r with E_{Kd3} and gets $E_{Kd3}\{K_r\}$. Agent node s_{ja} re-encrypts the message with E_{Kd2} and sends $M_8 = E_{Kd2}\{E_{Kd3}\{K_r\}\}$ to sensor node $s_{jc.}$

9. Sensor node s_{jc} receives and decrypts $M_{8,}$ then retrieves $E_{Kd3}\{K_r\}$. After that, s_{jc} encrypts $E_{Kd3}\{K_r\}$ with K_{d1}, shared direct key between s_{jc} and s_{jb}. Regular node s_{jc} sends $M_9 = E_{Kd1}\{E_{Kd3}\{K_r\}\}$ to $s_{jb.}$

10. Lastly, sensor node s_{jb} receives and decrypts $M_{9,}$ then retrieves E_{Kd3} $\{K_r\}$. By decrypting E_{Kd3} $\{K_r\}$, s_{jb} retrieves K_r. Using K_r, s_{ib} and s_{jb} can securely communicate with each other.

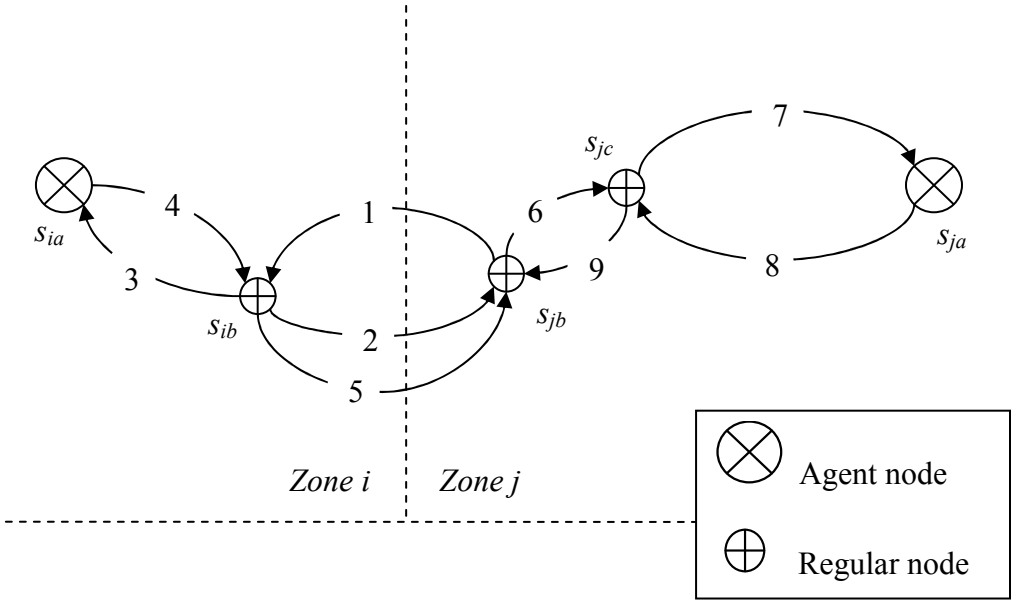

Figure 3-5 Example case: Two neighboring regular nodes, s_{ib} and s_{jb}, from different zones establish a secure link through inter-zone path key establishment process

4 PERFORMANCE EVALUATION

In this section, a detailed performance evaluation of our key predistribution scheme is provided. In order to evaluate the performance of our scheme, various simulations are used, each examining one of the performance metrics. Also for comparison, we simulated some of the well-known key predistribution schemes [1], [8], and [9]. Note that, in order to make a fair comparison with our scheme and [8], we had to modify [8]. Our key predistribution scheme has a very basic difference with some other key predistribution schemes, e.g. [8], [11], [15]: regular nodes cannot directly find a common key with neighboring nodes from adjacent zones; they have to first contact their nearest zone agent. Thus, at the first stage of key establishment, direct key establishment phase, our regular nodes can only find common keys with the same zone nodes, whereas in other schemes [8], [11], [15] nodes have mechanisms to directly establish keys with nodes from neighboring zones. In order to compare local connectivity results of our scheme with other schemes, we modified other schemes so that nodes from different zones cannot find shared keys.

4.1 Performance Evaluation Metrics

We are going to evaluate the performance of our scheme using the following metrics:

- **Connectivity:** Both global and local connectivity issues are analyzed. Local connectivity can be referred as the probability of two neighboring nodes sharing at least one key space, i.e. having a direct secure link. Assuming that key spaces are homogenously distributed among sensor nodes, local connectivity can also be defined as the average number of secure neighbors of a node. Global connectivity is the ratio of the number of nodes in the largest connected part of secure link graph to the size of the network.

- **Communication Cost:** After deployment, a node may not be able to find common key spaces with its neighbors. In that case, two neighboring nodes in the same zone can find a secure path to each other over their neighbors with which they are securely connected. If two nodes from adjacent zones need to establish a secure connection, they must include their zones' agent nodes into the key establishment process. We will analyze the communication overhead of our key predistribution scheme when two neighboring nodes cannot establish a direct secure link.

- **Resilience against Node Compromise:** One of the main threats against our scheme is the node compromise and its consequences on the rest of network. In our analysis, we assume that when an adversary captures a node, it can compromise all the cryptographic material on that node. Moreover, when a key space is compromised, some additional secure links that use this key space are also compromised. We will analyze the probability of a secure link being compromised after some number of nodes is captured.

- **Resilience against Node Fabrication and Wormhole Attack:** As described in [16], in a node fabrication attack, an attacker can fabricate nodes with new IDs using compromised cryptographic material and deploy these fabricated nodes into the sensor network. In a wormhole attack, an adversary tunnels messages received in one part of the network over a low latency link and replays them in a different part [48]. We explain how and to what degree, our scheme is resilient against node fabrication and wormhole attacks.

- **Scalability**: Our scheme's performance under different sensor network sizes is analyzed. We performed simulations for sensor networks with 100 zones and 1000 zones.

4.2 System Parameters

In our analysis and simulation, we use the following configuration.

- Deployment area is 1000m x 1000m
- Deployment area is divided into 10 x 10 zones, i.e. Z=100
- Each zone is 100m x 100m square area.
- Total number of sensor nodes is 10000 and there are 100 nodes in each zone, i.e. N=100.
- Communication range, R, for each node is 40m.
- Standard deviation of normal distribution, σ, is 50m and center of each zone is the deployment point of sensor nodes.

4.3 Local Connectivity

Local connectivity can be defined as the probability of two neighboring nodes sharing at least one key space (i.e. nodes can compute a pairwise key and establish a secure link) [9]. We denote this probability as P_{local}.

In Figure 4.1, local connectivity values of our scheme and Du et al.'s scheme [9] are shown. It can be observed that the ratio τ / ω is the determiner P_{local}, ω is the total number of keys spaces for a zone and τ is the number of key spaces installed in a sensor node. As τ increases and ω decreases, the probability that two neighboring nodes share at least one key space increases. Thus, memory used for key spaces in a sensor node does not directly affect the P_{local} values. P_{local} is independent of the size of each matrix share, λ. As explained in Section 3.2.1, the memory used for key spaces is $(\tau(\lambda+1)+1)m$ bits.

For comparison purposes, we simulated Du et al's scheme [9], which is also based on Blom's scheme [2]. Simulation results for local connectivity are shown in Figure 4.1. Du et al's scheme does not divide the sensor deployment area into grids and does not divide sensors into groups. It is a probabilistic scheme where ω key spaces are created before deployment and τ key spaces are randomly picked for each sensor and installed in each

sensor node, where $\tau < \omega$. If two neighboring nodes share one or more key spaces, they can generate a common key as described in [2, 9] and can establish a secure link.

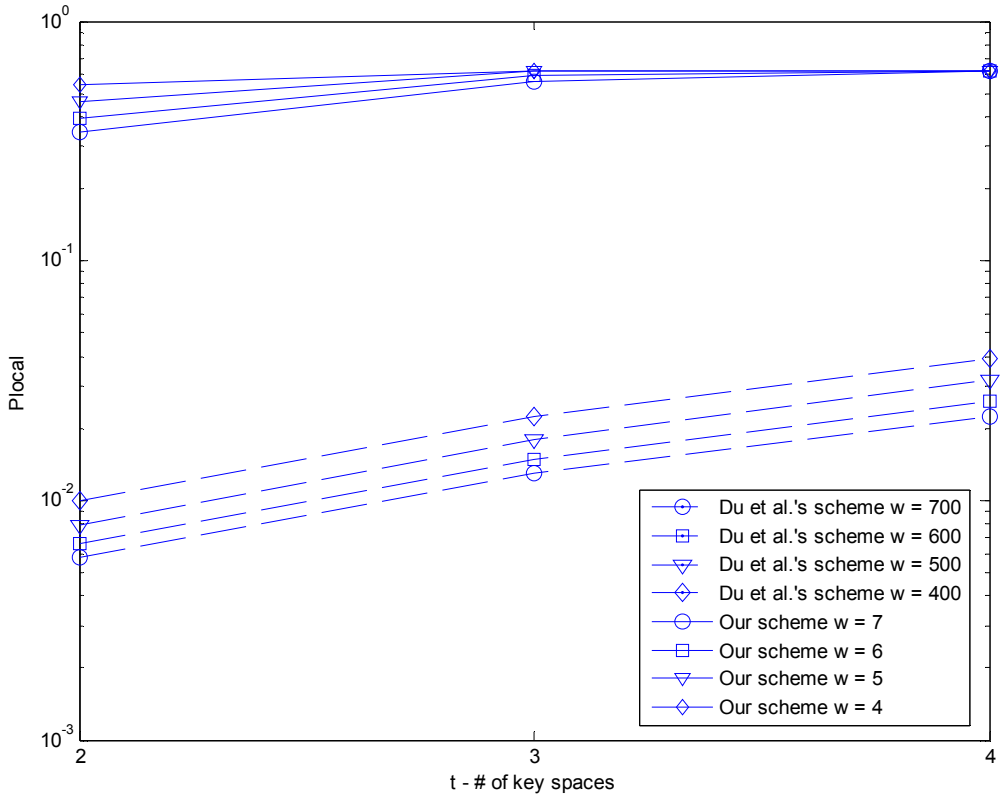

Figure 4-1 Local connectivity, P_{local}, vs. τ, # of key spaces installed in a node

Note that, in our simulations, we divide the deployment area into 100 zones, so when ω, number of key spaces available for a key space, is equal to x, total number of key spaces created for the whole network is $100x$. Therefore, both schemes are simulated under similar conditions; total number of key spaces for the whole network is the same for both our scheme and Du et al.'s scheme.

It can be observed from Figure 4.1 that local connectivity performance of our scheme is clearly higher than that of Du et al's scheme [9]. For example, when ω is 6 and τ is 3, our scheme reaches a P_{local} value of 0.5942. On the other hand, when ω is 600 and τ is 3 for Du et al.'s scheme, P_{local} is 0.0149. Figure 4.1 shows that using deployment knowledge by

dividing the sensor network into zones substantially increases local connectivity performance while keeping the memory cost the same.

