AN ENHANCED TREE BASED KEY MANAGEMENT SCHEME FOR SECURE COMMINCATION OF WIRELESS SENSORY NETWORK

Khadija Rasul
Student ID: 06110018

Nujhat Nuerie
Student ID: 06110030

Department of Electical and Electronic Engineering
April 2010

BRAC University, Dhaka, Bangladesh

DECLARATION

We hereby declare that this thesis is based on the results found by ourselves. Materials of work found by other researcher are mentioned by reference. This thesis, neither in whole nor in part, has been previously submitted for any degree.

_____	_____	_____
Signature of Supervisor	Signature of Author	Signature of Author

ACKNOWLEDGMENTS

First of all, we would like to thank all mighty Allah. We would like to give special thanks to Dr. Al-Sakib Khan Pathan who tirelessly helped us throughout our thesis, accepted all the difficult task of overseeing this work to completion and taught us how to think of independently. We are grateful to all the existing and previous faculty members of Department of Electrical & Electronic Engineering and Department of Computer Science & Engineering for helping all throughout last four years. Finally we would like to thank our parents for giving us all the opportunities and without whom we could not have come so far.

ABSTRACT

Wireless sensor networks (WSN) are mobile ad hoc networks in which sensors have limited resources and communication capabilities. Secure communications in some wireless sensor networks are critical. Key management is the fundamental security mechanism in wireless sensor network. To achieve security in WSN, it is important to be able to encrypt the messages sent between sensor nodes. In our thesis, we present an enhanced heterogeneous tree based key management scheme for security of wireless sensor networks. Our scheme combines efficiently different key management techniques in each architecture level and also it has it's our dynamic key renewal process. Here whenever a node is compromised key renewal is done by one way hash functions and simple XOR operations. This combination gives the scheme good performances in terms of key storage overhead as well as in terms of attack for node capture. We compared our scheme with most other heterogeneous schemes and overall our scheme gives better performances.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

CHAPTER I
OVERVIEW ON WIRELESS SENSOR NETWORK AND ITS APPLICATIONS

1.1 What Is Wireless sensor networks

Wireless sensor networks(WSNs) are the wireless networks that comprise a large number of spatially distributed small autonomous devices cooperatively monitoring environmental conditions and sending the collected data to a command center(called Base Station) using wireless channels. This small device, called sensor node, consists of sensor, wireless communication device, small micro- controller and energy source.

Wireless sensor network has some unique characteristics such as large scale of deployment, mobility of nodes, node failures, communication failures and dynamic network topology. In addition, each sensor node has constraints on resource such as energy, memory, computation speed and bandwidth because of the constraints on size and cost.

WSN have many applications in both military and civilian such as battlefield surveillance, habitat monitoring, healthcare, environmental monitoring, industrial monitoring, greenhouse monitoring, traffic control, etc. Many applications of the WSN require secure communications. However, Wireless sensor network are prone to different types of malicious attacks, such as impersonating, masquerading, interception for misleading because of the wireless connectivity, the absence of the physical protection and the unattended deployment, etc. Therefore, the security in sensor network is extremely important. However, the characteristics of the wireless sensor network make the incorporating security very challenge. The constraints on sensor make the design and operation exceedingly different from the contemporary wireless networks.

The existing security mechanisms for the wire-line and wireless networks cannot apply to the wireless sensor network because of the constrained energy, memory and computation capability. Thus, resource conscious security protocols and management techniques become necessity. Key management protocols are the core of the secure communications. The goal of the key management is to establish secure links between neighbor sensors at network formation phase. In our thesis we have tried to build a security scheme that seamlessly integrate WSN security in an energy efficient way.

1.2 Characteristics of WSN

Unique characteristics of a WSN include:

- Limited power they can harvest or store
- Ability to withstand harsh environmental conditions
- Ability to cope with node failures
- Mobility of nodes
- Dynamic network topology
- Communication failures
- Heterogeneity of nodes
- Large scale of deployment
- Unattended operation
- Node capacity is scalable only limited by bandwidth of gateway node.

Sensor nodes can be imagined as small computers, extremely basic in terms of their interfaces and their components. They usually consist of a processing unit with limited computational power and limited memory, sensors (including specific conditioning circuitry), a communication device (usually radio transceivers or alternatively optical), and a power source usually in the form of a battery. Other possible inclusions are energy harvesting modules, secondary ASICs, and possibly secondary communication devices (e.g. RS-232 or USB).

The base stations are one or more distinguished components of the WSN with much more computational, energy and communication resources. They act as a gateway between sensor nodes and the end user.

1.3 Applications Of WSN

The applications for WSNs are varied, typically involving some kind of monitoring, tracking, or controlling. Specific applications include habitat monitoring, object tracking, nuclear reactor control, fire detection, and traffic monitoring. In a typical application, a WSN is scattered in a region where it is meant to collect data through its sensor nodes.

1.3.1 Area Monitoring

Area monitoring is a common application of WSNs. In area monitoring, the WSN is deployed over a region where some phenomenon is to be monitored. For example, a large quantity of sensor nodes could be deployed over a battlefield to detect enemy intrusion instead of using landmines. When the sensors detect the event being monitored (heat, pressure, sound, light, electro-magnetic field, vibration, etc), the event needs to be reported to one of the base stations, which can take appropriate action (e.g., send a message on the internet or to a satellite). Depending on the exact application, different objective functions will require different data-propagation strategies, depending on things such as need for real-time response, redundancy of the data (which can be tackled via data aggregation and information fusion techniques), need for security, etc.

1.3.2 Environmental Monitoring

A number of WSNs have been deployed for environmental monitoring. Many of these have been short lived, often due to the prototype nature of the projects. Examples of longer-lived deployments are monitoring the state of permafrost in the Swiss Alps: The PermaSense Project, PermaSense Online Data Viewer and glacier monitoring.

1.3.3 Machine Health Monitoring Or Condition Based Maintenance

Wireless sensor networks have been developed for machinery condition-based maintenance (CBM) as they offer significant cost savings and enable new functionalities. In wired systems, the installation of enough sensors is often limited by the cost of wiring, which runs between $10–$1000 per foot. Previously inaccessible locations, rotating machinery, hazardous or restricted areas, and mobile assets can now be reached with wireless sensors. Often, companies use manual techniques to calibrate, measure, and maintain equipment. This labor-intensive method not only increases the cost of maintenance but also makes the system prone to human errors. Especially in US Navy shipboard systems, reduced manning levels make it imperative to install automated maintenance monitoring systems. Wireless sensor networks play an important role in providing this capability.

1.3.4 Industrial Monitoring

Water/Wastewater Monitoring:

There are many opportunities for using wireless sensor networks within the water/wastewater industries. Facilities not wired for power or data transmission can be monitored using industrial wireless I/O devices and sensors powered using solar panels or battery packs. As part of the American Recovery and Reinvestment Act (ARRA), funding is available for some water and wastewater projects in most states.

Landfill Ground Well Level Monitoring and Pump Counter:

Wireless sensor networks can be used to measure and monitor the water levels within all ground wells in the landfill site and monitor leachate accumulation and removal. A wireless device and submersible pressure transmitter monitors the leachate level. The sensor information is wirelessly transmitted to a central data logging system to store the level data, perform

calculations, or notify personnel when a service vehicle is needed at a specific well.

It is typical for leachate removal pumps to be installed with a totalizing counter mounted at the top of the well to monitor the pump cycles and to calculate the total volume of leachate removed from the well. For most current installations, this counter is read manually. Instead of manually collecting the pump count data, wireless devices can send data from the pumps back to a central control location to save time and eliminate errors. The control system uses this count information to determine when the pump is in operation, to calculate leachate extraction volume, and to schedule maintenance on the pump.

Flare Stack Monitoring:

Landfill managers need to accurately monitor methane gas production, removal, venting, and burning. Knowledge of both methane flow and temperature at the flare stack can define when methane is released into the environment instead of combusted. To accurately determine methane production levels and flow, a pressure transducer can detect both pressure and vacuum present within the methane production system.

Thermocouples connected to wireless I/O devices create the wireless sensor network that detects the heat of an active flame, verifying that methane is burning. Logically, if the meter is indicating a methane flow and the temperature at the flare stack is high, then the methane is burning correctly. If the meter indicates methane flow and the temperature is low, methane is releasing into the environment.

Water Tower Level Monitoring:

Water towers are used to add water and create water pressure to small communities or neighborhoods during peak use times to ensure water pressure is available to all users. Maintaining the water levels in these towers is important and requires constant monitoring and control. A wireless sensor network that includes submersible pressure sensors and float switches monitors the water

levels in the tower and wirelessly transmits this data back to a control location. When tower water levels fall, pumps to move more water from the reservoir to the tower are turned on.

Vehicle Detection:

Wireless sensor networks can use a range of sensors to detect the presence of vehicles ranging from motorcycles to train cars.

Agriculture:

Using wireless sensor networks within the agricultural industry is increasingly common. Gravity fed water systems can be monitored using pressure transmitters to monitor water tank levels, pumps can be controlled using wireless I/O devices, and water use can be measured and wirelessly transmitted back to a central control center for billing. Irrigation automation enables more efficient water use and reduces waste.

Windrow Composting:

Composting is the aerobic decomposition of biodegradable organic matter to produce compost, a nutrient-rich mulch of organic soil produced using food, wood, manure, and/or other organic material. One of the primary methods of composting involves using windrows.

To ensure efficient and effective composting, the temperatures of the windrows must be measured and logged constantly. With accurate temperature measurements, facility managers can determine the optimum time to turn the windrows for quicker compost production. Manually collecting data is time consuming, cannot be done continually, and may expose the person collecting the data to harmful pathogens. Automatically collecting the data and wirelessly transmitting the data back to a centralized location allows composting temperatures to be continually recorded and logged, improving efficiency, reducing the time needed to complete a composting cycle, and minimizing human exposure and potential risk.

An industrial wireless I/O device mounted on a stake with two thermocouples, each at different depths, can automatically monitor the temperature at two depths within a compost windrow or stack. Temperature sensor readings are wirelessly transmitted back to the gateway or host system for data collection, analysis, and logging. Because the temperatures are measured and recorded continuously, the composting rows can be turned as soon as the temperature reaches the ideal point. Continuously monitoring the temperature may also provide an early warning to potential fire hazards by notifying personnel when temperatures exceed recommended ranges.

Greenhouse Monitoring

Wireless sensor networks are also used to control the temperature and humidity levels inside commercial greenhouses. When the temperature and humidity drops below specific levels, the greenhouse manager must be notified via e-mail or cell phone text message, or host systems can trigger misting systems, open vents, turn on fans, or control a wide variety of system responses. Because some wireless sensor networks are easy to install, they are also easy to move as the needs of the application change.

1.4 Challenges In WSN

There are various types of challenges wireless sensor network has to face also due to the nature of their work. Along with different type of attack by adversary, developing sensor networks that ensures high-security features with limited resources is a challenge too. Wireless Sensor networks cannot be costly made as there is always a great chance that they will be deployed in hostile environments and captured for key information or simply destroyed by an adversary, which, in turn, can cause huge losses. Part of these cost limitation constraints includes an inability to make sensor networks totally tamper-proof. Other sensor node constraints that must be kept in mind while developing a key establishment technique include battery life, transmission range, bandwidth, memory, and prior deployment knowledge. The challenges are discussed details in the following sections.