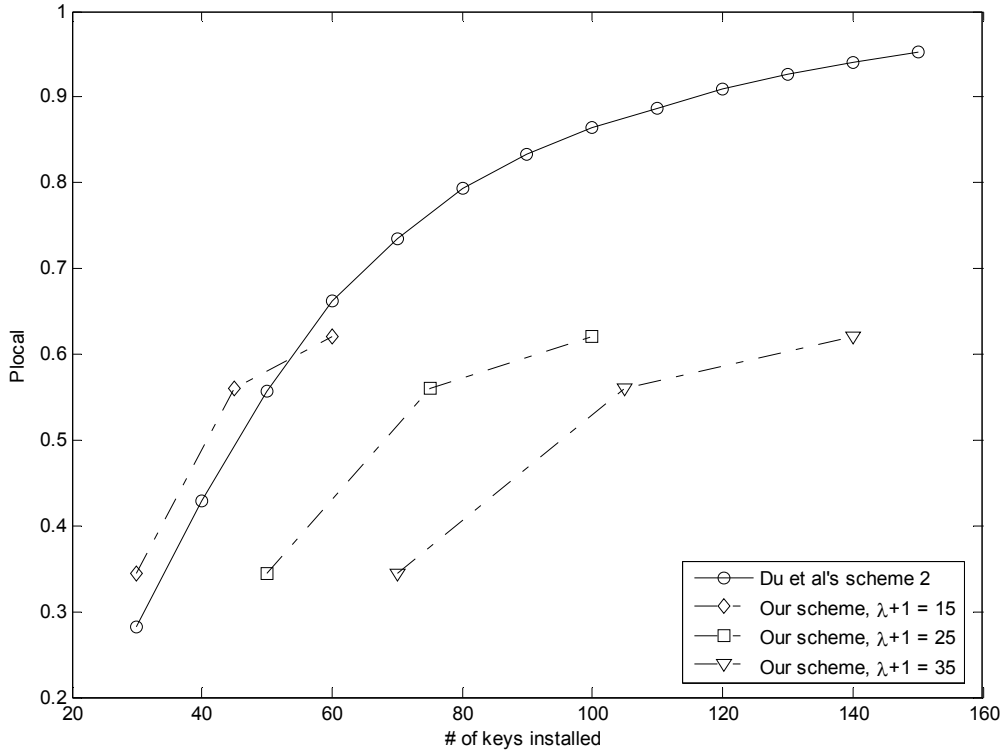

Figure 4-2 Local connectivity vs. memory usage for Du et al.'s scheme using deployment knowledge [8] and our scheme. For our scheme $\omega = 7$ and $\tau = 2, 3, 4$

Figure 4.2 shows local connectivity versus memory usage values obtained from simulation results of our scheme and Du et al's scheme using deployment knowledge [8], we will call this scheme "Du et al's scheme 2". In [8], Du et al. divide the sensor network into zones. Then, key pools of size S_c for each zone are created using a method described in [8] and randomly assign m of them to each sensor, where m is the number of keys in each sensor node. In Du et al's scheme 2, a zone's key pool is setup such that it contains keys from all neighboring zones. Their approach is a modified version of Eschenauer and Gligor's scheme [1]. They improve the scheme in [1] by using deployment knowledge. A more detailed description of key predistribution scheme in [8] can be found in Chapter 2.

For the simulations of Du et al's scheme 2, we used the following parameters; size of global key pool, $|S|$, is 100000 and the sensor network is divided into 100 zones and there are 100 nodes in each zone.

In Figure 4.2, we simulated our scheme for various values of $\lambda+1$. For our scheme, τ, number of key spaces installed in a node, and $\lambda+1$, size of a matrix share, give the number of keys in a node. For example, when $\tau=3$ and $\lambda+1=25$, number of keys is 75. When $\lambda+1$ is 15 and τ is less than 4, our scheme has better local connectivity than Du et al.'s scheme 2. However, local connectivity of our scheme does not increase more than 0.6209 and P_{local} for Du et al's scheme 2 reaches 0.9522 when number of keys are 150. Local connectivity for our scheme stops increasing after a specific value because regular nodes cannot establish direct secure links with their different-zone neighbors; they can only setup shared keys with different-zone neighbors via inter-zone path key establishment process. Whereas, in Du et al's scheme 2, nodes have the capability to share keys with nodes from neighboring zones. In our scheme, increasing $\lambda+1$ value does not change the local connectivity performance of our scheme. However, $\lambda+1$ value is one of the parameters that determines the resilience of our scheme, as we will show in the following section.

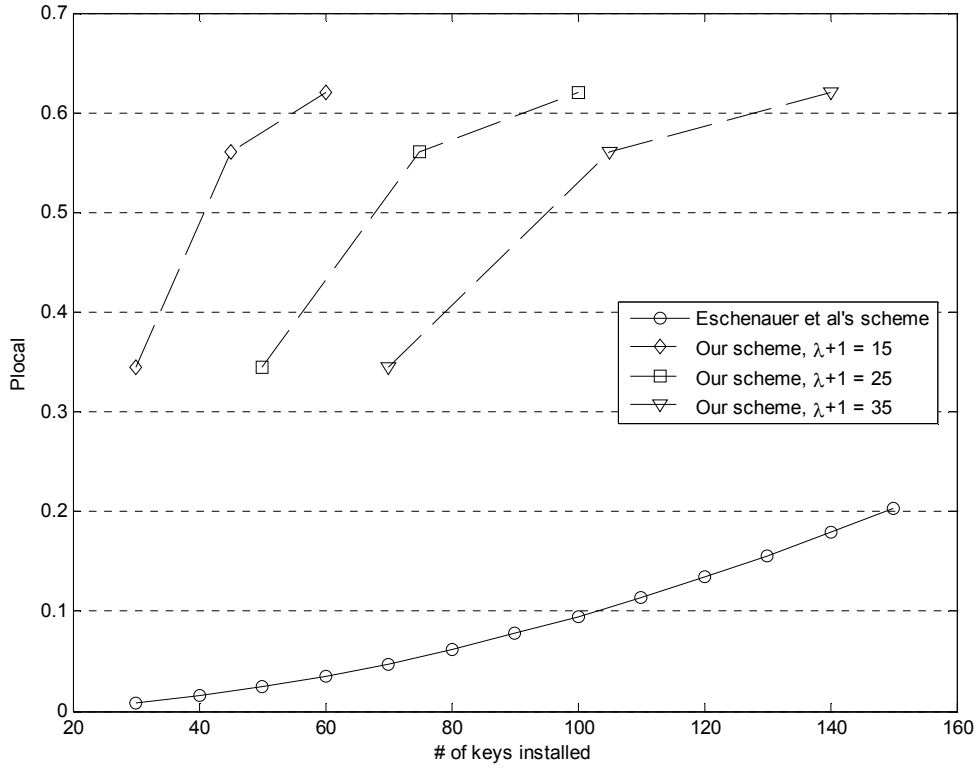

Figure 4-3 Local connectivity vs. memory usage for Eschenauer and Gligor's scheme [1] and our scheme. For our scheme, $\omega = 7$ and $\tau = 2, 3, 4$

In Figure 4.3, we show local connectivity performance of Eschenauer and Gligor's scheme [1]. Eschenauer and Gligor's scheme can be called as the basic probabilistic key predistribution scheme. Before deployment, a key pool of size S_c is setup and then m keys are randomly picked and installed into each node. In Eschenauer and Gligor's scheme, deployment knowledge is not used, so whole deployment area can be seen as a single zone. After deployment, neighboring nodes check if they share any keys and establish secure links. In our simulations for Eschenauer et al.'s scheme, we used a key pool of size 100000 and there are 10 000 nodes in the sensor network.

It can be seen in Figure 4.3 that our scheme clearly outperforms Eschenauer and Gligor's scheme for all values of $\lambda+1$. Our scheme's performance gain over Eschenauer and Gligor's scheme is mostly because our scheme leverages deployment knowledge while distributing cryptographic material to sensor nodes.

4.4 Global Connectivity

After node deployment, each node tries to establish secure links with their neighbors. As defined in Section 3.3, after direct key establishment phase, secure links form the *secure link graph* and the probability that secure link graph is connected gives the *global connectivity* as defined [1, 9]. We can compute global connectivity by finding the ratio of the largest connected block of nodes over the total number of nodes. The key predistribution scheme determines the connectedness of the key sharing graph and, therefore, the global connectivity. When sensor node distribution is uniform, i.e. a node can be anywhere in the deployment area regardless of its deployment point, we can estimate global connectivity using local connectivity and other sensor network parameters by using Erdös random graph theorem [19]. However, we use normal distribution to model our sensor node distribution, so Erdös random graph theorem is not applicable to our work. In our work, we only use simulations to estimate the global connectivity of sensor networks.

Global connectivity indicates the amount of wasted nodes. If some nodes have no connection with the main block of sensor nodes, then they cannot contribute to the sensor network. Because nodes can find shared key spaces only with same zone neighbors without the help of agent nodes, nodes far away from their deployment points may have too few or no secure links. However, these isolated nodes can still have many physical links with different zone nodes.

Figure 4.4 shows global connectivity of our scheme for τ=2, 3, 4 and ω=4, 5, 6. Simulation results indicate that even when τ = 2, more than 99% of nodes join and contribute to the sensor network. The rest can be seen as wasted sensor devices because they have no secure links with the largest body of sensor nodes.

Note that in Figure 4.4, nodes that have no other nodes in their communication range are not included in the computation of global connectivity since our key predistribution scheme has no effect on them.

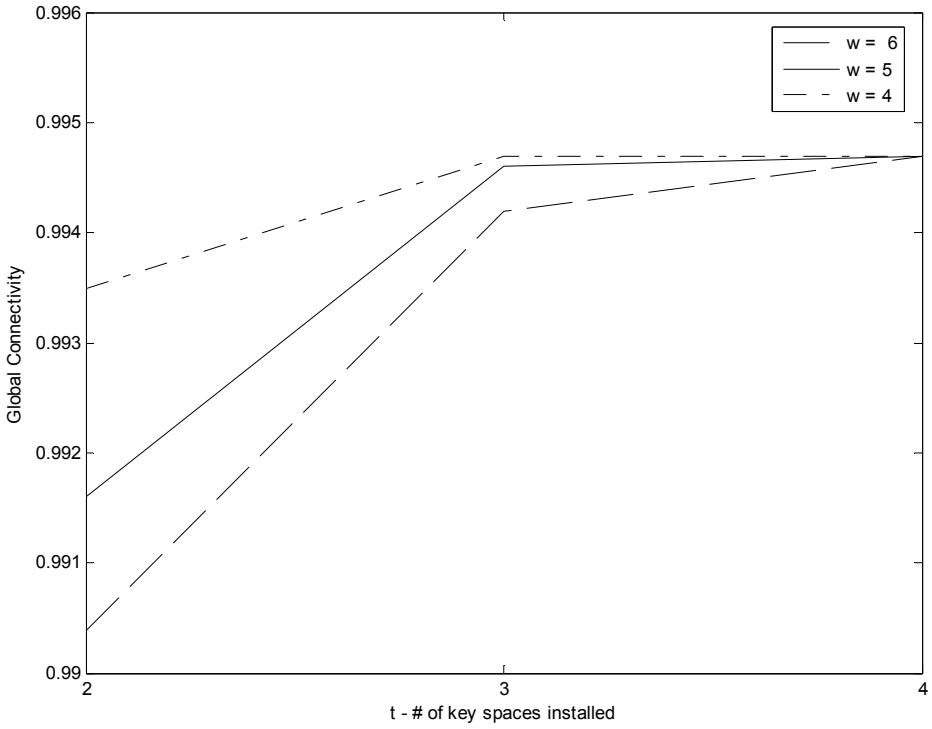

Figure 4-4 Simulation results of our scheme for global connectivity vs. τ

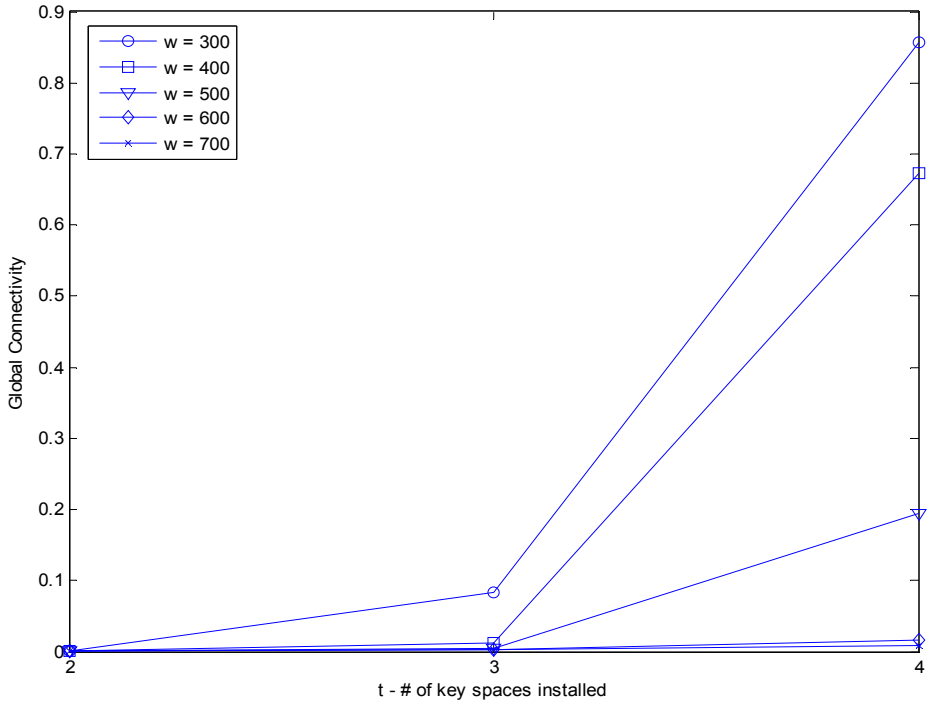

Figure 4-5 Simulation results of global connectivity vs. τ for Du et al.'s scheme [9]

Figure 4.5 illustrates global connectivity as a function of τ, number of key spaces installed in a node, for Du et al.'s scheme [9]. From the figure above, it can be observed that global connectivity has a threshold property. When τ is low, i.e. local connectivity is relatively low (see Figure 4.1), the secure link graph for Du et al.'s scheme is very shattered and global connectivity is very low. As the average number of secure links per node increases, global connectivity increases exponentially and quickly reaches high values.