1.4.1 Attacks:

Attacks against wireless sensor networks could be broadly considered from two different levels of views. One is the attack against the security mechanisms and another is against the basic mechanisms (like routing mechanisms). Here we point out the major attacks in wireless sensor networks.

DoS attack:

DoS attack tries to exhaust the resources available to sensor node by sending extra unnecessary packets and thus prevent network users from accessing services. In wireless sensor networks, several types of DoS attacks in different layers might be performed.

Attacks on Information in transit:

Attacks on Information in transit means attackers with high processing power and large communication range can monitor the traffic flow and fabricate them to provide wrong information to the base.

Sybil attack:

In case of Sybil attack a node tries to forge the identities of more than one node to degrade the integrity and security of data.

Blackhole attack:

In Black hole attack a malicious node tries to attract all the traffics in sensor network the attacker listens to requests for routes then replies to the target nodes that it contains the high quality or shortest path to the base station. Once the malicious device has been able to insert itself between the communicating nodes (for example, sink and sensor node), it is able to do anything with the packets passing between them.

Wormhole attack:

In case of Wormhole attack attacker tries to record and tunnel packets from one location to another in a network. Wormhole attack is a significant threat to wireless sensor networks, because; this sort of attack does not require compromising a sensor in the network rather, it could be performed even at the initial phase when the sensors start to discover the neighboring information.

Hello Flood Attack:

In case of Hello Flood Attack an attacker with a high radio transmission range sends HELLO packets to the sensor nodes. The sensors are thus convinced that the attacker is their neighbor. As a result, the victim nodes go through the attacker while sending information to the base and are ultimately spoofed by the attacker.

Clone attack:

Once a sensor is compromised, the adversary can easily launch clone attacks by replicating the compromised node, distributing the clones throughout the network, and starting a variety of other attacks such as insider attacks.

1.4.2 Battery life:

Sensor nodes have a limited battery life, which can make using asymmetric key techniques, like public key cryptography, impractical as they use much more energy for their integral complex mathematical calculations. This constraint is mitigated by making use of more efficient symmetric techniques that involve fewer computational procedures and require less energy to function.

1.4.3 Transmission range:

Limited energy supply also restricts transmission range. Sensor nodes can only transmit messages up to specified short distances since increasing the range may lead to power drain. Techniques like in-network processing can help

to achieve better performance by aggregating and transmitting only processed information by only a few nodes. This way it can save the dissipated energy.

1.4.4 Bandwidth:

It is not efficient to transfer large blocks of data with the limited bandwidth capacity of typical sensor nodes, such as the transmitter of the UC Berkeley Mica platform that only has a bandwidth of 10Kbps. To compensate, key establishment techniques should only allow small chunks of data to be transferred at a time.

1.4.5 Memory:

Memory availability of sensor nodes is usually 6–8 Kbps, half of which is occupied by a typical sensor network operating system, like TinyOS. Key establishment techniques must use the remaining limited storage space efficiently by storing keys in memory, buffering stored messages, etc.

1.4.6 Prior deployment knowledge:

As the nodes in sensor networks are deployed randomly and dynamically, it is not possible to maintain knowledge of every placement. A key establishment technique should not, therefore, be aware of where nodes are deployed when initializing keys in the network.

1.5 Requirements, Constraints And Evaluation Metrics Of WSN

Before going into detailed discussion about individual scheme we would like to discuss about the characteristics of secure communication in wireless sensor network.

The achievement of communication security is a challenging task because of the “fragile nature” of WSN. WSN have a set of characteristics which complicates the implementation of traditional security and key management solutions.

First of all, the wireless nature of communications in WSN makes it easier for attackers to intercept all transmitted packets. Second, WSN are constrained by the limited resources. Due to the following limitations, it is difficult to implement complicated security solutions in WSN. Third, in many cases, a large number of sensors are needed to be deployed in a hostile environment, which makes it very hard to have a continuous control on sensors. Finally, WSN are vulnerable to physical attackers. An attacker can capture one or more sensors and reveal all stored security information (particularly stored keys) which enables him to compromise a part of the WSN communications.

For all these reasons, an efficient key management scheme should be implemented in the sensor before its deployment. The key establishment technique employed in a given sensor network should meet several requirements to be efficient. These requirements may include supporting in-network processing and facilitating self-organization of data, among others. However, the key establishment technique for a secure application must minimally incorporate authenticity, confidentiality, integrity, scalability, and flexibility.

Authenticity:

The key establishment technique should guarantee that the communication nodes in the network can verify the authenticity of the other nodes involved in a communication. The receiver node should recognize the assigned ID of the sender node.

Confidentiality:

The key establishment technique should defend the expose of data from illegal parties. An opposition may attack a sensor network by obtaining secret keys. A better key Management scheme controls the compromised nodes so that data is not farther revealed.

Integrity:

Here integrity means only the nodes in the network should have access to the key sand only an assigned base station should have the privilege to change the keys. Unauthorized nodes should not be able to establish communications with network nodes and thus gain entry into the network.

Scalability:

Sensor networks should employ a scalable key establishment technique. Key establishment techniques employed should provide high-security features for small networks, but also maintain these characteristics when applied to larger ones.

Flexibility:

Key establishment techniques should be able to function well in any kind of environments and support dynamic deployment of nodes, i.e., a key establishment technique should be useful in multiple applications and allow for adding nodes at any time.

A key establishment technique is not judged solely based upon its ability to provide secrecy of transferred messages, but must also meet certain other criteria for efficiency in light of vulnerability to adversaries, including resistance, revocation, and resilience.

Resistance against node capture:

An adversary might attack the network by compromising a few nodes in the network and then replicate those nodes back into the network. Using this attack the adversary can populate the whole network with his replicated nodes and thereby gain control of the entire network. A good key establishment technique must resist node replication to guard against such attacks.

Revocation:

If a sensor network becomes invaded by an adversary, the key establishment technique should provide an efficient way to dynamically remove the compromised nodes.

Resilience:

If a node within a sensor network is captured by adversary, the key establishment technique should ensure that secret information about other uncompromised nodes is not revealed. A scheme's resilience is calculated using the total number of nodes compromised and the total fraction of communications compromised in the network. Resilience also means conveniently making new inserted sensors to join secure communications.

Key management protocols are the core of the secure communications. The goal of the key management is to establish secure links between neighbor sensors at network formation phase. Several key management schemes have been proposed in recent years. Recently, many key management schemes for the wireless sensor network have been proposed. Some researchers have investigated the wireless sensor networks key management schemes and divided them into different categories. The major categories will be discussed in the following chapter.

CHAPTER II
DIFFERENT SYMMETRIC KEY MANAGEMENT SCHEMES TO OVERCOME ATTACKS AND TO ENSURE SECURITY IN WSN

Key management schemes can be classified into three categories based on the encryption techniques: symmetric, asymmetric and hybrid. In our pre-the thesis we have mainly studied the symmetric schemes as Symmetric-key based schemes are widely used because of their relatively less computation complexity, which are suitable for the limited resource characteristics of the wireless sensor network. Most of the wireless sensor network uses the symmetric key schemes because these schemes consume less computation time than other schemes. Symmetric schemes can be broken down into homogeneous and heterogeneous schemes. There are two desirable characteristics of a sensor network lower hardware cost, and uniform energy drainage. While heterogeneous networks achieve the former, the homogeneous networks achieve the latter. However both features cannot be incorporated in the same network.

2.1 Homogeneous Key Management Schemes:

In homogeneous networks all the sensor nodes are identical in terms of battery energy and hardware complexity. With purely static clustering (cluster heads once elected, serve for the entire lifetime of the network) in a homogeneous network, it is evident that the nodes communicating with base station will be over-loaded with the long range transmissions to the remote base station, and the extra processing necessary for data aggregation and protocol coordination. As a result such nodes expire before other nodes. However it is desirable to ensure that all the nodes run out of their battery at about the same time, so that very little residual energy is wasted when the system expires. Another characteristic of homogeneous network is role rotation that is all the nodes should be capable of acting as the node communicating with base station and therefore should possess the necessary hardware capabilities. Few major

schemes of homogeneous key management technique are discussed in the following sections.

2.1.1 Entity based scheme

Entity based schemes or arbitrated scheme share those schemes in which key distributions and key establishment are based on trusted entity. There are different types of Entity based scheme. Master Key Technique Trusted third node based scheme and Base station participation scheme are examples of Entity based scheme and they are discussed below:

Master Key Technique:

In this scheme as proposed by Lai et al. (2002) [1] a single key, the master key, is preloaded into all the nodes of the network. After deployment, every node in the network can use this key to encrypt and decrypt messages. Scheme in [1] counters several constraints with less computation and reduced memory use, but it fails in providing the basic requirements of a sensor network by making it easy for an adversary trying to attack.

Benefits:

- This technique has minimal storage requirements and avoids complex protocols.
- Only a single key is to be stored in the nodes memory and once deployed in the network, there is no need for a node to perform key discovery or key exchange.
- Offers Infinite scalability.

Flaws:

- The main drawback is that compromise of a single node causes the compromise of the entire network through the shared key.

Improvement:

We can improve the scheme by erasing the master key after the pairwise keys are established. As a result the resilience is improved but when new nodes are added later they still have master key which makes the network not completely secure.

Base station participation scheme:

In this scheme as presented by Perrig et al. (2001) presented [2], the nodes are pre-initialized with a single key shared with an online server known as the key distribution center (KDC). When the users want to establish secure communication among them, each one of them has to obtain a new session key, encrypted with this pre-initialized key, from the KDC.

Benefit:

- The scheme in [2] has small memory requirement and perfectly controlled node replication.
- It is resilient to node capture and possible to revoke key pairs.

Flaws:

- The scheme in [2] is not scalable and the base station becomes the target of attacks.

Trusted third node based scheme:

Chan and Perrig (2005) [3] peer intermediaries for key establishment in sensor network called ‘‘PIKE’’. In this scheme, the key establishment between two sensor nodes is based on the common trust of a third node. For any two nodes of A and B, there is a node C that shares a key with nodes A and B.

2.1.2 Random key pre-distribution scheme

In the Basic Scheme proposed by Eschenauer and Gligor[4], key distribution is divided into three stages: key pre-distribution, shared-key discovery, and path-key establishment.

Stage 1: Key pre-distribution stage

In the key pre-distribution stage, a large key pool of |s| keys and their identifiers are generated. From this key pool, K keys are randomly drawn and pre-distributed into each node's key ring, including the identifiers of all those keys. This key pre-distribution process ensures that, though the size of the network is large, only a few keys need to be stored in each node's memory, thereby saving storage space. These few keys are enough to ensure that two nodes share a common key, based on a selected probability.

Stage 2: Shared-key discovery stage:

Once the nodes are initialized with keys, they are deployed in the respective places where they are needed, such as hospitals, war fields, etc. After deployment, each node tries to discover its neighbors with which it shares common keys. There are many ways for finding out whether two nodes share common keys or not. The simplest way is to make the nodes broadcast their identifier lists to other nodes. If a node finds out that it shares a common key with a particular node, it can use this key for secure communication. This approach does not give the adversary any new attack opportunities and only leaves room for launching a traffic analysis attack in the absence of key identifiers.

Stage 3: Path key establishment stage:

The path key establishment stage makes provision for link between two nodes even when they do not share a common key. Let us suppose that node u wants to communicate with node v, but they do not share a common key between them. Node u can send a message to node y saying that it wants to communicate with node v; this message is then encrypted using the common key shared between node u and node y and, if node y has a key in common with node v, it can generate a pair wise key K_{uv} for nodes u and v, thereby acting like

a key distribution center or a mediator between the communication of nodes u and v. As all the communications are encrypted using their respective shared keys, there will not be a security break in this process. After the shared key discovery stage is finished there will be a number of Keys left in each sensor's key ring that are unused and can be put to work by each sensor node for path key establishment.

In the Basic Scheme, node revocation is conducted by the controller node. When a node is revoked, all the keys in that particular node key ring have to be deleted from the network. After the matching keys are completely deleted from all the nodes, there may be links missing between different ones and they then have to reconfigure themselves starting from the hared key discovery stage so that new links can be formed between them. As only few keys are removed from the network, the revocation process only affects a part of it and does not incur much communication overhead.

Analysis of the basic pre-distribution Scheme:

If the probability that a common key exists between two nodes in the network is p, and the size of the network is n. The degree of a node d is derivable using both p and n. since the degree of any node is simply the average number of edges connecting that node with other nodes in its neighborhood, therefore,

$$d = p \cdot (n-1) \tag{2.1}$$

The value of d is such that a network of n nodes is connected with a given probability P. We then must calculate the key ring size k and the size of the key pool |S|.

Connectivity of the wireless sensor network can be analyzed by the random-graph theory .A random graph G (n, p) is a graph of n nodes, in which the probability that a link exists between two nodes is p. Given a desired probability P_c for graph connectivity, the function p is defined as follows:

$$P_{\lim \to \infty} = \Pr[G(n,p) \text{ is connected}] = e^{e^{-c}} \tag{2.2}$$

$$p = \frac{\ln(n)}{n} + \frac{c}{n} \quad , \tag{2.3}$$

here c is a real constant.

Eschenauer and Gligor have shown that for a pool size S = 10,000 keys, only 75 keys need to be stored in a node's memory to have the probability that they share a key in their key rings to be p = 0.5. If the pool size is ten times larger, i.e. S= 100,000, then the number of keys required is still only 250.

Benefits:

- Advantages of [4] include flexible, efficient, and fairly simple to employ,
- Also offering good scalability.

Flaws:

- It cannot be used in circumstances demanding heightened security and node to node authentication.
- It does not provide the node-to-node authentication property that ascertains the identity of a node with which another node is communicating.

Improvement on the Basic scheme:

Many key management schemes are proposed as extensions of the Basic Scheme to make it even more secure and reliable. Improvement can be brought In Basic scheme version to enhance the security. Three improvements have been proposed by Chan and Perrig .The first one is called q-composite keys scheme introduced by Chan, Perrig, and Song [5]. This scheme employs q common keys to set up the common key with a hash function rather than only one. They showed that this scheme strengthens the network's resilience against node capture attack when the number of node capture is small .But it may make the network more vulnerable once a large number of nodes have been breached. The second scheme is called multi-path key reinforcement presented also in [5] .

This scheme establishes the link key through multiple paths to strengthen the security. The tradeoff is that it increases the communication overhead in wireless sensor network. There is a third variation which is an enhancement of the commonly known Pair wise Scheme, called Random Pair wise Scheme was also proposed by them in [5] .

Q-Composite Random Key Pre-distribution Scheme:

The Q-Composite Random Key Pre-distribution introduced in [5] requires that two nodes have at least q common keys to set up a link. In the process, nodes will fail to establish a link if the number of keys shared is less than q; otherwise, they will form a new communication link using the hash of all the q keys

$$K = \text{hash}(k_1 i\ k2i\ .\ .\ .\ .\ .\ .\ k_q i) \tag{2.4}$$

As the amount of key overlap between two nodes is increased, it becomes harder for an adversary to break their communication link. But S, the size of the key pool, is the critical parameter that must be calculated for the Q-Composite Scheme to be efficient. If S is large, then the probability that two nodes share a common key and therefore can communicate is decreased. However, if S is decreased, an adversary's job may be easier as he can now gather most of the keys in the key pool by capturing only a few nodes. Thus, S must be chosen such that the probability of any two nodes sharing at least q keys is larger than or equal to p.

Scheme in [5] offers greater resilience compared to the Basic Scheme when a small number of nodes have been captured in the network. The amount of communications that are compromised in a given network with the Q-Composite Scheme applied is 4.74 percent when there are 50 compromised nodes, while the same network with the Basic Scheme applied will have 9.52 percent of communications compromised. Though [5] performs badly when more nodes are captured in a network, this may prove a reasonable concession as adversaries are more likely to commit a small-scale attack and preventing smaller attacks can push an adversary to launch a large-scale attack, which is far easier to detect.

Benefits:

- It provides better security than the Basic Scheme by requiring more keys for two nodes to share one for communication, which makes it difficult for an adversary to compromise a node.
- Offers greater resilience compared to the Basic Scheme.

Flaws:

- Disadvantages of this scheme include that it is vulnerable to breakdown under large-scale attacks
- Does not satisfy scalability requirements.

Multipath key reinforcement scheme:

The Multipath Reinforcement Scheme presented in [5] offers good security with additional communication overhead for use where security is more of a concern than bandwidth or power drain.

The idea of using a multipath to reinforce links in a random key establishment scheme was first explored by Anderson and Perrig. Chan, Perrig, and Song further developed the Multipath Key Reinforcement Scheme for establishing a link between two nodes of a given network that is stronger than that in the Basic Scheme.

The links formed between nodes after the key discovery phase in the Basic Scheme are not totally secure due to the random selection of keys from the key pool allowing nodes in a network to share some of the same keys and, thereby, possibly threaten multiple nodes when only one is compromised. To solve this problem, the communication key between nodes must be updated when one is compromised once a secure link is formed. This should not be done via the already established link, as an adversary might decrypt the communication to obtain the new key, but should be coordinated using multiple independent paths for greater security.

$$k' = k \oplus g_1 \oplus g_2 \oplus \ldots \oplus g_h, \tag{2.5}$$

If node A needs an updated communication key with node B, all possible disjointed paths to node B must be used. Assume that there are h such disjointed paths from node A to node B. Then node A generates h random values (g1,g2, . . . ,gh) each equal to the size of an encryption key, and sends one down each available disjointed path to node B. When node B has received all h random values, it computes the new encryption key at the same time as node A does form a new and secure communication link with using equation (2.5) .Here k is the original key.

With the new link in place, the only way an adversary can decrypt the communications is to compromise all the nodes involved in the formation of the key. The larger h is, the more paths and nodes involved and the greater the security of the new link. This increase in network communications causes excessive overhead in finding multiple disjointed paths between two nodes. Also, as the size of a path increases, it may grow so long as it leaves a chance for an adversary to eavesdrop, which makes the whole path insecure.

Benefits:

- It offers better security than in [2] or the Q-Composite.
- The only way an adversary can decrypt the communications is to compromise all the nodes involved in the formation of the key.

Flaws:

- It creates communication overhead that can lead to depleted node battery life.
- Enhance the chance for an adversary to launch DOS attacks.

Improvement:

A 2-hop approach to the Multipath Key Reinforcement Scheme considers only 2-link paths to minimize the overhead of path length by using disjointed paths that are only one intermediate node away from the two original nodes (A and B).

Random pair wise key scheme:

Chan, Perrig, and Song developed the Random Pair wise scheme in [5] as an extension of the Pair wise Scheme to help overcome this drawback.
In the basic version of this technique, called the *trivial solution*, each node should store exactly ($N - 1$) pair-wise keys; one key with each other sensor in the network. This basic version offers a high level of resistance against node capture attack. However it suffers from important memory consumption for key storage. It is a non scalable solution, and may only be used in small or medium WSN.

In Random pair-wise key scheme of [5], authors proposed to use only $N_p = Nxp$ keys instead of $(N - 1)$ keys by each node, where p is the probability that two nodes in the network are connected. They stated that not all n -1 keys are required to be stored in a node's key ring. As we have already seen with the Basic Scheme, not all nodes must be connected as long as node connections meet some desired probability P, which dictates that only (nxp) keys are needed to be stored in a given node's key ring, where n being the number of nodes in the network and p being the probability that two nodes can communicate securely. Given this, if k is the number of keys in a node's key ring, the maximum allowable network size can be determined with $n = k/p$ for the Random Pair wise Scheme.

By adapting the probability *p* to the WSN characteristics, we can reduce the number of stored keys, and hence the scalability, while keeping a good level of connectivity between nodes.

Advantages:

- Is offers the best security of all the above schemes with perfect resilience to node capture as the keys used by each node are unique.
- Also provides resistance against node replication.

Disadvantages:

- Disadvantages of this scheme include that it does not support networks of large size
- Does not satisfy scalability requirements.

2.1.3 Polynomial-based key pre-distribution scheme

Polynomial key pre-distribution scheme is the basis of pair-wise keys pre-distribution schemes developed by Liu and Ning (2003) [6] .There is two form of this scheme. One is basic Polynomial pool-based key pre-distribution scheme and the other is Grid- based key pre-distribution scheme**.**

In general in this scheme, in order to pre-distribute pairwise keys, one keyset-up sever randomly generates a t-degree polynomial $f(x , y)=\sum^{t}_{i,j} a_{i,j} x^{i} y^{j}$ over a finite field F_q , where q is a prime number that is large enough to accommodate a cryptographic key, and has the property of $f(x , y)= f(y , x)$.

In sensor network, each sensor is assumed to have a unique ID. For each sensor i, the set-up server computes a polynomial share of f(x, y) that is, f(i , y). For any two sensor nodes i and j, node i can compute the common key f(i, j) by evaluating f(i, y) at point j, and node j can compute the common key with i by evaluating f(j , y) at i.

To enhance the security of the above scheme, Liu and Nin developed a polynomial pool-based pair wise key pre-distribution based on above scheme. The basic idea is the combination of polynomial- based key pre-distribution and the key pool idea.

Polynomial pool-based key pre-distribution:

Polynomial pool-based pair wise key pre-distribution uses a pool of multiple random bivariate polynomials and can be considered as extension of the polynomial-based scheme. When the polynomial pool has only one polynomial the general framework degenerates into the polynomial-based key pre-distribution. When all the polynomials are 0-degree ones, the polynomial pool degenerates into a key pool. Pair wise key establishment needs three phases: setup, direct key establishment, and path key establishment.

In the setup phase, setup server randomly generates a set F of bivariate t-degree polynomials over the finite field F_q. In the direct key establishment, if both

sensors have polynomial shares on the same bivariate polynomial, they can establish the pair wise key directly using the polynomial-based key pre-distribution. The third phase is needed if direct key establishment fails. Two sensor nodes will establish a pair wise key with the help of other sensors. Compared with previous schemes, this scheme improved the security and the scalability.