When we compare our scheme with Du et al's scheme, global connectivity of our scheme is clearly higher, especially for low τ values. We simulated both our scheme and Du et al's scheme under similar conditions. Note that, the sample network we used for simulations has 100 zones. Therefore, when ω is equal n, total number of key spaces in the whole sensor network is $100n$. Total number of key spaces for both our scheme and Du et al's scheme is comparable.

4.5 Communication Cost

In this section, we investigate the communication overhead of our key predistribution scheme when two neighboring nodes cannot establish a direct secure link. After direct key establishment phase, average ratio of neighbors that could not be securely connected is $1 - P_{local}$. When a node shares at least one key space with one if its neighbors, they are obviously connected at one hop, i.e. they are one-hop neighbors. For the rest of its neighbors, a node has to make multi-hop communication in order to establish secure links. In our key predistribution scheme, a sensor network incurs most of communication cost during three operations: intra-zone path key establishment, hybrid key establishment and inter-zone path key establishment. During intra-zone path key and hybrid key establishment processes, each key is setup via flooding.

4.5.1 Intra-Zone Path Key Establishment

When two neighboring nodes from the same zone cannot find a common key space, they have to reach each other through a multi-hop communication. As we discussed in Section 3.5.1, these two nodes have to find a secure path between them. We used simulations to estimate the average number of hops required to connect these two nodes under various conditions. Figure 4.6 illustrates number of hops and connectivity of corresponding secure link graphs for various τ and ω combinations.

It can be observed from Figure 4.6 that when τ / ω ratio is high, a node can reach all of its same-zone neighbors in only one hop, i.e. can establish direct links. For example, when τ is 3 and ω is 6, a node can reach, on average, 0.9503 of its same-zone neighbors in one hop, and the rest in two hops. Note that, while calculating communication overhead in our simulations, we ignored nodes that have no secure links, because in order to establish path keys, nodes must have some secure links beforehand.

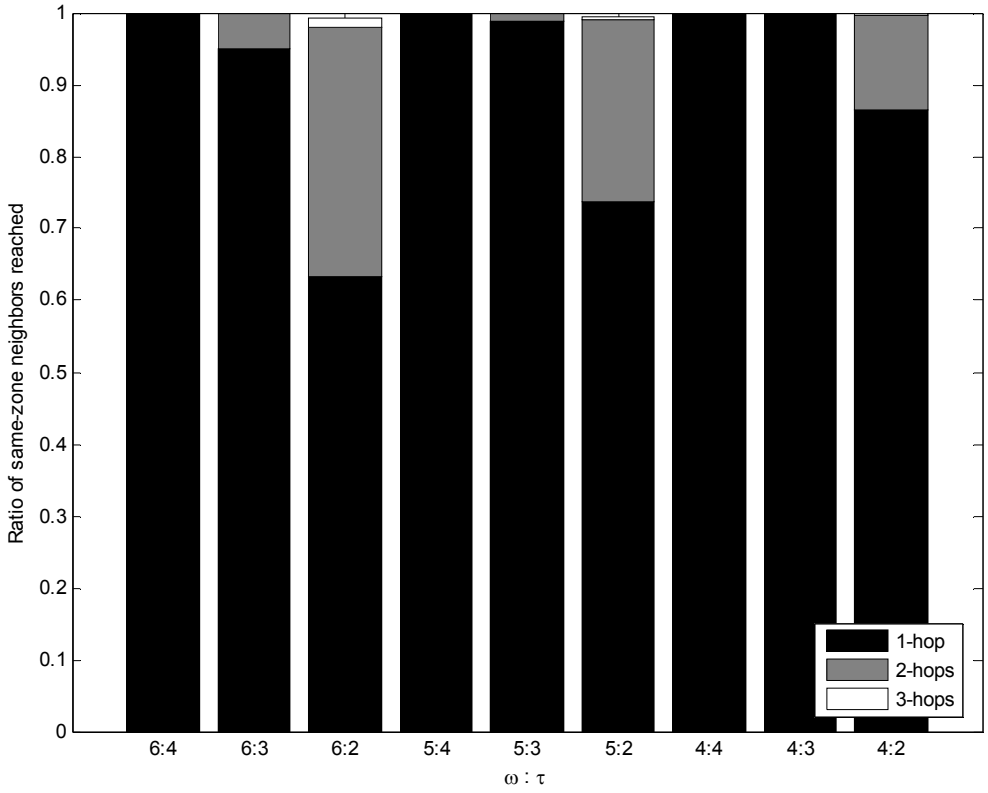

Figure 4-6 Communication overhead for intra-zone path key establishment

4.5.2 Hybrid key establishment

In our scheme, it is crucial for regular nodes to share a hybrid key (see Section 3.4) with an agent node in order to perform inter-zone key establishment process. When a node wants to establish a secure link with a node from a different zone, it sends a message to its nearest agent node (nearest in terms of number of hops over a secure path) over a path of secure links and then the agent node sends back a message through the same path. A detailed description of this process can be found in Section 3.5.2, inter-zone path key establishment process. Therefore, the number of hops, a regular node can reach its nearest agent node, is an important indicator of network connectivity and an important parameter in overall communication cost. We show in Figure 4.8 that majority of regular nodes can reach their nearest agent nodes in two hops when the number of agent nodes in a zone is five, i.e. $A_z = 5$.

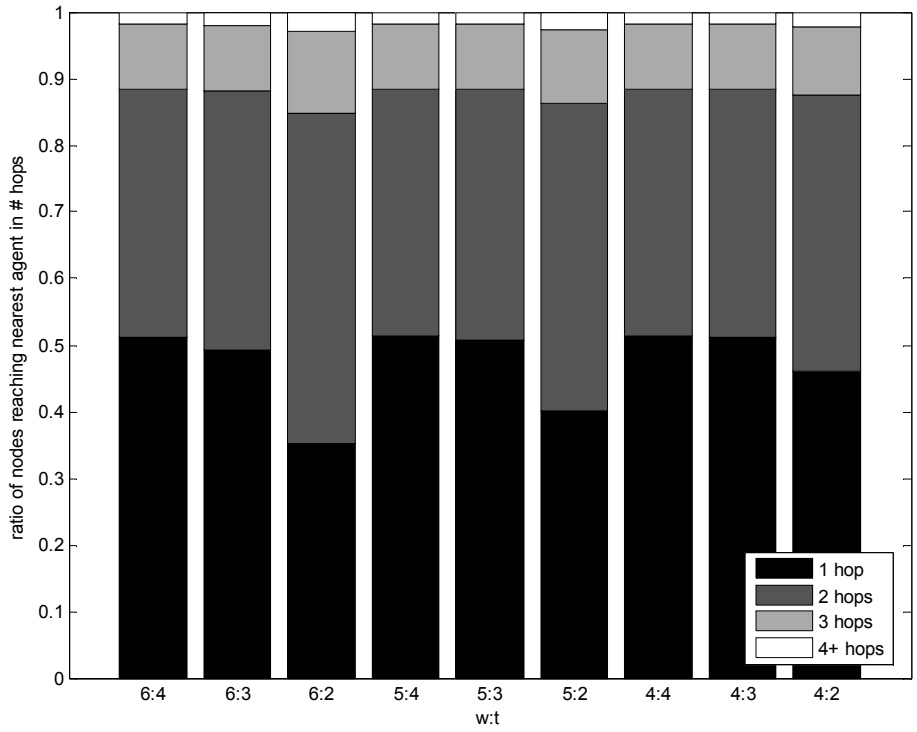

Figure 4-7 Ratio nodes reaching their nearest zone agent in *i* hops when A_z =2, *i*=1,…,4+

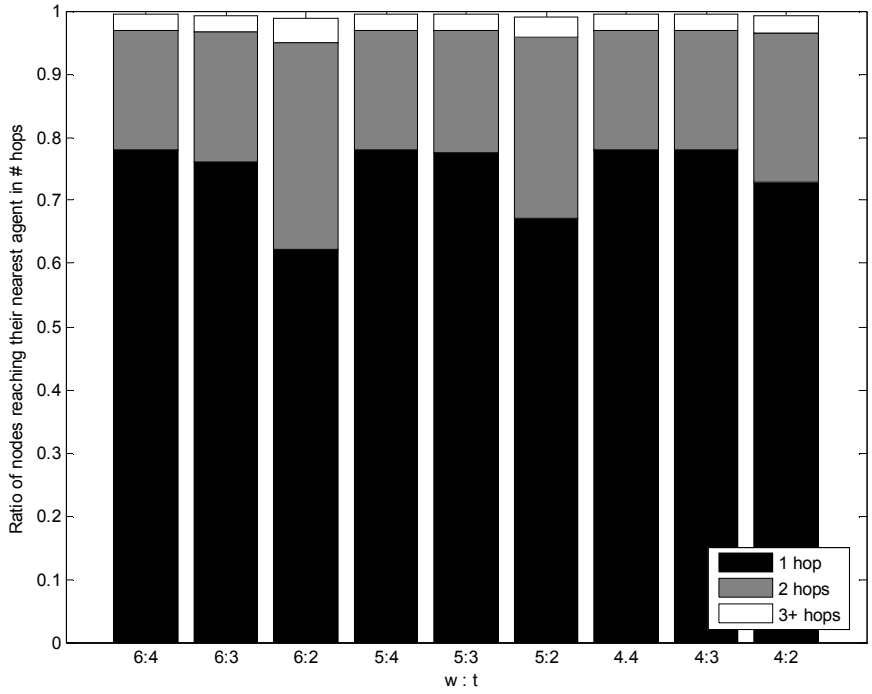

Figure 4-8 Ratio nodes reaching their nearest zone agent in *i* hops when A_z =5, *i*=1,…,3+

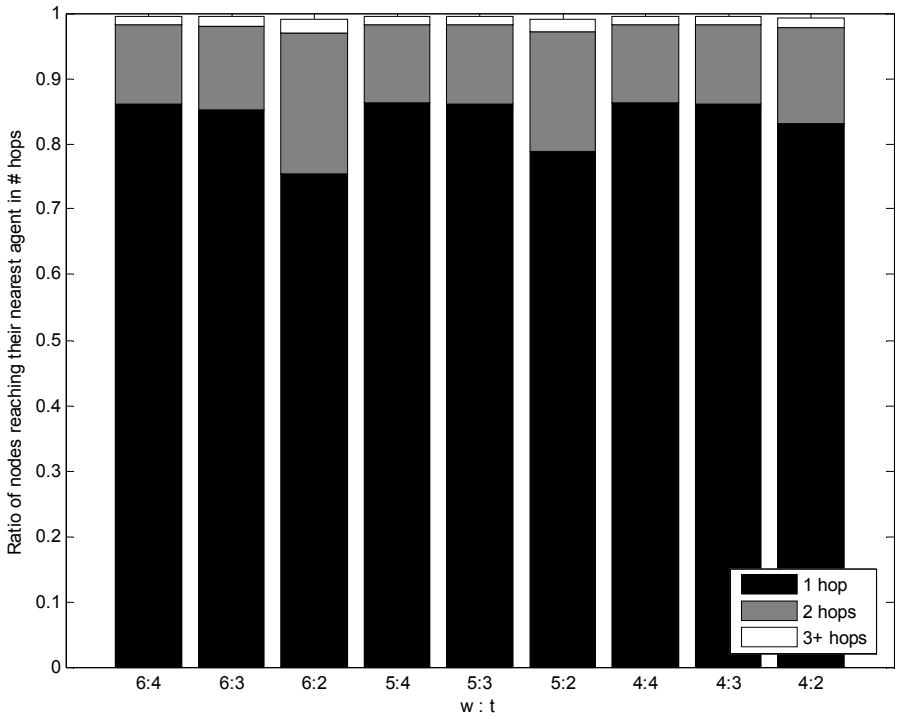

Figure 4-9 Ratio nodes reaching their nearest zone agent in *i* hops when A_z =8, *i*=1,…,3+

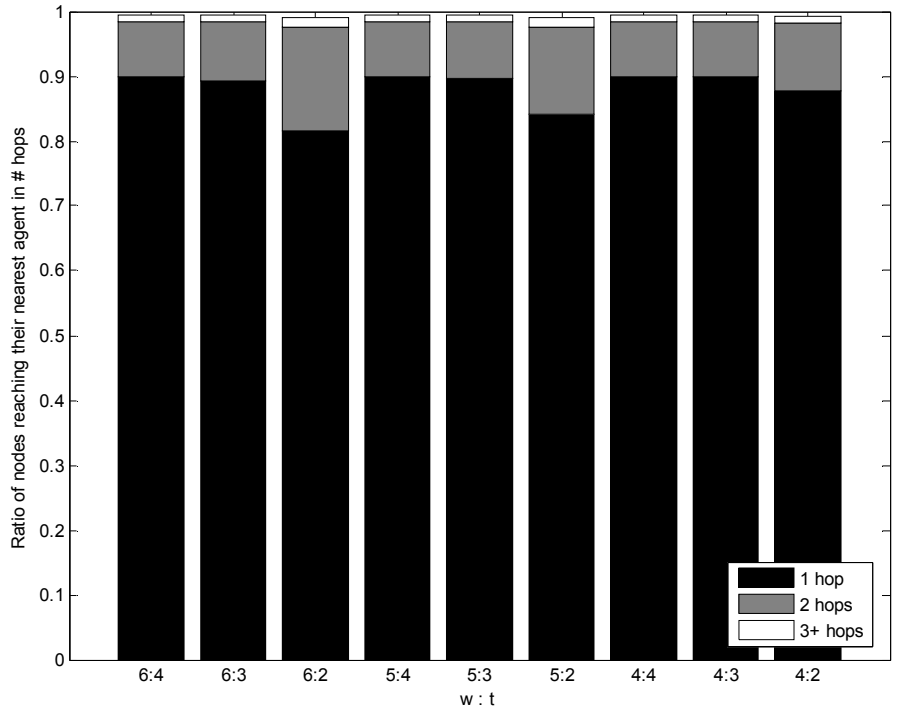

Figure 4-10 Ratio nodes reaching their nearest zone agent in *i* hops when A_z=10

From the figures above, it can be seen that even with limited number of agent nodes, any node can reach their nearest zone agent in at most 3 hops. Note that, in order to find a secure path to a zone agent, a regular node uses the flooding technique as explained in Section 3.4. This method can cause large number of messages. However, in our scheme, flooding technique is used only for establishing path keys with same-zone neighbors and while finding a secure path to nearest zone agent. While a regular node is establishing a hybrid key with a zone agent, flooding is required only for one time. A secure path to nearest zone agent is found only once. Then the same path can be used for all subsequent inter-zone path key establishment processes.

During hybrid key establishment processes, agent nodes have to carry out more communication than regular nodes, because an agent node is usually responsible for multiple regular nodes. For example, when the number of agent nodes in a zone is 5, on average 19 nodes use one agent node. For each of these 19 nodes, an agent node performs the hybrid key establishment method. Assuming a message containing τ key space indices and node ID is 64 bits long and the cost of transmitting a bit over radio is 0.01μJ [54], we show cost of communication for one agent node during hybrid key establishment processes in Table 4.1. Note that an AA type NiMH battery can produce 11 kJ energy, on average. Thus, hybrid key establishment process does not incur significant amount of energy cost on an agent node carrying two AA type batteries.

Table 4-1 Energy consumption of an agent node during hybrid key establishment

A_z	Energy consumption (μJ)
2	31,36
5	12,16
8	7,36
10	5,76

4.5.3 Inter-Zone Path Key Establishment

When two neighboring regular nodes are from different zones, they try to establish a secure link by inter-zone path key establishment process, as discussed in Section 3.5.2. In this process, regular nodes communicate with both their zone agents and each other. Regular nodes send encrypted messages to their agent nodes over a secure path or a direct link. Then, agent nodes send back messages over the same path.

Assuming that one of the regular nodes is h_1 hops away from its zone agent and hop length between the other node and its zone agent is h_2, total number of messages exchanged during inter-zone path key establishment process can be found as:

$$2(h_1 + h_2) + 3 \tag{4.1}$$

This process incurs most of the communication cost on the sensor network, because every neighboring node pair from different zones has to establish secure links using this process. Our simulations show that average number of different-zone neighbors of a node is 18.0226 and average number of physical neighbors is 48.8934. Table 4.2 shows number of inter-zone path keys, i.e. number of different-zone neighbor pairs, for different A_z values, number of agent nodes in a zone. Values in Table 4.2 are obtained from simulations on the 10000 nodes sensor network described in Section 4.2. Although there are seemingly many inter-zone path keys, each process usually involves more than two nodes. Regular nodes, near the center of zones, also contribute to inter-zone path key establishment process by acting as possible intermediaries between zone agents and owners of path keys.

Table 4-2 Communication cost

A_z	2	5	8	10
# of inter-zone path keys	88379	85599	82840	81118

There is a cost of each inter-zone path key establishment process to the sensor network. Through simulations, we calculated the number of protocol messages exchanged for each inter-zone path key. Figures 4.11, 4.12 and 4.13 illustrate the ratio of inter-zone path keys established versus number of messages exchanged for various ω and τ values.

For example, when $\omega = 6$, $\tau = 3$ and $A_z = 5$, 83 % of all inter-zone path keys are established by exchanging 9 or less protocol messages. Maximum number of messages required in order to establish all inter-zone path keys is 13 when $\omega = 6$, $\tau = 3$ and $A_z = 10$.

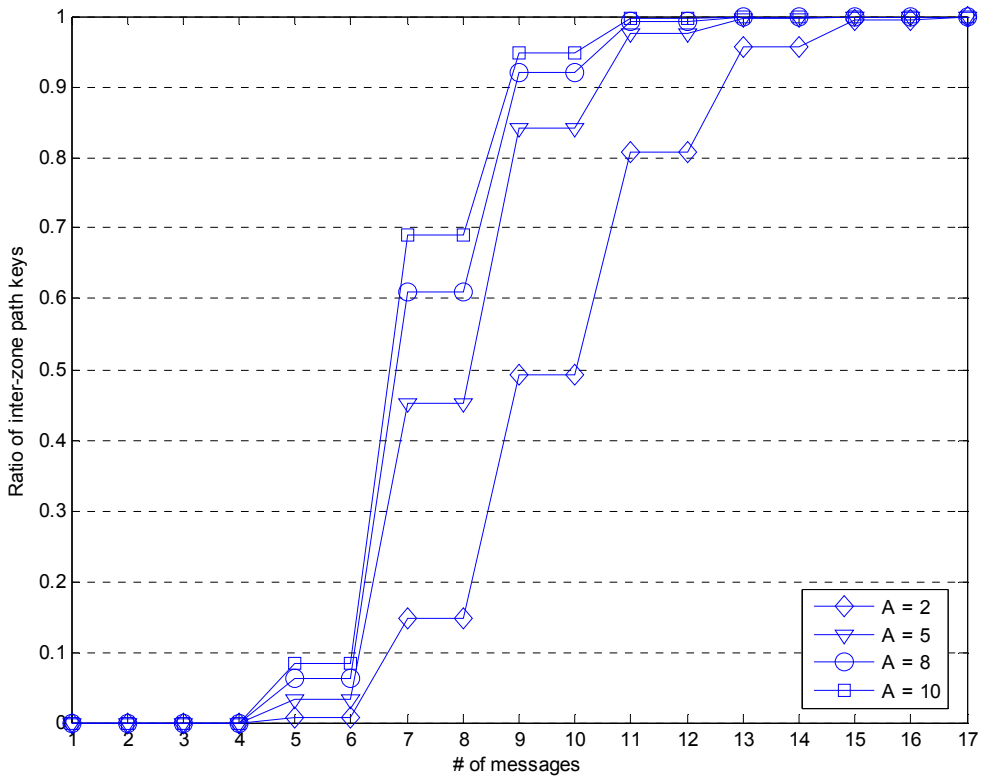

Figure 4-11 Communication cost for inter-zone path key establishment, $\omega = 6$ and $\tau = 4$

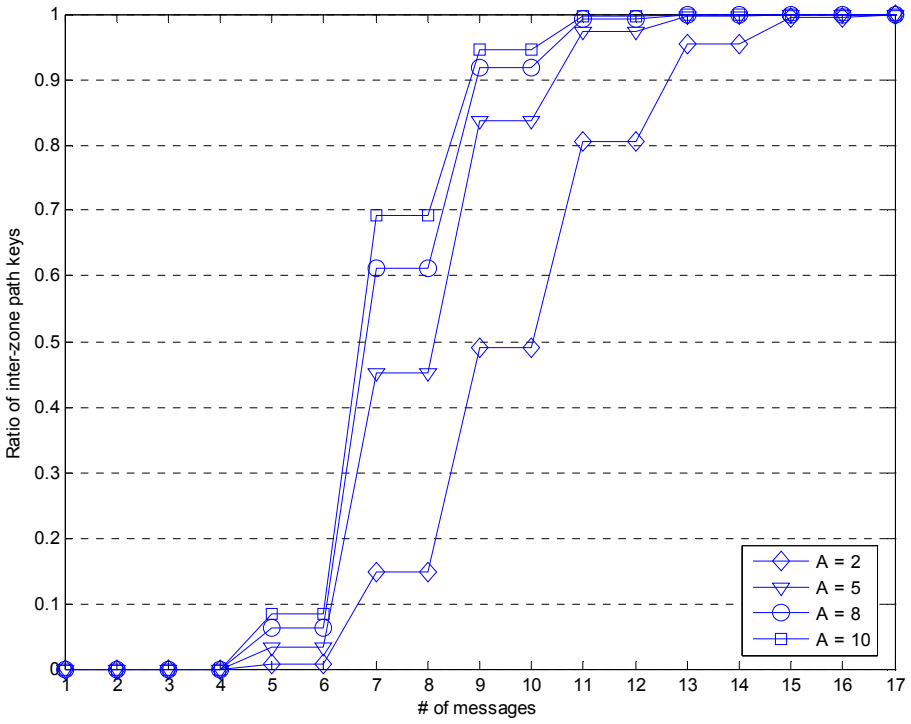

Figure 4-12 Communication cost for inter-zone path key establishment, $\omega = 6$ and $\tau = 3$

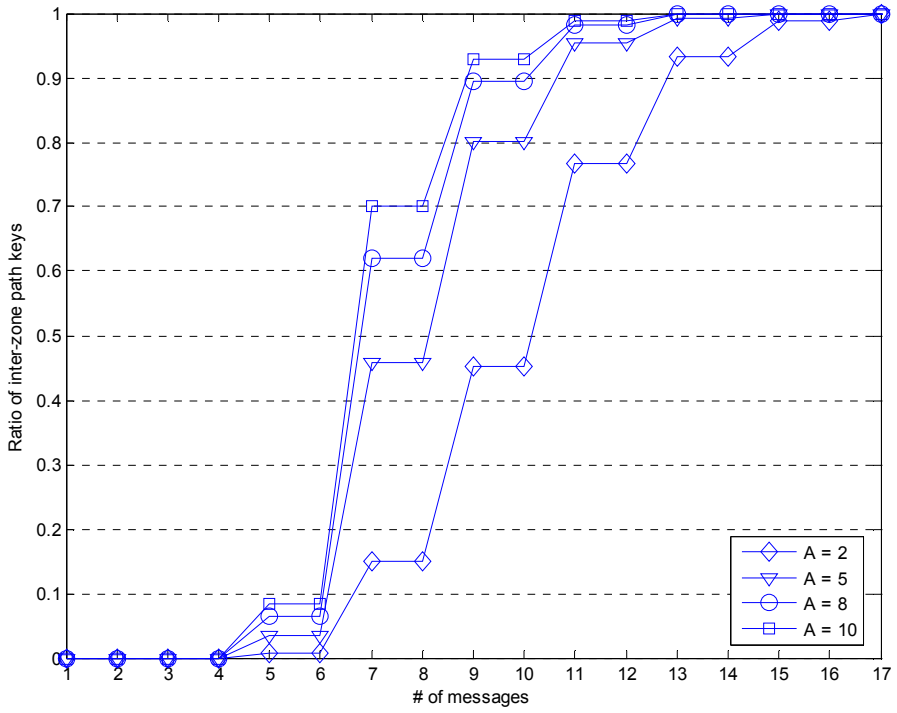

Figure 4-13 Communication cost for inter-zone path key establishment, $\omega = 6$ and $\tau = 2$

It can be observed from the figures above that increasing A_z, number of agent nodes in a zone, decreases distance between regular nodes and their nearest agent nodes and consequently, decreases number of messages exchanged. The most dominant factor that affects communication cost for inter-zone path key establishment process is the hop distance between regular nodes and their nearest zone agents.