One instantiation of the general framework is called random subset assignment scheme. Subset assignment scheme uses a random strategy for subset assignment during the setup phase. The setup server selects a random subset of polynomials in F and assigns their polynomial shares to the each sensor. The main difference with the basic probabilistic scheme is that this scheme randomly chooses polynomials from a polynomial pool and assigns their polynomial shares to each sensor instead of randomly selecting keys from a large key pool and assigning them to sensors. Therefore, this scheme can be considered as an extension to the basic probabilistic scheme. The probability of two sensors sharing the same bivariate polynomial is the same as the probability of the two sharing a common key as described in the Basic Scheme.

Grid- based key pre-distribution scheme:

Another instantiation of the general framework is called grid- based key pre-distribution scheme presented in [6]. The setup server assigns each sensor in the network to a unique intersection in this grid. This scheme has some better properties over previous schemes. It is resilience to node compromise and there is no communication overhead during polynomial share discovery.

If a network consists of N sensor nodes, an (m x m) grid with a set of 2m polynomials is constructed, calculated as $\{ f_i^{c}(x, y), f_i^{r}(x, y)\}$ i=0 to m- 1, Where the value of m is the square root of N. each row i in the grid is associated with a polynomial $f_i^{r}(x, y)$ and each column of the grid is associated with a polynomial share $f_i\, c(x, y)$.

In the first stage, the setup server distributes an intersection in the grid to each node, and then distributes the polynomial shares of that particular column and row to the node to provide each node with the information required for key discovery and path key establishment. In the second stage, if a node i want to establish a pair wise key with node j, it checks for common rows or columns with j i.e., $c_i = c_j$ or $r_i = r_j$. The pair wise key can be established using the polynomial shares of a row or column that matches. If there is no match, then nodes i and j must find an alternate path. To do so, node i find an intermediate node through which it can establish a pair wise key with node j. Even if some intermediate nodes are compromised there exist many connecting paths in the grid between the two nodes.

Advantages:

- There will be a greater chance for nodes to establish a pairwise key with others without communication overhead as the sensors are deployed in a grid-like structure.
- Nice resilience to node capture until a certain percentage of nodes are compromised (60 percent).

Disadvantages:

- This grid-based approach to the Polynomial Pool-Based Scheme has reasonable overhead when compared to other schemes. Each node must store 2 bivariate t-degree polynomials and IDs of the compromised nodes with which it can establish a pairwise key.

2.1.4 Matrix-based key pre-distribution scheme

Matrix based key pre-distribution scheme is built on Blom's key pre-distribution scheme and combines the random key pre-distribution method with it. This scheme is thoroughly discussed in [7]. It has the following λ-secure property: 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. The

threshold λ can be treated as a security parameter in that selection of a larger λ leads to a more secure network. However, λ also determines the amount of memory to store key information, as increasing λ leads to better security with higher memory usage. According to [7] , if two nodes carry key information from a common space, they can compute their pairwise key from the information; when two nodes do not carry key information from a common space, they can conduct key agreement via other nodes which share pairwise keys with them.

Basic ideas:

Basic idea of this scheme is introduced in [8].In [7], during the pre-deployment phase, the base station first constructs a (λ + 1) × N matrix G over a finite field where N is the size of the network. G is considered as public information; any sensor can know the contents of G.

$$G = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ s & s^2 & s^3 & \cdots & s^N \\ s^2 & (s^2)^2 & (s^3)^2 & \cdots & (s^N)^2 \\ & & & \vdots & \\ s^\lambda & (s^2)^\lambda & (s^3)^\lambda & \cdots & (s^N)^\lambda \end{bmatrix} \tag{2.6}$$

Then the base station creates a random (λ+1)×(λ+1)symmetric matrix D over GF(q), and computes an N × (λ + 1) matrix $A = (D \cdot G)^{T,}$ where $(D \cdot G)^T$ is the transpose of D · G. Matrix D needs to be kept secret, and should not be disclosed to adversaries or any sensor node Because D is symmetric, it is easy to see:

$$A \cdot G = (D \cdot G)^T \cdot G = G^T \cdot D^T \cdot G = G^T \cdot D \cdot G = (A \cdot G)^T.$$

This means that A·G is a symmetric matrix. If we let K = A·G, we know that $K_{ij} = K_{ji}$, where K_{ij} is the element in K located in the i^{th} row and j th column. We use K_{ij} (or K_{ji}) as the pairwise key between node i and node j. Fig. 1 illustrates how the pairwise key $K_{ij} = K_{ji}$ is generated.

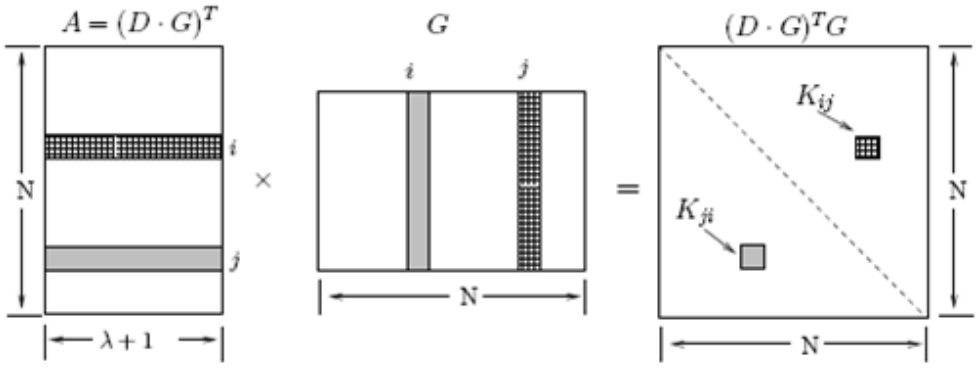

Figure 2.1: Generating keys in Blom's Scheme

To carry out the above computation, nodes i and j should be able to compute K_{ij} and K_{ji}, respectively. This can be easily achieved using the following key pre-distribution scheme, for K = 1. . .N

1. store the kth row of matrix A at node K
2. store the kth column of matrix G at node K

Therefore, when nodes i and j need to find the pairwise key between them, they first exchange their columns of G, and then they can compute K_{ij} and K_{ji}, respectively, using their private rows of A. The scheme is λ-secure if any λ + 1 columns of G are linearly independent. This λ-secure property guarantees that no nodes other than i and j can compute K_{ij} or K_{ji} if no more than λ nodes are compromised.

Here G is a Vander monde Matrix hence when we store the kth column of G at node K, we only need to store the seed s_k at this node, and it can regenerate the column given the seed.

In the improved version of [8], we change the complete graph into connected graph so that each node needs less memory to store key information. It assigns keys only to connected graph. Hence this scheme is scalable and more resilient to node capture.

How the scheme works:

Step 1: Key Pre-distribution Phase

During the key pre-distribution phase, we need to assign key information to each node, such that after deployment, neighboring sensor nodes can find a secret key between them. Assume that each sensor node has a unique identification, whose range is from 1 to N. the base station create a generator matrix G of size (λ+1)×N. Let G(j) represent the jth column of G. We provide G(j) to node j. As mentioned above, although G(j) consists of (λ+1) elements, each sensor only needs to remember one seed. Then Base station generate ω symmetric matrices D1,. . ., Dω of size (λ + 1) × (λ + 1). We call each tuple Si = (Di,G), i = 1, . . . , ω, a key space. We then compute the matrix Ai = $(Di \cdot G)^T$. Let $A_i(j)$ represents the jth row of A_i. We randomly select τ distinct key spaces from the ω key spaces for each node. For each space S_i selected by node j, we store the jth row of A_i (i.e. $A_i(j)$) at this node. This information is secret and should stay within the node; According to [8], two nodes can find a common secret key if they have both picked a common key space. Since A_i is an N × (λ + 1) matrix, $A_i(j)$ consists of (λ + 1) elements. Therefore, each node needs to store (λ+1)τ elements in its memory.

Step 2: Key Agreement Phase

Assume that nodes i and j are neighbors, and they have received the above broadcast messages. If they find out that they have a common space, e.g. S_c, they can compute their pairwise secret key using [8]: Initially node i has $A_c(i)$ and seed for G(i), and node j has $A_c(j)$ and seed for G(j). After exchanging the seeds, node i can regenerate G(j) and node j can regenerate G(i); then the pairwise secret key between nodes i and j, $K_{ij} = K_{ji}$, can be computed in the following manner by these two nodes independently:

$$K_{ij} = K_{ji} = A_c(i) \cdot G(j) = A_c(j) \cdot G(i). \quad (2.7)$$

After secret keys with neighbors are set up, the entire sensor network forms a connected graph as mentioned above. We now show how two neighboring nodes, i and j, who do not share a common key space could still come up with a pairwise secret key between them. Assume that the path is v_i, v_1,. . ., v_t,v_J. To find a common secret key between i and j, i first generates a random key K. Then i sends the key to v_1 using the secure link between i and v_1; v_1 sends the key to v_2 using the secure link between v_1 and v_2, and so on until j receives the key from v_t. Nodes i and j use this secret key K as their pairwise key. Because the key is always forwarded over a secure link, no nodes beyond this path can find out the key.

Calculation of parameters:

Here the probability that a connected graph,p_c, is formed is when local connectivity p_{actual} must be greater than a minimum value known as $p_{required}$.

Let d be expected degree of a node,N size of network and n be expected number of neighbor

$$P_{required}=d/N=\frac{(N-1)[\ln(N)-\ln(-\ln(Pc))]}{N} \quad (2.8)$$

For a given ω and τ,

P_{actual}=1-(2 nodes not sharing a any space)

$$P_{actual}=1-\frac{((\omega-\tau)!)^2}{2!(\omega-2\tau)!\omega!} \quad (2.9)$$

For a given $P_{required}$ and P_{actual} we can find ω and τby solving below equation

$$P_{actual} >= P_{required}$$

$$1 - \frac{((\omega-\tau)!)^2}{2!(\omega-2\tau)!\omega!} >= \frac{(N-1)[\ln(N)-\ln(-\ln(Pc))]}{nN} \quad (2.10)$$

Costs incurred in the scheme are as follows:

Storage: m = (λ + 1) τ.

Communication cost: When τ is large, communication cost is 2 hops(or number of hops neighbor is away)

Computation cost: Regeneration of corresponding column of G from a seed + doing inner product of corresponding row of $(D \cdot G)^T$ with column of G is equal to (λ - 1)+ (λ + 1)=2(λ)

Benefit:

- Here is x nodes are compromised and x<λ, then no additional uncompromised nodes communication can be affected so the fraction of communication links compromised should be the same as the fraction of the spaces compromised .
- Less memory is needed as this scheme assigns keys only to nodes in connected graph instead I of entire graph .
- Less energy consumed then any asymmetric scheme
- this scheme is scalable

Flaws:

- Limitation of λ remains. If more than λ rows are compromised ,the entire secret matrix can be derived or broken by adversaries.

Improvement:

In this scheme resilience is greatly improved by using 2 hop neighbor. When we treat a two-hop neighbor as a neighbor, the radius of the range covered by a node doubles, so the area that a node can cover is increased by

four times. Therefore, the expected number of neighbor's n′ for each node in connected graph G_{eks} is about four times as large as that in one hop connected graph G_{ks}. to achieve the same connectivity P_c as that of G_{ks}, the value of required for G_{eks} is one fourth of the value of required for G_{ks}. Thus, the value of p_{actual} for G_{eks} is one fourth of the value of p_{actual} for Gks. As we have already shown, when τ is fixed, the larger the value of ω is, the smaller the value of p_{actual} is. For example, assuming a network size N = 10, 000 and the desirable connectivity = 0.99999, if we fix τ = 2, we need to select ω = 7 for the G_{ks}-based key agreement scheme; however, using G_{eks}-based scheme, we can select ω = 31. The security of the latter scheme is improved significantly. there is about 4 times security improvement of the two-hop-neighbor scheme over the basic 1-hop-neighbor scheme.

Attacks:

It undergoes replication attack that is when the key space is broken by compromising λ keys adversaries can generate all the pairwise keys in that space and keys in that space can no longer be used for authentication purposes.

2.2 Heterogeneous Key Management Schemes:

In a heterogeneous sensor network, two or more different types of nodes with different battery energy and functionality are used. The motivation being that the more complex hardware and the extra battery energy can be embedded in few cluster head nodes, thereby reducing the hardware cost of the rest of the network. However using the cluster head nodes means that role rotation is no longer possible. When the sensor nodes use single hopping to reach the cluster head, the nodes that are farthest from the cluster heads always spend more energy than the nodes that are closer to the cluster heads. On the other hand when nodes use multi hopping to reach the cluster head, the nodes that are closest to the cluster head have the highest energy burden due to relaying . Consequently there always exists a non-uniform energy drainage pattern in the network. Two tree based heterogeneous schemes are discussed in following sections.

2.2.1 Star-like tree-based key pre-distribution schemes:

A Star-like tree-based key pre-distribution scheme was proposed by Lee and Stinson (2005)[9] which were proposed to improve the resilience against node capture. The difference between these scheme and the random pre-distribution schemes is that they are based on strongly regular graphs and random graph correspondingly. A strongly regular graph with parameters (n, r, λ, μ) is a graph on n vertices, without loops or multiple edges, regular of degree r (with $0 < r < n - 1$), and such that any two distinct vertices have l common neighbors when they are adjacent, and m common neighbors when they are nonadjacent. A complete bipartite graph $K_{n,n}$ is a(2n, n, 0, n) strongly regular graph.

The first scheme is called basic ID-based on-way function scheme. This scheme uses a public one-way hash function h in order to reduce the number of keys stored in a node. In this scheme, the connected regular graph G of order n and even degree r can base on network graph G to construct a key pool $K= K_V: v \epsilon G$. Each sensor node is assigned a unique ID for computing secret key. For a sensor node u, it will be allocated a secret key K_u and hashed keys $h (K_V || ID (u))$ if it is contained in a star-like sub graph centered at v. Since a node v can compute $h(K_V || ID(u))$ by evaluating function h at the concatenation of its unique key Kv and ID(u), both u and v can establish their secret key $h(K_V || ID(u))$.

Benefits:

- One advantage over the pre-distribution schemes is that it reduced the number of keys per sensor by almost 50%.This is because each node v stores one secret key K_v and r=2 hashed keys for the node u such that v is contained in a star-like sub graph centered at u. Therefore, the total number of keys stored in a sensor node is given by r/2+1.

- This scheme also has perfect resiliency. This is because if an adversary compromised a node and obtained K_u as well as $h(K_V || ID(u))$ for r=2 adjacent nodes V_i, it is infeasible to compute K_{vi} even though he knows

the key h(K_V || ID(u)) since h is a one-way function. Therefore, an adversary cannot compromise any link between two non-compromised nodes.

Flaws:

- The shortage of the basic ID-based one-way function scheme is that it can accommodate only O(k) sensor nodes for the node storage of k keys. Thus, it is not suitable for large size network. To meet the requirement for large size sensor network, multiple ID- based one-way function schemes are proposed based on the basic ID-based one-way function scheme at the cost of weakening the resiliency.

2.2.2 Logical tree-based key pre-distribution schemes:

Wallneretal. (1999)[10] proposed a group key management algorithm using the hierarchical binary tree (HBT). In this approach, only one group controller maintains a tree of keys, where each node corresponds to a KEK (key encryption key). Each group member corresponds to a leaf of the tree, and holds a node's KEK from its leaf to tree root. The group key is the key held by the root (Fig. 1).

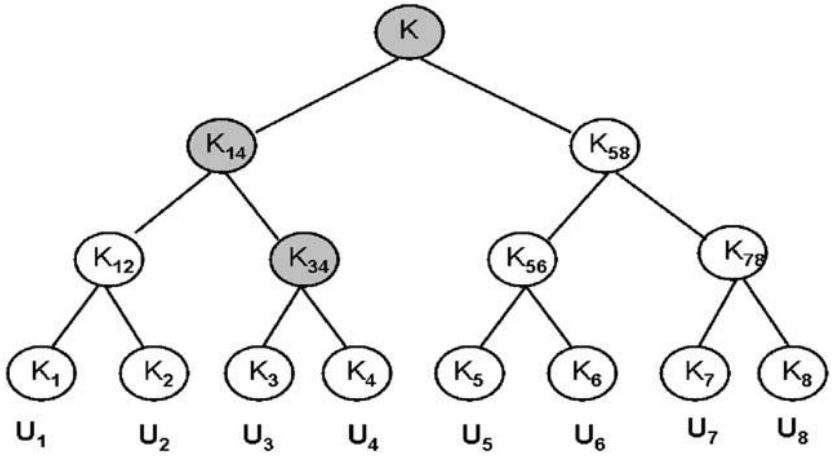

Figure 2.2: Hierarchical Binary Tree

For a balanced tree, each member needs to store $\log_2 n$ keys; here n is the number of members and $\log_2 n$ is the height of the tree. For example, see Fig. 1,

where member U3 knows K3, K34, K14 and K, the number of members is 8 and the height of the tree is 4. When a new member joins the group, a new leaf is attached to the group tree. All the KEKs in the nodes from the new leaf's parent to the root have to be changed for backwards secrecy .Each of the new KEKs is encrypted with its respective node's children KEKs, and then sent to the related users. The size of this re-key message will be at most $O(2\log_2 n)$. When a member is removed from the group, this process is similar. This logical key tree technique for multicast key distribution has been extended to wireless sensor networks in Di Pietroetal. (2003) [11] and Lazos and Poovendran (2003,2002) [12]. This technique is grouped as the centralized group key management protocol. Centralized solutions are often not ideal for wireless sensor network. However, such a technique offers some utility to allow a more powerful base station to offload some of the computations from the less powerful sensor nodes. In [11] proposed a directed diffusion-based secure multicast scheme for wireless sensor network(LKHW). This scheme merges the logical key hierarchy with directed diffusion and takes the advantages of both of them. Directed diffusion is a data-centric, energy saving dissemination technique for wireless sensor networks (Intanagonwiwatet al., 2000)[13]. In directed diffusion, the human operator's query would be transformed in to an interest. The interest is then diffused throughout the network nodes (called source).The node will activate its sensors to collect information. The dissemination sets up gradients designed to draw data matching the interest. The collected data can be sent back to the originators (called sink) along multiple paths. The sensor network reinforces one, or a small number of these paths. In this LKHW scheme, secure group has to be established with group initialization before directed diffusion. Here, the security of query is supposed as same as the security as the data transferred. The protocol is described as follows:(1) The key distribution center (KDC) sends out ''interest about interest to join''.(2)The interested nodes reply with "interest to join''.(3)The KDC supplies key set and then secure interest and data encrypted with the group key.

For dynamic groups, there are two protocols for the leave and join. When a node applies to join, a join ''interest'' is generated which travels down the gradient that have previously established by ''interest about interest to join''.

When a node joins, a key set is generated for the new node based on keys within the key hierarchy. The similar process for the leave is also given.

Benefits:

- This type of scheme is data-centric and is energy saving dissemination technique for wireless sensor networks.
- A lot of attention is paid toward security by using KEK.

Flaws:

- Whenever a node is compromised a lot of messages are exchanged to update KEK and as there is only one group controller so if that is compromised the entire network suffers

2.3 Comparison

After discussing different types of symmetric key management scheme we can finally derive a comparison table that will enable us to choose a scheme as per requirement and resource present.

Table 2.1:

Comparison table of different types of key management schemes of WSN

	Computation	Communication	Storage
entity based schemes	Depends on Network size	1X1	1X1
Random key pre-distribution schemes	Depends degree of a node	dX1	dXk
Pair wise key pre-distribution	Depends on Network size	1X1	2N
polynomial-based key pre-distribution schemes	Depends on Network size	dX1	dXk
matrix-based key pre-distribution	2(λ)	2 hops	(λ + 1)τ
Hierarchical binary tree-based key pre-distribution schemes	$2\log_2 n$	$\log_2 n$	$\sum_{i=0}^{h} 2^{l}$

CHAPTER III
RELATED WORKS

So far most of the key distribution schemes in wireless sensor networks we have studied through our survey assume that there is the need for every sensor node to be able to securely communicate with every other node from the network. This is a very strong assumption that might not be realistic in a real world sensor scenario. In case of the random pre-distribution schemes every sensor node receives a huge subset of an even larger set of pool-keys from the user. If two nodes want to communicate they need to have at least one common key in their subset. In such a way a lot of memory is wasted, which is especially critical for low memory sensor devices.

Another assumption that most of the pre-distribution schemes makes is that the base station will be responsible for every key exchange which is an unrealistic assumption. Because in a real world sensor network, this base station might not be available at all times, especially not for each and every key exchange.

From our survey we found that, homogeneous ad hoc networks have poor performance and scalability. Furthermore, many security schemes designed for homogeneous sensor networks suffer from high communication overhead, computation overhead, and high storage requirement. But recently deployed sensor network systems are increasingly following heterogeneous designs.

The performance and the lifetime of WSN can be improved greatly by using heterogeneous sensor nodes instead of homogenous ones without significantly increasing the cost. That's why we have studied and analyzed few heterogeneous sensor networks as part of our thesis work. We have studied four heterogeneous key management schemes as our related work and found out the flaws and benefits. We will be discussing about these schemes in the next few sections of this chapter.

3.1 TLA: A Tow Level Architecture for Key Management in Wireless sensor Networks

Boushra Maala, Hatem Bettahar and Hatem Bettahar [14] proposed a two level architecture for key management in WSN called TLA.

They organized their proposed scheme into two layers. The first layer has only Normal nodes called N_n and the second layer has super nodes called S_n. To become a super node a sensor node should have several properties. It should have enough energy resources to enhance its life time as long as possible. The super node will need a wide communication range to cover maximum number of Nn nodes in the sensor network and high processing capacity to facilitate processing of achieved data.