Results from Figures 4.11, 4.12 and 4.13 comply with results from Figures 4.7, 4.8, 4.9 and 4.10. Figures 4.8, 4.9 and 4.10 show that for A_z =5, 8, or 10, majority of nodes can reach their nearest agent node in only one hop. Based on Equation 4.1, we can expect that a quick rise in ratio of inter-zone path keys after number of messages reach 7 since $2(1+1)+3=7$. It can be seen from Figures 4.11, 4.12 and 4.13 that less than 10 percent of inter-zone path keys are established with 5 messages; however, there is steep increase when number of messages reach 7, especially for high A_z values. For example, the ratio of established inter-zone path keys reaches 0.7 for A_z=10 when number of messages reach 7.

Note that there is only a slight difference between Figure 4.11 and Figure 4.12. Decreasing τ, results in lower local connectivity. However, it can be seen from Figures 4.7, 4.8, 4.9 and 4.10 that decreasing τ from 4 to 3 does not change the average hop distance between regular nodes and their nearest agent nodes much. That is the reason behind why lowering local connectivity in Figures 4.11 and 4.12 does not affect the ratio of inter-zone path keys too much.

During inter-zone path key establishment, an agent node may be intermediary for multiple regular nodes. During a inter-zone path key establishment process, an agent node transmits only one message. However, an agent node has to contribute to all inter-zone path key establishment processes of all its regular nodes. For example, when A_z is 5, number of regular nodes in a zone is 95, so, on average, 19 regular nodes use an agent node. Our simulations show that each of these regular nodes has 17 different-zone neighbors. At each inter-zone path key establishment process, an agent node sends a message containing a

secret key and encrypted version of that key, and we assume that the transmitted message is 160 bits long. The cost of transmitting a bit over radio is 0.01 μJ [54]. In Table 4.3, we show the energy consumed by an agent node while transmitting messages for inter-zone path key establishment processes. Note that an AA type NiMH battery can produce 11 kJ energy, on average. Thus, inter-zone path key establishment process does not incur significant amount of energy cost on an agent node carrying two AA type batteries.

Table 4-3 Energy consumption of an agent node during inter-zone path key establishment of all its regular nodes

A_z	Energy consumption (μJ)
2	1385,8
5	520,4
8	304,8
10	233,6

4.5.4 Flooding

When flooding method is used during establishment of a pairwise key, a node broadcasts a query and each neighboring node forwards this query to all their next hop neighbors. Thus, during flooding, excessive number of messages are transmitted. Apparently, flooding incurs heavy communication cost on the sensor network. In our scheme, number of intra-zone path keys and hybrid keys determine the number of flooded messages. In Figure 4.14, Figure 4.15, Table 4.4 and Table 4.5, we show number of different types of keys established in our scheme and in Du et al's scheme 2.

In our scheme, only intra-zone path keys and hybrid keys are established via flooding. In Figure 4.14 and Table 4.4, sum of intra-zone path keys and hybrid keys are shown as "keys using flooding". The values in Table 4.4 and Figure 4.14 are results from simulations performed using parameters described in Section 4.2. Communication cost of establishing other keys, direct keys and inter-zone path keys, are less than establishing keys using flooding. It can be seen that keys established via flooding cover only a small ratio of

all keys. As ω / τ ratio increases, less flooding is used. In Table 4.4, it can be observed that by increasing A_z, we can decrease the number of hybrid keys established.

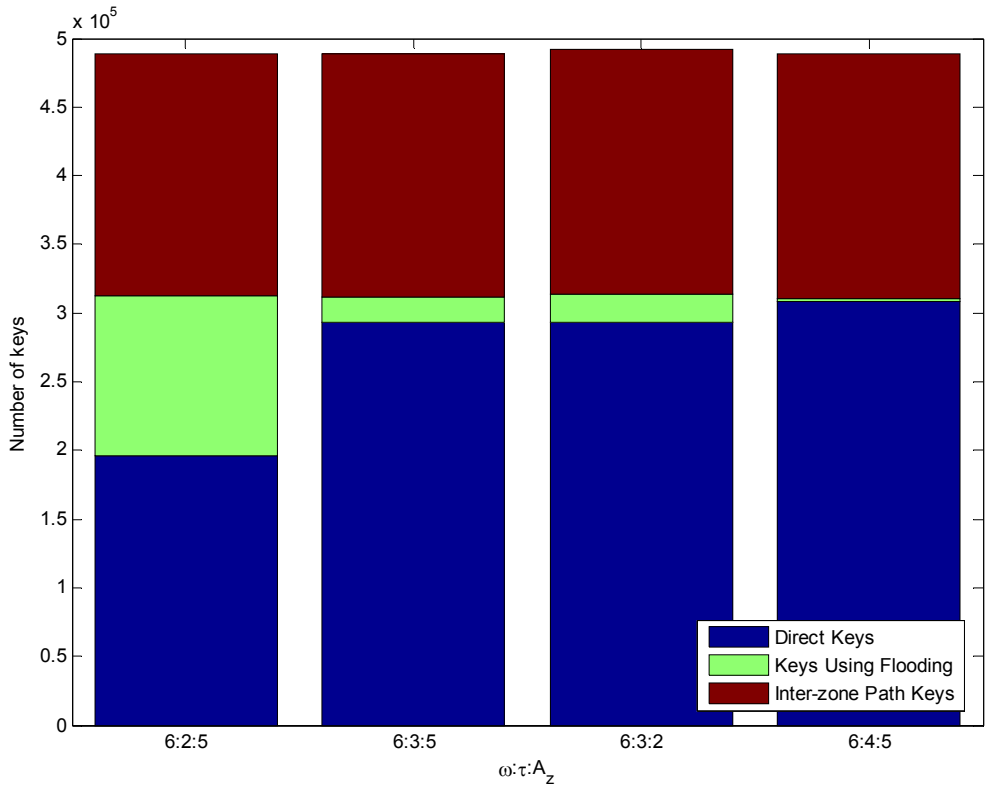

Figure 4-14 Number of keys established in our scheme. "Keys Using Flooding" shows total number of intra-zone path keys and hybrid keys.

Table 4-4 Number of keys established in our scheme for various ω : τ : A_z values

ω:τ:A_z	Direct Keys	Keys using flooding (Intra-zone path keys + hybrid keys)	Inter-zone Path Keys	Total Keys
6:2:5	195596	116754 (114623+2131)	176368	488718
6:3:5	293542	17511 (15344+2167)	177899	488952
6:3:2	293542	20219 (15372+4847)	177899	491660
6:4:5	308354	2527 (358+2169)	178045	488926

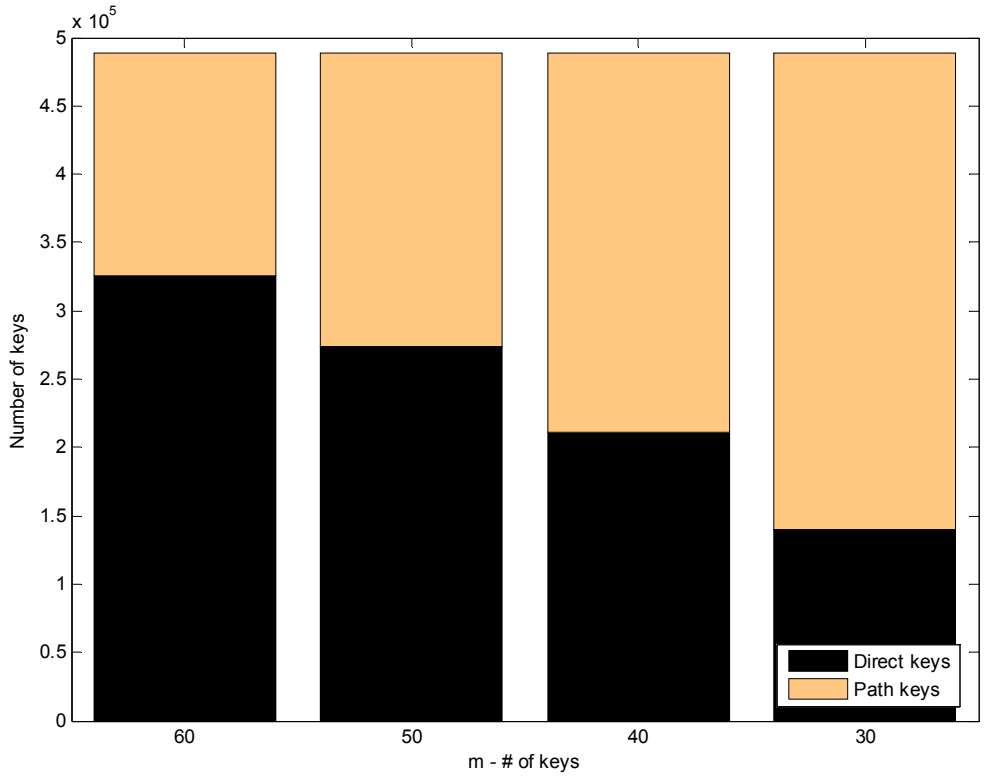

Figure 4-15 Number of keys established in Du et al's scheme 2. Path keys use flooding.

Table 4-5 Number of keys established in Du et al's scheme 2 for various *m* values

	Direct Keys	Path Keys (use flooding)	Total Keys
60	325320	163549	488869
50	274138	214729	488867
40	211530	277389	488919
30	139366	349410	488776

In Du et al's scheme 2, path keys are established via flooding. In Figure 4.15 and Table 4.5, we present simulation results for a sensor network with 10000 nodes using Du et al's scheme 2. As *m*, number of keys stored in a node, increases, number of direct keys increases and, thus, less flooding is required. It can be observed from Figure 4.15 and Table 4.5 that, when *m* is less than 50, majority of the pairwise keys are path keys. Thus, majority of the keys are established using flooding.

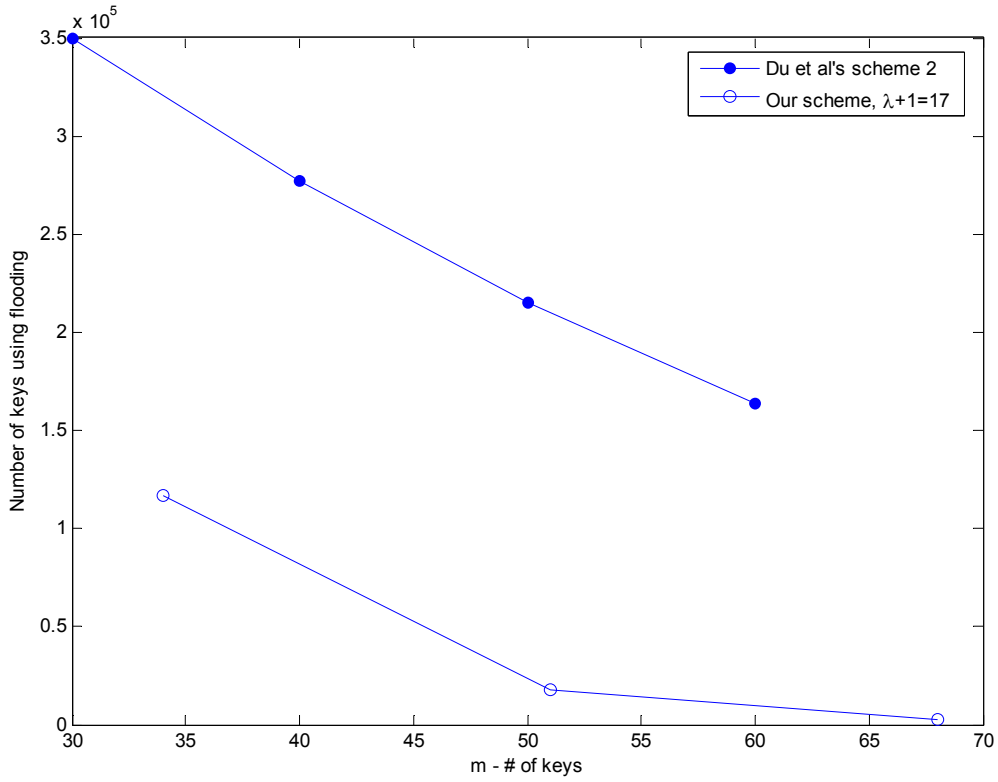

Figure 4-16 Number of keys established using flooding in our scheme and Du et al's scheme 2

In Figure 4.16, we compare use of flooding in our scheme versus Du et al's scheme 2. In our scheme, number of keys, m, is calculated as $m = (\lambda + 1)\tau$. In Figure 4.16, we took λ+1 as 17, which is the value used in node capture resiliency simulations, and showed the change of number of keys using flooding with respect to τ values 2, 3 and 4 (that corresponds to m values 34, 51, and 68).

From Figure 4.16, it can be seen that for all m values, our scheme uses less flooding than Du et al's scheme 2. Table 4.4 and Table 4.5 show that total number of keys in both schemes are approximately equal. Thus, the ratio of keys established via flooding in our scheme is less than that of Du et al's scheme 2. The rest of the keys are either direct keys or inter-zone path keys in our scheme and the rest of the keys are direct keys in Du et al's scheme 2. Establishment of inter-zone path keys causes more communication cost than

direct keys. However, they do not use flooding, so they are cheaper than path keys in Du et al's scheme 2.

As a result, although our scheme has inter-zone path keys, which has additional communication cost as compared to direct keys, our scheme uses less flooding than Du et al's scheme 2.

4.6 Resiliency Against Node Capture

The most obvious attack against a sensor network is capturing sensor nodes. Therefore, we will investigate the effects of node capture attack on direct key establishment, as done in other key predistribution schemes, such as [7], [8], [9], [16] and [20]. We will assume when a node is captured, all of its cryptographic material is compromised. Using those compromised material, attacker can also compromise some additional links. A key distribution scheme's resiliency against node capture can be defined as the ratio of additional compromised links over total number of links except those of captured nodes. Since, there is no way to prevent an attacker from capturing nodes and gaining access to their cryptographic material, we do not include secure links of captured nodes into our resiliency calculations.

In our scheme, at the intra-zone key predistribution phase, we store matrix shares into each sensor node. For a key space to be compromised, $\lambda+1$ nodes carrying shares from that key space must be compromised as in [2, 9]. An attacker with λ shares from the same matrix cannot gain any extra information about that key space and cannot learn private shares of nodes that are not captured.

When an attacker captures an agent node, attacker can compromise both its matrix shares and its pairwise keys, shared with other agent nodes. When an agent node is captured, attacker can gain information about only that node's secure links, because there

are only two copies of a pairwise key, and a pairwise key can be used only in one secure link. In other words, pairwise keys have perfect resiliency against node capture.

While analyzing the node capture resiliency of a key predistribution scheme, it is commonly assumed that attacker does not record encrypted messages, [7, 8, 9, 16, 20]. Considering that sensors are deployed on a very wide area, it is impractical for an attacker to eavesdrop all communication and record them. A sensor node may contribute to some hybrid key or path key establishment processes. Because of this assumption, when attacker captures that node, attacker cannot learn any of hybrid or path keys by using that node's keys.

In Figure 4.17, we show node capture resiliency of our scheme, Du et al's scheme 2 [8] and Du et al's scheme [9]. While computing resiliency performances, we simulate key predistribution schemes with the similar P_{local} and memory usage values. For our scheme, ω=7, τ=3, λ+1=17 and P_{local} =0.5605. For Du et al's scheme 2, m=50, S_c=1000 and P_{local}=0.5569. For Du et al's scheme, ω=43, τ=4, λ+1=13 and P_{local} =0.56.

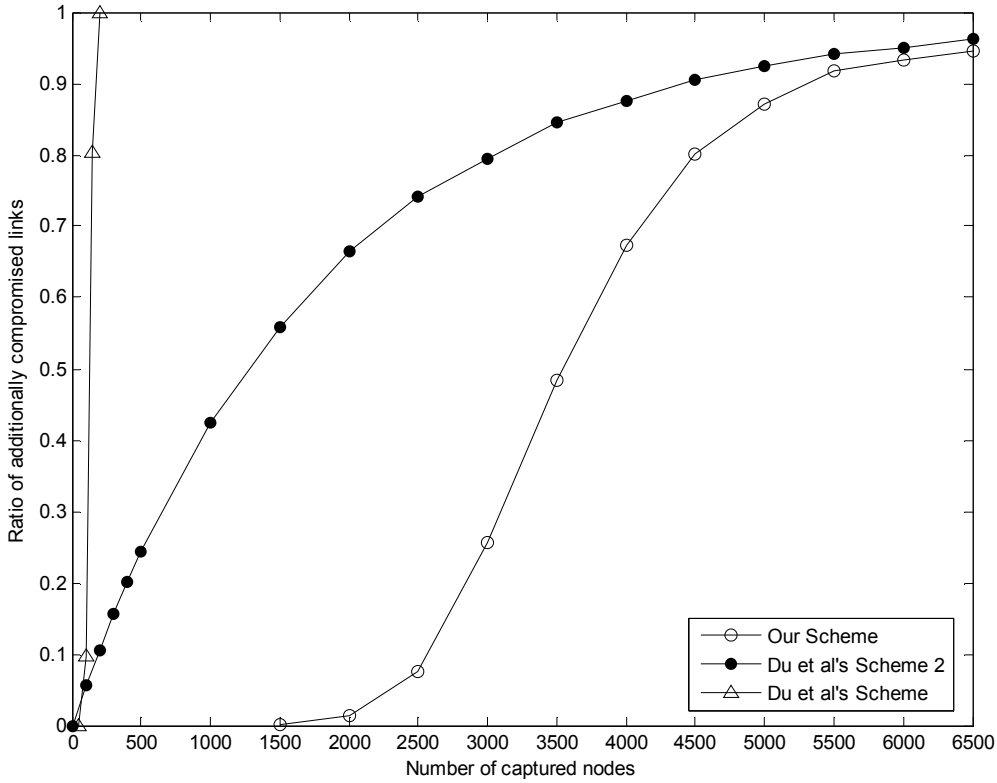

Figure 4-17 Ratio of additionally compromised links when # of nodes are captured for our scheme, Du et al's scheme 2 [8] and Du et al's scheme [9]. P_{local}=0.56.

In Figure 4.18, we show node capture resiliency of our scheme, Du et al's scheme 2 [8] and Du et al's scheme [9]. For our scheme, ω=7, τ=2, λ+1=17 and P_{local}=0.3444. For Du et al's scheme 2, m=34, $|S|$=100000 and P_{local}=0.3412. For Du et al's scheme, ω=11, τ=2, λ+1=17 and P_{local}=0.3445.

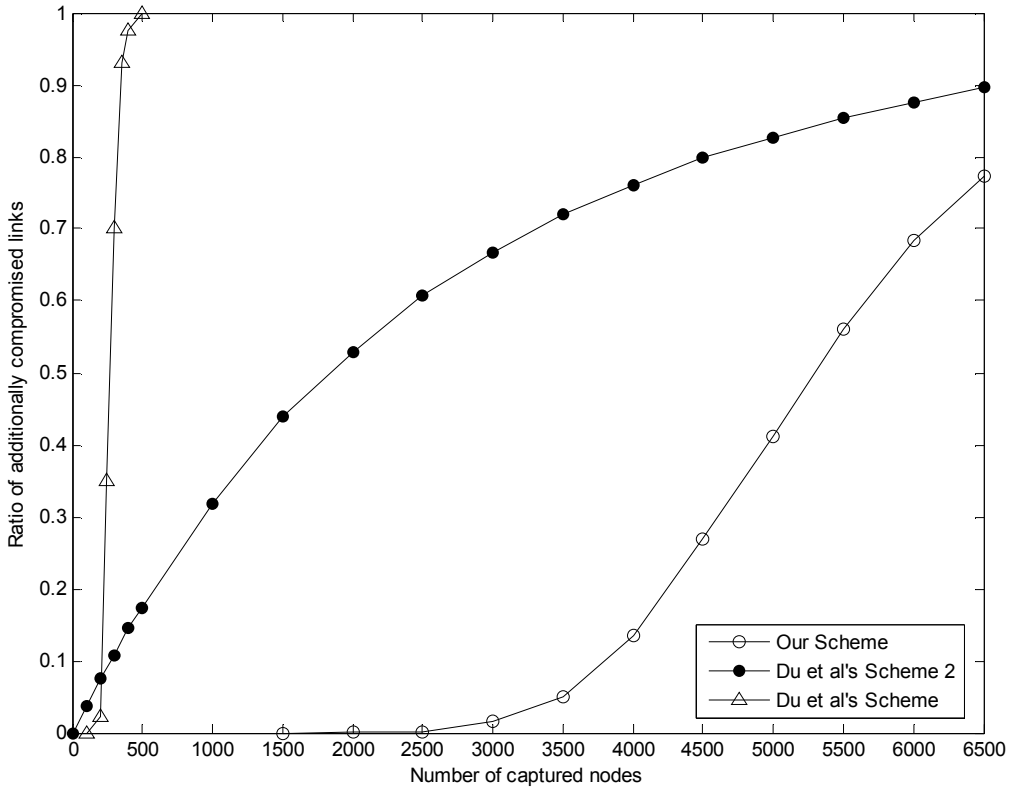

Figure 4-18 Ratio of additionally compromised links when # of nodes are captured for our scheme, Du et al's scheme 2 [8] and Du et al's scheme [9]. P_{local}=0.34.

It can be observed from Figure 4.17 and Figure 4.18 that both Du et al's scheme 2 and our scheme have substantially better resiliency than Du et al's scheme [9]. The most important reason for such a difference is that scheme in [9] does not utilize deployment knowledge and treats the whole sensor network as a single zone and thus, has to lower ω in order to achieve high local connectivity. When ω is low, same key spaces have to be repeated frequently in a 10000-node sensor network. When P_{local} is 0.34, attacker can compromise two matrix shares for each captured node and this lead to quick compromise of the whole network; by capturing only 500 nodes, an attacker can compromise all 11 key spaces. Whereas, Du et al's scheme 2 and our scheme use deployment knowledge and deployment area is divided into 100 zones. In our scheme, when ω is 6, there are 600 different key spaces in the whole sensor network.

When ratios of additionally compromised links are compared between Figure 4.17 and Figure 4.18, it can be seen that decreasing local connectivity, improves node capture resiliency for all three analyzed schemes. For our scheme, we reduce the keying information revealed at each captured node by decreasing τ from 3 to 2.

Our scheme has better resiliency than Du et al's scheme 2 in general. Especially, when number of captured nodes is less than 2000 and P_{local} is 0.56, our scheme causes zero or negligible number of additionally compromised links. In order to decrease P_{local} to 0.34 for our scheme, we dropped τ to 2 and kept $\lambda+1=17$. This way, even when 3000 nodes are captured, negligible amount of additional links are compromised. However, when $P_{local}=$ 0.56, in Du et al's scheme 2, an adversary can compromise 62 percent of secure links by capturing only 2000 nodes. Our scheme has stronger resiliency against small-scale attacks. The reason is that our scheme is based on Blom's scheme and in Blom's scheme an attacker can gain no information on a key space with less than $\lambda+1$ shares. Therefore, attacker must capture a substantial number of nodes before compromising any additional links. However, with Du et al's scheme 2, when an attacker captures only one node, he can start to compromise additional secure links where copies of captured node's keys are used.

It can be observed from Figure 4.17 that after an adversary starts to capture more than 2500 nodes, number of compromised links rises quickly for our scheme. Only after 5500 captured nodes, our scheme catches up with Du et al scheme 2 in terms of compromised links. Our scheme has slightly better node capture resiliency even when number of captured nodes is high.

4.7 Resiliency Against Node Fabrication and Wormhole Attacks

In a node fabrication attack, an adversary can capture a few nodes and use the compromised cryptographic material to fabricate new sensor nodes with identities of uncompromised sensor nodes or to fabricate sensor nodes with new identities [16]. Then, the adversary can deploy these fabricated nodes with new identities into any part of the

sensor network or deploy fabricated nodes with identities of uncompromised nodes into the parts of the network where original nodes are not present.