Any Normal node or N_n in WSN periodically pick up sensing information and send them towards one of the Sn (Super node) in the second layer. The duty of a Super node is to collect sensing information from a set of Nn nodes, and then process these data's to achieve the intended results. Super nodes may also cooperate with each other to combine these calculated results.

The authors in [14] proposed that the communications between normal nodes are also possible if the Sn nodes take the responsibility of conveying information between any two Nn nodes. Sn nodes also communicate with the central sink node to periodically send their final results.

In the Key Distribution Model of this scheme the authors proposed to use a limited pairwise key scheme at the first level between Nn nodes and Sn nodes and a complete pairwise key distribution at the second level between Sn nodes.

The scheme in [14] required that before deploying the nodes in the WSN, the right number of Sn nodes should be calculated in order to optimize the key storage overhead (KSO). They considered a WSN with N sensors divided into N_s Super nodes and N_n Normal nodes. It has been shown that In TLA, each N_n sensor stores only one key. On the other hand, a S_n sensor stores (n_s −1).

pairwise keys shared with other S_n sensors in the network and $(N-n_s)/n_s$ keys shared with its supervised children. As a result the number of stored keys in a S_n sensor is:

$$K_{Sn} = \frac{N}{n_s} + n_s - 2 \quad (3.1)$$

At ns = √N this function gets its minimum. So with √N Sn nodes in the WSN, the average number of stored keys by a Sn sensor is:

$$K_{Sn} = 2\sqrt{N} - 2. \quad (3.1)$$

After the Sensor nodes are configured properly and initialized with the right keys the sensors establishes secure channel with its neighbors. Each Sn sensor broadcasts a hello message which contains all the pairwise keys identifiers of its N_n children. When an Nn sensor receives a hello message, it verifies whether its key identifier is listed within the message or not. If the key identifier is listed, the Nn sensor responds back with an Acknowledgement (ACK) message and a secure channel is established using the shared pairwise key. To establish a Sn to Sn secure channels a full pairwise key scheme is used between all Sn sensors in the WSN. Super nodes exchanges hello and ACK messages between them to establish Sn to Sn secure channels.

The analysis in this paper [14] shows that this scheme gives a good resistance degree compared to other key management schemes. By using a limited pairwise scheme at the N_n level, whenever an N_n sensor is compromised, only the concerned channel between this Nn sensor and its Sn parent will be revealed to the attacker. In the same way, when a Sn node is compromised it will have a localized impact since only the secured channels from and to this Sn node will be compromised. Another advantage of the TLA scheme is the optimized key storage overhead.

Performance Evaluation of TLA

For Sn nodes, each node should store $2\sqrt{N}-1$ pair-wise keys to communicate with the Nn nodes in its group and with other Sn nodes in the WSN. However, for Nn nodes, each node needs only to store one key which is a pair-wise shared key to communicate with its Sn node.

The Resistance Degree against node capture (RD) of a key distribution scheme is the ratio of non compromised links over the total network links when one sensor node of the WSN is compromised:

$$RD = 1 - \frac{NCL}{NTL} \tag{3.3}$$

Where, NCL is the number of compromised links when one node is compromised, and NTL is the total number of available links in the network.

A link between two sensor nodes is considered as compromised if the associated encryption key is revealed. A scheme with a good resistance degree against node capture will have RD ≈ 1 meaning that capturing one sensor node will not compromise any secured channel in the WSN. This is the case of pair-wise schemes. In contrast, a scheme with a very bad resistance degree against node capture will have RD ≈ 0 meaning that capturing one sensor node will compromise all secured channels in the WSN. This is the case of Master key schemes.

In TLA scheme, the total number of channels includes all pair-wise channels between all Sn nodes in the network and all pair-wise channels between each Sn node and Nn nodes of its cluster. As a result the total number of Link for TTL is:

$$NTL = \left(N - \sqrt{N}\right) + \frac{\sqrt{N\left(2\sqrt{N}-1\right)}}{2} \tag{3.4}$$

When a Sn node is compromised, all its pair-wise channels with Nn nodes inside its cluster and its pair-wise channels with other Sn nodes will be revealed.

On the other hand, when an Nn node is compromised only one channel will be compromised.

This makes an average of compromised channels:

$$NCL = 3.\sqrt{N} - 2. \quad (3.5)$$

Dividing this value by the total number of channels gives:

$$RD = 1 - \frac{3\sqrt{N}-2}{N\left(2\sqrt{N}-\frac{3}{2}\right)} \quad (3.6)$$

Comparing the Resistance Degree of their proposed scheme against three combined technique scheme like LEAP, ESA and EKMSH the authors showed that TLA gives a better resistance degree against node capture compared to all others. They also showed that TLA as the only scheme that gives a very good resistance degree against node capture while using a small key storage overhead.

3.2 A Tree Based Approach for Secure Key Distribution in Wireless Sensor Networks

This paper proposed by Michael Conrad, Erik Oliver Blaß and Martina Zitterbart [15] presents a new key distribution method in sensor networks. It shows that communication in sensor networks follows a certain tree-like scheme. They called this process aggregation. This paper [15] showed that if the sensors follow a tree like hierarchy it can not only be memory efficient and energy saving but also a secure key distribution is possible. On the other hand when new sensor nodes will join the network they will be able to autonomously share the keys they need to complete their operation. Unlike other works in wireless sensor

networks, this paper states that there is no need to distribute keys between random sensor nodes because in real-world sensor networks often communicate like a tree-like aggregation towards the sink.

The authors of these papers [15] proposed that sensors nodes in a WSN can be divided into two types, normal nodes and aggregation nodes. Sensor nodes measure data and forward them towards a data sink. On the way to the sink data can be aggregated by so called aggregation nodes. These nodes are able to collect data from other sensors nodes and process them, for example computing a mean value and forward the aggregate to the sink.

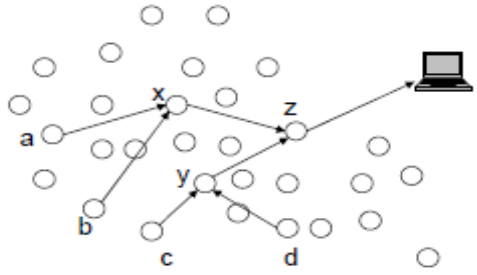

Figure: 3.1:Aggregated sensor structure

Figure 3.2.1 shows an example network. Sensor nodes a and b measure the temperature in room 1 at different positions. And nodes c and d measure the temperature in room 2 respectively. The sink is however only interested in the mean temperature of the complete building. Therefore a tree-like scheme has to be established for sensor communication. Aggregation node x collects temperature measurements from nodes a and b and computes their mean value forwarding this to aggregation node z. Aggregation node y does the same for node c and d. Finally node z computes the mean temperature for the whole building, i.e. two rooms, and reports it to the sink. This communication scheme

forms a hierarchy, sensor nodes (vertexes) and communications paths (edges) form a graph, more precisely a tree.

We can observe from the above scenario that sensor nodes do not have to communicate randomly with each other. Sensor node needs to exchange data only with its parent node in the tree structure. Therefore, it needs a shared key only with its parent or aggregation node. On the other hand Communication is unlikely to happen between nodes from other categories or between nodes within the same category. If needed, they might transport or forward data in multi-hop situations but there is no need for any end-to-end communication.

Key distribution

The authors of this paper [15] assumed that aggregation in sensor networks forms a complete binary tree. Before a new sensor joins the network, it must be paired by the user or a Master Device. The pairing is essential for the node to obtain its new position inside the aggregation tree, to identify the parent or the first aggregation node.

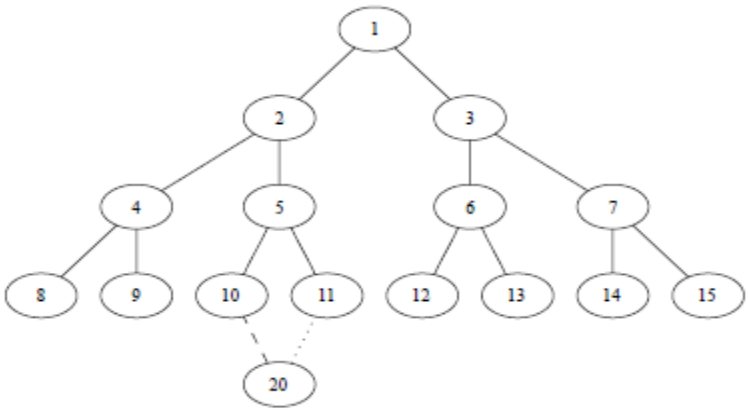

Fig: 3.2 A sample aggregation tree

Figure 3.2.2 shows a complete binary tree exists before a new node starts to join. This node is now paired by the user.

The user identifies, that this node will have to communicate with aggregation node 10 because of its use. For this the user assigns the new node a new ID node. As the new node will be a child of 10, ID node could be 20 or 21 to comply with the binary tree scheme. As the new node is the first child of 10, it becomes 20. Now, 10 is the primary parent (p1) of 20. Furthermore the user computes a secondary parent P2 for 20 using:

$$P2 = \begin{cases} \frac{IDnode}{2} + 1 & \text{if IDnode is even} \\ \frac{IDnode + 1}{2} & \text{if IDnode is odd} \end{cases} \quad (3.7)$$

P2 for 20 is therefore node 11.The user cans now handout two tickets (including keys) to 20 that allow secure communication for 20 with 10 and 11. This can be done efficiently as each node in the network might share a pairwise different key with the user, thus allowing the user to securely access distinct nodes.

As 20 can now establish secure channels to parents 10 and 11, it will ask them, which other aggregation nodes are on the way to the sink 1. Even in the presence of one cheater 20 will come to know 5, 2 and 1. The next step is to build secure channels to these nodes by securely exchanging shared secrets with them, first of all with 5. The main idea is, that 20 generates a new key K and splits it into two parts K1 and K2, these parts are encrypted with the key for 10 or 11 respectively and sent to 10 and 11 to forward them towards 5. As K1 is encrypted from 20 with the key shared with 10 only 10 can decrypt it. Then 10 encrypt K1 with the secret key 10 shares with 5 and send the result to 5. On the other hand 11 do the same with K2. Finally 5 can decrypt both transmissions from 10 and 11 and restore K. As neither 10 nor 11 come to know the other part of K, K is finally secretly transmitted from 20 to 5 even in the presence of one malicious node. Also changing K1 or K2 maliciously would not help any node as this would only deny communication between 20 and 5, but impersonation attacks are not possible. To secure communication between 20 and 2 or 1, the same procedure can be repeated. 20 sends one half of the encryption key K to

10, the other one to 11. As the aggregation tree was build inductively both have already a secure channel to node 2 and can transfer their part of K to node 2 directly (10s parents were 5 and 6).

An analysis of this mechanisms complexity shows, that each setup of a secure communication channel only needs 4 symmetric encryptions and 4 communication steps inside aggregation overlay, completely independent from the position of the new node inside the tree or the depth of the tree. Also each node has to store only keys from other nodes, which are absolutely necessary because of its mission.