Node fabrication attack is a serious threat to key predistribution schemes like [1] and [8] because there is no connection between ID of sensors and their keys. In schemes [1] or [8], uncompromised nodes cannot detect fabricated nodes since fabricated nodes carry legitimate keys captured from other nodes. For example, assume that in Eschenauer and Gligor's scheme [1] each node carries m keys. If an attacker captures only two nodes, he can easily fabricate $\binom{2m}{m}$ new nodes and put these new nodes into the sensor network without being detected.

In our scheme, each sensor node's ID is linked with its matrix share, as explained in Section 2.6.3. When two nodes that share a key space want to generate a pairwise key, they only need each others' ID; IDs are used to identify the row of secret matrix A and the column of public matrix G. Consequently, an adversary cannot use matrix shares of captured nodes to fabricate nodes with new IDs.

In addition, our scheme provides a partial resiliency against wormhole attacks. In a wormhole attack, an adversary tunnels messages received in one part of the network over a low latency link and replays them in a different part. Wormhole attack is more effective if one end of the wormhole is near the sink and the other end is as far as possible. For more detailed information on wormhole attacks, see Section 2.5.7.

In our scheme, we use a zone-based approach, as explained in Section 3.1. Nodes can establish secure links only with other nodes from the same zone or neighboring zones. A pair of nodes cannot establish a secure link, if they are two or more zones away from each other. Therefore, an adversary can successfully create a wormhole only if both ends of the wormhole are in the same zone or in neighboring zones. If the wormhole extends for three or more zones, nodes at different ends of the wormhole cannot communicate with each other and as a result, the wormhole will not be functional. Thus, in our scheme, only short wormholes can be created but they can do only limited harm to the sensor network. In our

scheme, creation of longer wormholes, which can have severe affects on the sensor network, is prevented.

4.8 Scalability

Scalability is an important performance factor. In a scalable key predistribution scheme, size of the sensor network could be increased without incurring any significant additional overhead and without changing any critical performance measures. In this analysis, we increase the number of zones in our simulated sensor network from 100 to 1000 and we keep *N*, size of each zone, fixed. In this way, total number of nodes in the sensor network increase from 10000 to 100000.

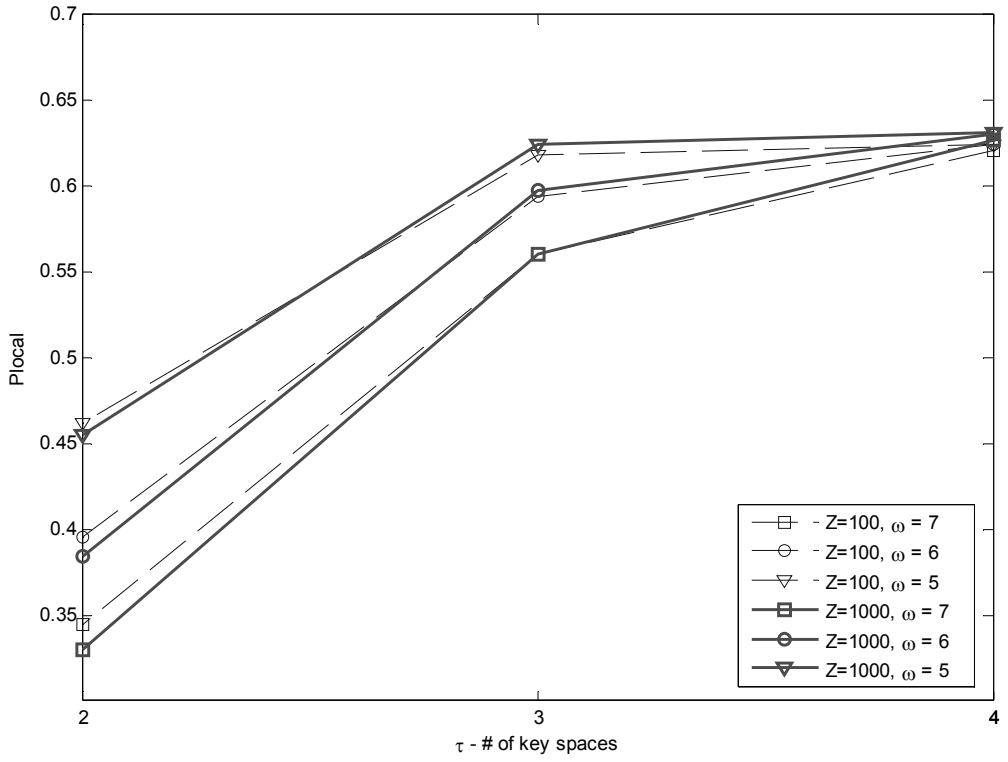

Figure 4-19 Local connectivity for our scheme when Z =100 and Z =1000

We performed simulations of our scheme with the deployment region divided into a 40 × 25 grid and the rest of the parameters are the same as the parameters given in Section

4.2. Local connectivity and global connectivity of our scheme when Z, number of zones, is equal to 1000 and 100 are plotted in Figure 4.19 and in Figure 4.20, respectively. Results show that connectivity of our scheme does not change significantly when the sensor network grows. When the number of zones is increased ten times, the ratio of secure neighbors a node can reach stays almost the same. Thus, adding new zones to the sensor network does not increase the memory cost on the sensor nodes.

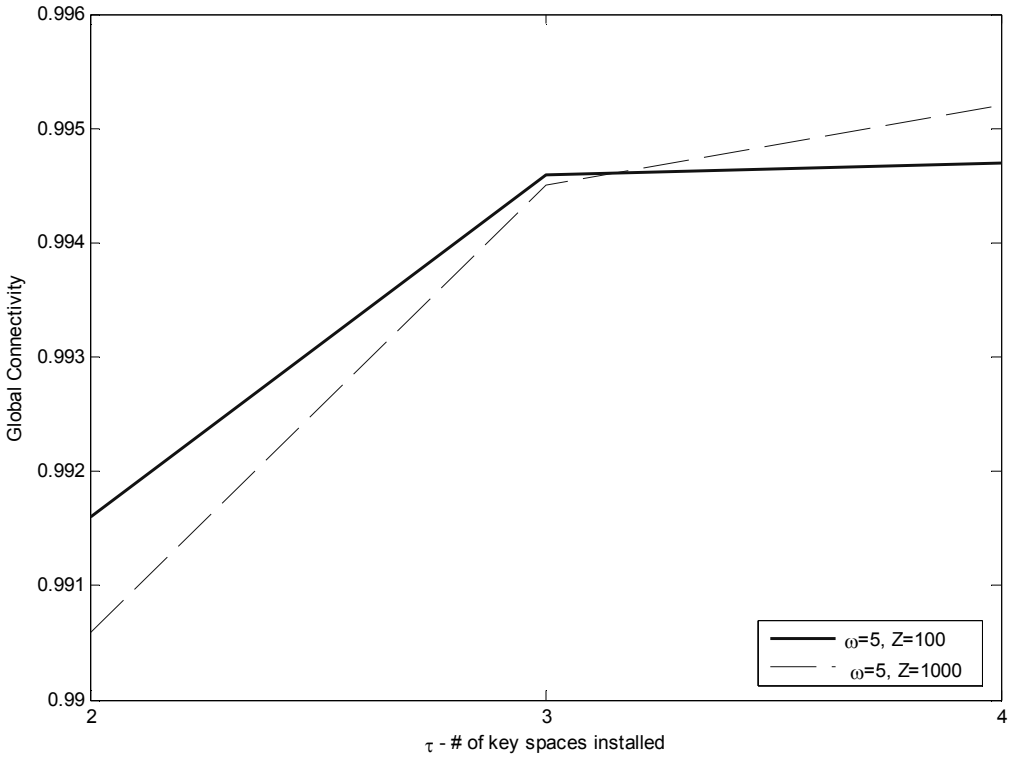

Figure 4-20 Global connectivity of our scheme when Z=100 and Z=1000.

In Figure 4.21, we show the communication cost of intra-zone path key establishment for ω:τ equal to 6:4, 6:3 and 6:2. We analyze the communication cost of intra-zone path key establishment by computing the ratio of same-zone neighbors a node can reach after i hops, i=1, 2, 3. The first three bars show the communication cost when Z=1000 and the last three show the communication cost when Z=100. It can be observed that enlarging the sensor network by adding new zones does not significantly increase the communication cost of intra-zone path key establishment.

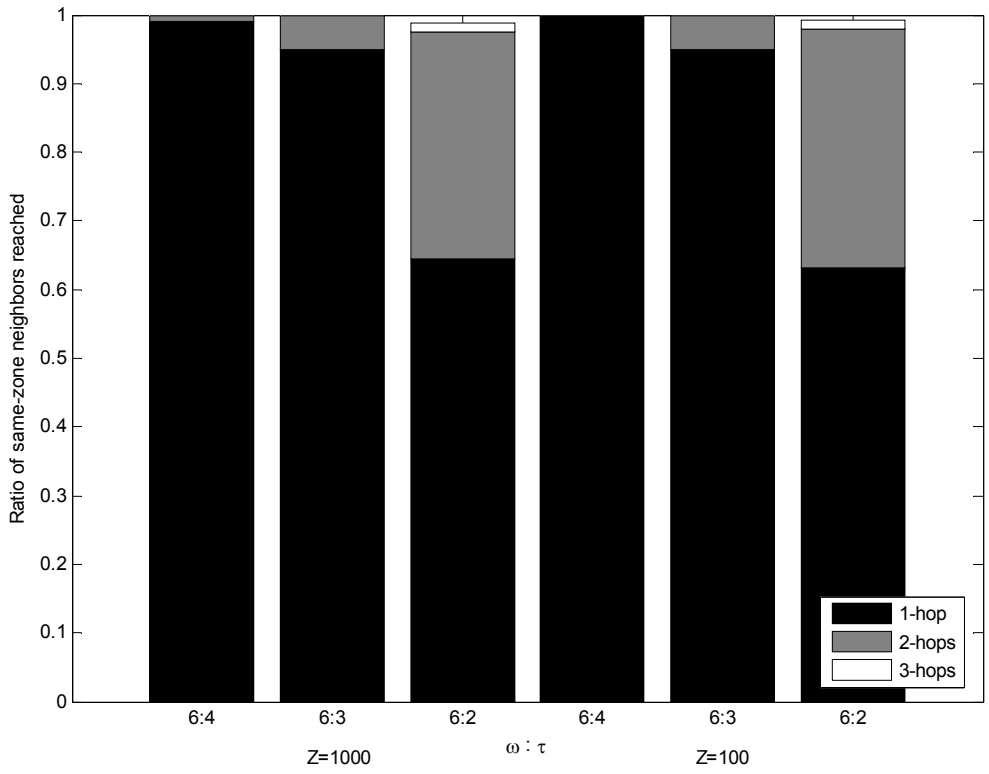

Figure 4-21 Ratio of same-zone neighbors reached when Z=1000 and Z=100

In Figure 4.22, we show the resiliency of our scheme against node capture for different sensor network sizes when P_{local} = 0.56. The ratio of compromised links are almost the same for Z=100 and Z=1000 when the ratio of captured nodes are the same. Thus, increasing the size of the sensor network does not degrade the node capture resiliency of our scheme.

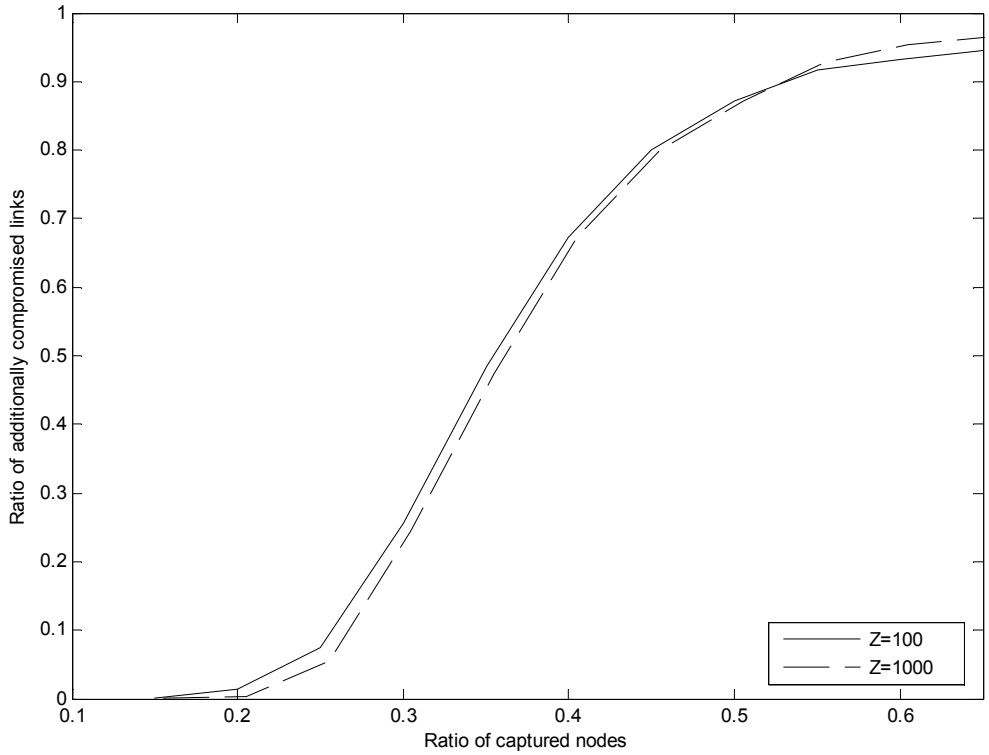

Figure 4-22 Ratio of additionally compromised links for our scheme when Z=100 and Z=1000

5 CONCLUSIONS

In this thesis, we presented a two-tier random key predistribution scheme for sensor networks. In our scheme, we used a zone-based approach, in which each zone has its own separate key spaces. Secure links between zones are established through agent nodes, which are higher capacity nodes. We utilized Blom's scheme [2] for key establishment between nodes from the same zones.

Our scheme achieves high local and global connectivity values while consuming minimal memory. When ω, number of key spaces in a zone, is 7 and τ, number of key spaces stored in a node, is 3, local connectivity for our scheme is 0.5605. Our scheme provides substantially better connectivity than key predistribution schemes that do not use deployment knowledge, such as Du et al's scheme [9] and Eschenauer and Gligor's scheme [1]. Under similar conditions, Du et al's scheme 2 [8], which utilizes deployment knowledge, performs P_{local} = 0.5335. In addition, our scheme has high global connectivity. For example, when ω is 6 and τ is 2, more than 99% of the sensor network is connected.

The communication cost of our scheme is within practical limits. At the intra-zone path key establishment phase, when ω is 6 and τ is 3, more than 90% of the nodes can establish secure links with their same-zone neighbors in only one hop and the rest can reach their same-zone neighbors in two hops. When ω is 4 and τ is 3, all the nodes can reach their same-zone neighbors in only one hop. Some communication cost is incurred while establishing hybrid keys with agent nodes. When number of agent nodes in a zone is 5, ω is 6 and τ is 4, 78 % of the nodes can reach their nearest agent node in one hop and 97 % of the nodes can reach their nearest agent node in at most two hops. While establishing inter-zone path keys, agent nodes and, occasionally, some intermediate nodes are required. When number of agent nodes in a zone is 10, 70 % of the inter-zone path key establishment processes causes only 7 messages and almost all inter-zone path keys can be established by exchanging at most 11 messages.

Our scheme achieves substantially strong node capture resiliency. In Du et al's scheme 2 [8], which is one of the major key predistribution schemes, an adversary can compromise 62 % of links by capturing 2000 nodes. Whereas, in our scheme, ratio of additionally compromised links when 2000 nodes are captured is almost zero. In addition, our scheme shows node-to-node authentication property. In this way, it becomes resilient against node fabrication attack. Besides, our key predistribution scheme is easily scalable due to the employed zone-based approach. Finally, since we create distinct key spaces for different zones, our scheme has partial resistance against wormhole attacks.

6 REFERENCES

[1] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks", in Proceedings of the 9th ACM conference on Computer and communications security, Washington, DC, USA, November 18-22 2002, 41–47.

[2] R. Blom, "An Optimal Class of Symmetric Key Generation System", in Advances in Cryptology - Eurocrypt'84, LNCS vol. 209, p. 335-338, 1985.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, August 2002.

[4] David Malan Crypto for Tiny Objects, Harvard University Technical Report TR-04-04.January, 2004.

[5] G. Gaubatz, J. Kaps, B. Sunar, Public Keys Cryptography in Sensor Networks - Revisited, Proceedings of the 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS), 2004.

[6] R. Watro, D. Kong, S. Cuti, C. Gardiner, C. Lynn, P. Kruus, Securing sensor networks with public key technology, Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks, October 2004

[7] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Research in Security and Privacy*, pages 197–213, 2003.

[8] W. Du, J. Deng, Y. S. Han, S. Chen, and P. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *Proceedings of IEEE INFOCOM'04*, March 2004.

[9] W. Du, J. Deng, Y. S. Han, and P. Varshney. A pairwise key predistribution scheme for wireless sensor networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*, pages 42–51, October 2003.

[10] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*, pages 52–61, October 2003.

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

[12] S. Zhu, S. Setia, and S. Jajodia. LEAP: Efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*, pages 62–72, October 2003.

[13] R. Anderson and M. Kuhn, "Tamper Resistance – a cautionary note," in *Proceedings of the Second Usenix Workshop on Electronic Commerce*, November 1996, pp. 1-11.

[14] J. Spencer, *The Strange Logic of Random Graphs*, Algorithms and Combinatorics 22, Springer-Verlag 2000, ISBN 3-540-41654-4.

[15] D. Liu, P. Ning, W. Du. Group-Based Key Pre-Distribution in Wireless Sensor Networks, Proceedings of 2005 ACM Workshop on Wireless Security

[16] Dijiang Huang, Manish Mehta, Deep Medhi, Lein Harn. *Location-Aware Key Management Scheme for Wireless Sensor Networks SASN'04,* October 25, 2004, Washington, DC, USA.

[17] Li Zhou, Jinfeng Ni, Chinya V. Ravishankar. *Efficient Key Establishment for Group-Based Wireless Sensor Deployments*. *WiSe'05,* September 2, 2005, Cologne, Germany.

[18] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. New York, NY: Elsevier Science Publishing Company, Inc., 1977.

[19] Erdös and Rényi, "On random graphs I," *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.

[20] Yang, H., Ye, F., Yuan, Y., Lu, S., and Arbaugh, W. 2005. Toward resilient s ecurity in wireless sensor networks. In *Proceedings of the 6th ACM international Symposium on Mobile Ad Hoc Networking and Computing* (Urbana-Champaign, IL, USA, May 25 - 27, 2005). MobiHoc '05.

[21] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, M. Yung, "Perfectly-Secure Key Distribution for Dynamic Conferences," Lecture Notes in Computer Science, vol 740, pp 471–486, 1993.

[22] Seyit A. Camtepe, Bulent Yener, "Key Distribution Mechanisms for Wireless Sensor Networks: a Survey" Technical Report, March 23, 2005.

[23] Lav Rai, Kameswari Kotapati. "A Survey on Sensor Network", http://www.cse.ohio-state.edu/□duttap/echelon/readings/rai03survey. pdf.

[24] Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. Wireless sensor networks: a survey. Computer Networks 38 (2002) 393-422

[25] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next century challenges: Mobile networking for smart dust," in *Proceedings of the 5th Annual ACM/IEEE Internation Conference on Mobile Computing and Networking (MobiCom)*, 1999, pp. 483–492.

[26] A. Chandrakasan, R. Amirtharajah, S. Cho, J. Goodman, G. Konduri, J. Kulik, W. Rabiner, A. Wang, Design considerations for distributed micro-sensor systems, Proceedings of the IEEE 1999 Custom Integrated Circuits Conference, San Diego, CA, May 1999, pp. 279–286.

[27] A. Cerpa, J. Elson, M. Hamilton, J. Zhao, Habitat monitoring: application driver for wireless communications technology, ACM SIGCOMM'2000, Costa Rica, April 2001.

[28] T.H. Keitt, D.L. Urban, B.T. Milne, Detecting critical scales in fragmented landscapes, Conservation Ecology 1 (1) (1997) 4. Available from <http://www.consecolo.org/vol1/iss1/art4>.

[29] P. Bonnet, J. Gehrke, P. Seshadri, Querying the physical world, IEEE Personal Communications (October 2000) 10–15.

[30] http://www.alertsystems.org.

[31] P. Johnson and D. C. Andrews, Remote continuous physiological monitoring in the home, Journal of Telemed Telecare 2 (2) (1996) 107–113.

[32] M. Ogawa, T. Tamura, T. Togawa, Fully automated biosignal acquisition in daily routine through 1 month, International Conference on IEEE-EMBS, Hong Kong, 1998, pp. 1947–1950.

[33] E.M. Petriu, N.D. Georganas, D.C. Petriu, D. Makrakis, V.Z. Groza, Sensor-based information appliances, IEEE Instrumentation and Measurement Magazine (December 2000) 31–35.

[34] N. Priyantha, A. Chakraborty, H. Balakrishnan, The cricket location-support system, Proceedings of ACM MobiCom'00, August 2000, pp. 32–43.

[35] G.J. Pottie, W.J. Kaiser, Wireless integrated network sensors, Communications of the ACM 43 (5) (2000) 551–558.

[36] J. Rabaey, J. Ammer, J.L. da Silva Jr., D. Patel, Pico-Radio: ad-hoc wireless networking of ubiquitous low energy sensor/monitor nodes, Proceedings of the IEEE Computer Society Annual Workshop on VLSI (WVLSI'00), Orlanda, Florida, April 2000, pp. 9–12.

[37] Wireless Integrated Network Sensors, University of California, Available: http://www.janet.ucla.edu/WINS.

[38] Crossbow. Wireless sensor networks. http://www.xbow.com/Products/Wireless Sensor Networks.htm.

[39] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, A. Chandrakasan, Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks, Proceedings of ACM MobiCom'01, Rome, Italy, July 2001, pp. 272–286.

[40] C. Intanagonwiwat, R. Govindan, D. Estrin, Directed diffusion: a scalable and robust communication paradigm for sensor networks, Proceedings of the ACM Mobi-Com'00, Boston, MA, 2000, pp. 56–67.

[41] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar, SPINS: security protocols for sensor networks, Proceedings of ACM MobiCom'01, Rome, Italy, 2001, pp. 189–199.

[42] A. Porret, T. Melly, C.C. Enz, E.A. Vittoz, A low-power low-voltage transceiver architecture suitable for wireless distributed sensors network, IEEE International Symposium on Circuits and Systems'00, Geneva, Vol. 1, 2000, pp. 56–59.

[43] B. Warneke, B. Liebowitz, K.S.J. Pister, Smart dust: communicating with a cubic-millimeter computer, IEEE Computer (January 2001) 2–9.

[44] Mani Srivastava, Sensor Node Platforms & Energy Issues, Mobicom2002, Tutorial2.

[45] M.A.M. Vieira, C.N. Jr. Coelho, D.C da Silva Jr., J.M da Mata, Survey on wireless sensor network devices. Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. 16-19 Sept. 2003 Volume: 1, On page(s): 537- 544 vol.1

[46] http://bwrc.eecs.berkeley.edu/Research/Pico_Radio/

[47] Zhou, L. and Haas, Z. 1999. Securing ad hoc networks. IEEE Network Magazine.

[48] Karlof, C. and Wagner, D, Secure routing in wireless sensor networks: attacks and countermeasures. Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on. 11 May 2003, pp 113-127

[49] Shi, E. Perrig, A. Designing Secure Sensor Networks. Wireless Communications, IEEE, Dec. 2004, vol: 11, Issue: 6, pp: 38- 43

[50] Newsome, J.; Shi, E.; Song, D.; Perrig, A., "The Sybil attack in sensor networks: analysis & defenses," *Information Processing in Sensor Networks*, 2004. IPSN 2004. Third International Symposium on , vol., no.pp. 259- 268, 26-27 April 2004

[51] J. R. Douceur. The Sybil attack. In First International Workshop on Peer-to-Peer Systems (IPTPS '02), Mar. 2002.

[52] A. Banerjea "A taxonomy of dispersity routing schemes for fault tolerant real-time channels" in *Proceedings of ECMAST:* vol. 26**.** May 1996, pp. 129-148.

[53] K. Ishida. Y. Kakuda and T. Kikuno. "A routing protocol for finding two node-disjoint paths in computer networks" in *International Conference on Network Protocols***,** November 1992, pp. 340-347.

[54] Cheekiralla, S. Engels, D.W. "A functional taxonomy of wireless sensor network devices" in Broadband Networks, 2005 2nd International Conference, 3-7 Oct. 2005, pp 26- 33