3.3 An Efficient Key Distribution Scheme for Heterogeneous Sensor Networks

Sajid Hussain, Firdous Kausar and Ashraf Masood [16] proposed a Heterogeneous Sensor Network consists of a small number of powerful High-end sensors (H-sensors) and a large number of Low-end sensors (L-sensors) in which the more powerful H-sensors act as cluster heads (CHs).The advantage of Clustering of sensors is that it enables local data processing, which reduces communication load in the network as well as provide scalable solutions.

Key Pre-Distribution Phase

This scheme in [16] uses a key pool K consists of M different key chains. A key chain C is a subset of K. Each key chain is generated independently via a unique generation key and publicly known seed S by applying a keyed hash algorithm repeatedly. Publicly known seed value is same for every key chain.

$$K = C_o|C_1| \ldots |C_{M-1} \tag{3.8}$$

The nth key of the key chain Ci is computed as:

$$k_{Ci,n} = HASH_n(S, g_i) \tag{3.9}$$

The total number of keys in a key chain is N, where N =K/M.

Before deploying the nodes, each node is loaded with its assigned key ring R, where R is the generation knowledge of a number of key chains. Each L-sensor node is assigned with r randomly selected generation keys of corresponding key chains. From these r generation keys, r×N random keys can be calculated effectively. Each H-sensor node is pre-loaded with S randomly selected generation keys of corresponding key chains, where S >> r.

Cluster Formation Phase

During the cluster formation phase, all H-sensors broadcast Hello messages to nearby L-sensors .The Hello message include the ID of the H-sensor. The transmission range of the broadcast is large enough so that most L-sensors can receive Hello messages from several H-sensors. Then each L-sensor selects the H-sensor whose Hello message has the best signal noise ratio (SNR) as the cluster head. Each L-sensor also records other H-sensors from which it receives the Hello messages, and these H-sensors are listed as backup cluster heads in case the primary cluster head fails. The H-sensor acts as a cluster head (CH), and the L-sensors act as cluster members.

Cluster head based Shared Key Discovery Phase

After cluster formation is done the shared key discovery phase begins. *E*ach cluster member sends a message to its cluster head, which includes its ID, the ids of the generation keys, and its neighboring nodes information. Some L-sensors may not share any pre-loaded generation key with their neighbors. For each pair of L-sensors that do not share any generation key, CH generates a pair-wise key for each pair (X and Y), and securely sends the key to them.

Performance Evaluation

This paper in [16] proposed a key distribution scheme for heterogeneous sensor networks based on random key predistribution. In this scheme, in place of storing all the assigned Keys in a sensor node, they stored a small number of

generation keys. As a result it can significantly reduce the storage requirements as compared to other random key pre-distribution schemes. The analysis In this paper showed that requirements can be reduced by 8 times as compared to AP [17], and 33 times as compared to basic scheme [18].

If p is the probability that an L-sensor and H-sensor share at least one common key in their key ring, then the number of possible key ring assignment for an L-sensor is:

$$\frac{M!}{r!(M-r)!} \tag{3.10}$$

The number of possible key ring assignment for an H-sensor is

$$\frac{M!}{S!(M-S)!} \tag{3.11}$$

The total number of possible key ring assignment for an L-sensor and H-sensor is

$$\frac{M!}{r!(M-r)!} \times \frac{M!}{S!(M-S)!} \tag{3.12}$$

The probability that an L-sensor and H-sensor share a common key can be given as

$$p = 1 - \frac{(M-r)!(M-S)!}{M!(M-r-S)!} \tag{3.13}$$

The analysis of this paper showed that where basic scheme needs 100 keys[5] and AP scheme needs 20[17] the same probability of key sharing among nodes can be achieved by just loading 2 generation keys in sensor node for this scheme.

For instance, if there are 1000 L-sensors and 10 H-sensors in an HSN, where each L-sensor is pre-loaded with 2 generation keys and each H-sensor is pre-loaded with 100 generation keys, the total memory requirement for this proposed scheme is 2×1000+100×10=3000 (in the unit of key length). However, in AP scheme [17], if each H-sensor is loaded with 500 keys and each L-sensor is loaded with 10keys, the total memory requirement for storing these keys will be 500×10+1000×20=25,000, which is 8 times larger than our proposed scheme. Further, for a homogeneous sensor network with 1000 L-sensors, where each L-sensor is pre-loaded with 100 keys, the memory requirements will be 100×1000=100,000, which is 33 times larger than our proposed scheme.

Security Evaluation

In this scheme, If there are n compromised nodes, the probability that a given key is not compromised is (1− r M) n. The probability of total number of compromised keys, where n number of L-sensors is captured, is as follows: has knowledge of r×N keys. The probability that

$$p = 1 - (1 - \frac{r}{M})^n \quad (3.14)$$

For a given parameters: M=1000, K=50,000, r=5, and m=100, the results show that when the compromised communication is 100 percent for basic scheme, the proposed scheme has compromised communication of only 12 percent.

3.4 LEAP: Efficient security for large-scale WSNs

Hierarchical Key management for WSN is often known as LEAP. This scheme was proposed by Zhu, Setia, and Jajodia [19]. The authors of this scheme believe that different types of messages exchanged between nodes need to have different security requirements .The packets received by a node should always be authenticated and the packets transmitted by a node should always be encrypted to meet the security requirements. Zhu, Setia, and Jajodia

[19] used four types of keys in Leap to handle different types of packets. There four types of keys that must be stored in each sensor: individual, pairwise, cluster, and group.

Individual key:

A unique key that is shared between the base station and each sensor node is called Individual key. Sensor nodes use this key to calculate the MACs on their messages to the base station like alert signals. In the same way, a base station can use it to send messages to each and every node in the network.

Pairwise shared key:

This is a unique key which is shared each node and its neighboring node. A node can use it to transfer individual messages like sharing a cluster key or sending data to an aggregator node.

Cluster key:

This is a key that is shared between a node and its neighboring nodes. A node may decide not to send a message to the base station if its neighboring node is sending the same message with a better signal. This discovery is only possible to implement if a node shares a common key with its neighboring nodes. With such a cluster key, a node can select which messages to transfer which can reduce the communication overhead.

Group key:

The base station shares this key with all the nodes in the network to send queries to them. Group key used requires an efficient rekeying mechanism for updating it as there is a chance for an adversary to know the key whenever a node is compromised.

Establishing individual keys:

Before a node is deployed in the sensor network each node is pre-loaded with a unique key that it shares with the base station. For example, the individual key K_u^m for node U is calculated as:

$$K_u^m = f(K^m(u)) \tag{3.15}$$

Here, f is a pseudo-random function and K^m is the master key known only to the controller or base station. The base station do not need to store all the individual keys, because the base station can generates them on the fly whenever it attempts to communicate with a node. The base station can give these keys to the individual nodes.

Establishing pairwise shared keys:

Pairwise key is shared between each node and its neighbor. There should be a way so that the neighbors can identify each other when deployed in the network as they do not have any pre-deployed information. Whenever a node is deployed in a WSN, it requires some minimum time to identify neighbors and establish keys with them, which will be test. It is assumed that the node cannot be compromised before that tome.

There are four stages that represent the key establishment of new node U deployed in the network:

- key pre-distribution
- neighbor discovery
- pairwise key establishment
- Key erasure

During the initial stage of key predistribution, node U is loaded with the key K_i by the controller and derives the master key K_u using it.

For neighbor discovery, node U first initializes a timer to activate at timer, t_{min}.Then it starts communicating with its neighbors by broadcasting a HELLO message containing its ID. Node V responds to this message with a reply containing its own ID. The acknowledgement (ACK) of V is then authenticated using its master key Kv derived from Ki. Node U verifies the authentication of V by generating the master key Kv as node V shares Ki with it:

$$U \text{ ->*: } U \text{ and } V \text{ -> } U\text{: } V, \text{MAC } (K_v, U|V) \qquad (3.16)$$

For the third stage of pairwise key establishment, node U computes the pairwise key K_{uv} with node V using V's identity. Node V can also do the same thing with U. There is no need for authenticating node U to V as any future messages authenticated with K_{uv} will prove node U's identity.

In the fourth and final stage is key erasure, where node U erases K_i and all the master keys of the other nodes after the time expires. Then node U will not be able to establish pairwise keys with any other nodes in the WSN so that, though an adversary captures a node, the communications between it and another node cannot be decrypted without the key K_i.

Establishing cluster keys:

The cluster key establishment is based on the pairwise key establishment. If node U wants to establish a cluster key with its neighbor's v_1, v_2, $v_{3,\ldots\ldots}$,v_n, first it generates a key K_c and then encrypts that key using the pairwise key which it shares with each neighbor. Node U then transmits this encrypted message to its neighbors. Node v_1 decrypts the key using the pairwise key which it shares with U, and then stores the key in a buffer. Next it sends back its own cluster key to node U. When any of the nodes are revoked, node U generates a new cluster key in the same way and transmits the key to all remaining nodes.

Establishing group keys:

A group key is shared between a base station and all the nodes in a WSN. When the base station wants to send a message or query to all the nodes of that WSN it uses the group key.

One way of broadcasting messages by the base station can be done by using the hop-by-hop method in which the base station can encrypts messages using the cluster key which it shares with its neighbors and then broadcasts the message to all the nodes in its neighborhood. The nodes would decrypt the message and then encrypt it using the cluster key which they share with their neighbors. In this way, the message can be received by all the nodes in the network. This is efficient, but has an overhead of encryption and decryption at every node.

A simple method to establish a group key is to preload each node with the group key before deployment, but this is still within the scope for rekeying the group key which will be necessary. Unicast-based group rekeying can also be considered for which the base station needs to send the group key to each node in the network, but this involves much communication overhead. However, Zhu, Setia, and Jajodia [19] proposed an efficient scheme based on cluster keys in which the transmission cost will only be one key.

Local broadcast authentication:

In local broadcast a node generally does not know what packet it is going to generate next and messages generally consist of aggregated sensor readings or routing protocols. In case of local broadcast authentication is needed immediately. For local broadcast, One-Way Key Chain-Based Authentication is used. This scheme is based on µTESLA in that each node generates a one-way key chain and sends the commitment of it to their neighbors. This Transferring is done using the pairwise keys already shared with neighbors. If a node wants to send a message to its neighbors, it attaches the next authorization key from its key chain to the message. The receiving node can verify the validation of the key based on the commitment it has already received. The One-Way Key Chain-

Based Authentication is designed based on two observations: a node only needs to authenticate to its neighbors and that a node V will receive a packet before a neighboring X receives it and resends it to V. This observation is true because of the triangular inequality among the distances of nodes involved. An adversary may still try to attack the nodes by shielding node V while U is transmitting a message, and then later send a modified packet to V with the same authorization key; but this attack can be prevented by combining the authorization keys with the cluster keys. When this is done, the adversary does not have the cluster key and so cannot impersonate node U. However, this scheme does not provide a solution for attacks from inside where the adversary knows U's cluster key.

Evaluation if the Scheme

Computation cost:

Computation cost depends on the number of encryption and decryption. If the size of network is M, total number of encryption is M and total number of decryption is M and if the density of network is d, in [6] it has been stated that the average number of symmetric operations of the scheme is about

$$2(d-1)^2/(N-1)+2 \qquad (3.17)$$

Communication cost:

The communication cost also depends on the density of network. The average communication cost is $(d-1)^2/(N-1)+2$ for this scheme.

Storage Cost:

The storage requirement of LEAP depends upon the density of the network. The storage requirement of this scheme is a bit high because each node must store four types of keys in it. Considering the degree of node to be d, a node has to store one individual key, d pairwise keys, the cluster keys, and one group key. Also, a node must store a one-way key chain and a commitment for each neighbor for local broadcast. If L is the number of keys stored in a key

chain, the total number of keys the node has to store in this scheme will be 3d + 2 + L.

Security:

LEAP is an efficient scheme for key establishment that resists many types of attacks on the network, including the Sybil attack, sinkhole attack, wormhole attack, and so on. LEAP also provides efficient schemes for node revocation and key updating in WSNs.

Benefits of this scheme:

- LEAP supports various communication patterns, including unicast (addressing a single node), local broadcast (addressing a group of nodes in a neighborhood), and global broadcast (addressing all the nodes in a WSN).

- LEAP provides survivability such that compromising of some nodes does not cede the entire network.

- LEAP is energy efficient since it supports techniques like In-network Processing and Passive Participation that greatly reduce network communication overhead and, in turn, increase node battery life.

- In final stage of key establishment, $_{I}$,master keys of all node are erased after t_{min} time expires so that even if a node gets captured by adversary, the communication between it and other nodes can't be decrypted without the key $K_{i.}$

- One of the unique advantages of the scheme above is that once pairwise keys are established between neighboring nodes in an area of a WSN, they cannot be established again, which protects the network from clone attacks.

- There is no need for the base station to store all the individual keys, because the base station generates them on the fly whenever it attempts to communicate with a node.

Flaws of the scheme:

- Disadvantages of this scheme include that it requires excessive storage with each node storing four types of keys and a one-way key chain

- Computation and communication overhead dependent upon network Density (the denser a network, the more overhead it has).

CHAPTER IV
OUR PROPOSED SCHEME

4.1 Introduction To Our Scheme

In our thesis, we are proposing a tree based key management scheme for Heterogeneous WSN, often known as HSN. A HSN consists of a small number of powerful high-end sensors (H-sensors) and a large number of low-end sensors (L-sensors). H sensors are more powerful nodes with more storage capability, computation, communication and energy supply. L-sensors are ordinary sensor nodes with limited computation, communication, energy supply and storage capability. Sensor nodes are assumed to be immobile; these nodes organize themselves into clusters. The size of the cluster we are assuming here is a small group of sensor nodes. The size of cluster depends on the network density.

Here, hierarchical architecture of sensor networks is considered, where data is routed from sensor nodes to base station through cluster heads. Cluster head are H-sensors who are responsible for its cluster's security which comprises of many L-sensors.

In such network the role of Base station is to interfaces sensor network to the outside network. A cluster head is chosen from each cluster to handle the communication between the cluster nodes and the base station. Our proposed scheme proposes method that talks about different issues like addition of new nodes, key renewal when a node is compromised and key refresh at regular intervals in order to achieve key freshness.

In this scheme well established security is achieved as it's our main priority. In this tree based key management schemes each user shares a key called private key with the key server and key at the root of the tree is the group key which is shared by all users in the group. Other keys (other than private key and group key) are called auxiliary keys which are known only for certain subset of users and are used to encrypt new group key whenever there is a group

membership change. Overall all the keys are referred as Key Encryption Keys (KEK). Whenever a node is compromised new group key (PK') is distributed to other nodes using one way hash functions and simple XOR operations.

Our scheme uses m-ary tree and at each level m KEK are maintained. Since m-ary tree is used we are reducing number of levels of the tree which reduces storage of each sensor. Other schemes like the scheme of I.Chang, R.Engel, D.Kandlur, D.Pendarakis and D.Daha (1999) [20] has storage of O ($\log_2$ N) .In our scheme at every level we are maintaining only m keys so which reduces server side storage to O ($\log_m$ N). In [20] binary tree structure is used. When the cluster size is large, the number of levels in the binary tree will be more which increases number of keys to be stored by each sensor node. Extending the scheme to m-ary tree will reduce the height of the tree reducing number of keys at each sensor node. Since encryptions are replaced by hash functions and simple XOR operations, computation and communication cost incurred will also be reduced when compared to exsisting scheme in [20].

4.2 Key Establishment Between H-sensor And L-sensors

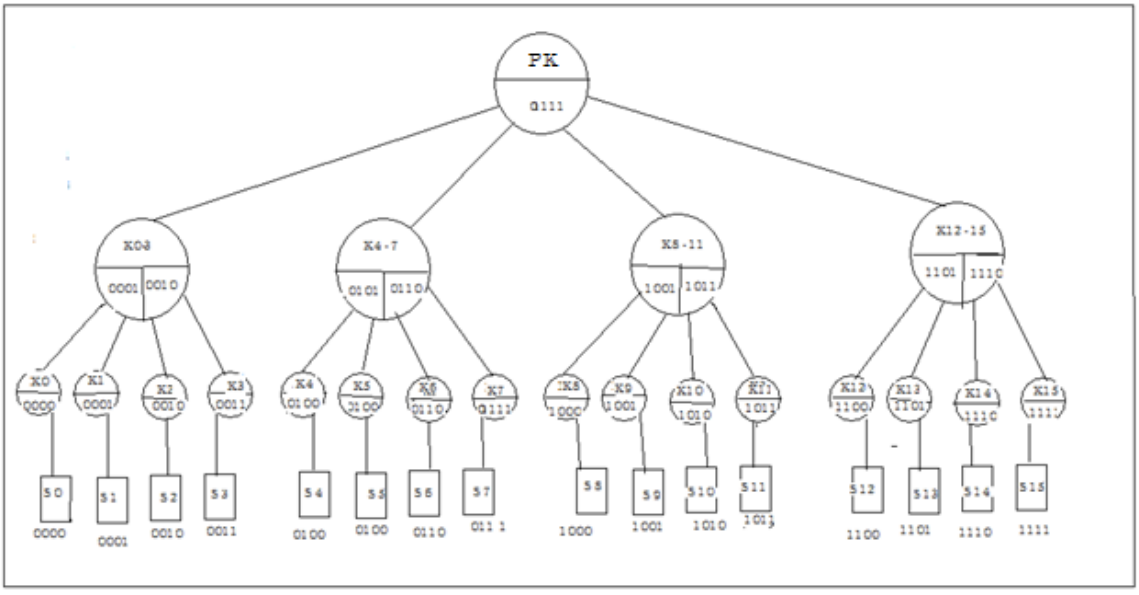

Figure 4.1: B+ m-ary tree of key encryption key (KEK)

Initially all H-sensors are pre loaded with their own ID, function f (used to authenticate L sensors) and initial key K_i. And L-sensors are preloaded with their own ID, function f (used to authenticate cluster head), and initial key K_i . The function f is used for key generation and authentication purpose hence it's a common function between all sensors.

After deployment all H-sensors broadcast an encryption advertisement message (adv) (encrypted using K_i) containing its ID and random key (ID_{CH}, K_{CH}) to inform the whole network. Only L-sensors with K_i is only able to decrypt the message. Each genuine L-sensor will reply back to the H-sensor whose Hello message has the best signal noise ratio and select it to be its cluster head.

To join the H-sensor (named CH) with the best Hello message, L-sensor (named Ai) will use key function f and K_{CH} to generate K_{AiCH}. Then Ai will send a join message (join) encrypted with K_i .The message consists of a tuple as <ID_{Ai}. K_{AiCH}> to CH and make the message as its initialization key.

So we can say now that CH has received nodes Ai's join messages protected by K_i. At the same time, CH decrypts all the messages from all other L-sensors, willing to join its cluster , and collect all their ID and keys.Then CH creates index of binary string of length x ($x=\log_2 N_{Opt}$) and assign to each node. After that CH generates a B+ tree-based key structure using the dual-data, according to the sort of keys' value it assigns keys for each node in the tree. Finally creation of the B+ tree is completed in the CH-sensor. The degree of the B+ tree depends on the optimum size of network.Maximum degree of b+ tree for a preferred height of h is,

$h=\log_m N_{opt}$

$m^h = N_{opt}$ (N_{opt} is the optimum no. of nodes that can be deployed in an area)

$h\log_2 m=\log_2 N_{opt}$

$m=2^{(\log 2 Nopt)/h}$

For each node,Ai, CH sends a join-reply message, encrypted with K_i, back to Ai. The join-reply message consists of receiving nodes id and all the KEK

along the path from leaf to root of the B+ tree.Refering to the figure (4.1), CH sends < ID_{Ai}, K_0,K_{0-2}, PK> to Ai,where K_0 is private key,K_{0-2} auxilary key and PK is the group key. If we look into details, actually each of the keys(K_0,K_{0-2} and PK) is sent separately in messages encrypted by K_i.

After joining a cluster, an L sensor undergoes key erasure. That is it remembers its KEK, function f and it's ID. It no longer remembers the key K_i.

As we consider Base station to be an entity with huge resource and is fully in our control that is it cannot undergo any kind of attack. Thus to maintain proper control of the network, the entire network is divided into clusters. The size of cluster depends on total network density. The base station must have full access to each cluster. Rather than maintaining separate connection with each sensor, to reduce storage and communication overhead, Base Station keeps itself connects to each cluster head (H-sensors).Thus once the cluster is formed, the CH send the application messages (app) to BS to establish secure links between CHs and BS. And the second level B+ tree will be established simultaneity.

The second level key tree establishment is similar to the process as the first level key tree. Firstly, the cluster head sends join application message, encrypted by K_i, to base station and the message consists of a tuple as <ID_{CH}, K_{CH} > where ID_{CH} is the CH 's ID and K_{CH} is its initialization keys which will be exchanged. When BS receives the application messages, it generates the B+ tree similar to the first level key tree found in the cluster heads. Then, from the B+ tree key pool Group Key for Cluster Head CH, BCHK , along with other auxilary key will be sent to it by Base Station in an encrypted message.

In order to achieve key freshness, it is required for the Base Station to change PK and BCHK to PK' and BCHK' periodically or whenever needed. The cluster key PK is changed to PK' by respective cluster heads and is distributed securely to nodes in the cluster by hash function that will be soon described. Similarly base station will change BCHK to BCHK' and distributes it to all cluster heads securely by using hash function.

4.3 Key Establishment Among L-sensors

L-sensors can communicate with each other only after they have joined a cluster under a cluster head and obtained group key PK. Group key PK is the only key encryption key that is common between all nodes under a cluster head. Each L-sensor Lu can use PK and function f to generate its master key K_{Lu}=fPK(ID_{Lu}). And then node Lu broadcasts an advertisement message (ID_{Lu}, $Nonce_u$), encrypted using PK,which contains a nonce and waits for each neighbor Lv to reply. And then Lv replies to Lu with an encrypted message by PK and the message contains <ID_{Lv}, PK(ID_{Lv}|Nonce $_{Lu}$)>.

Simultaneously, Lv can also generate the key K_{Lv} = f PK (ID_{Lv}) .And then both nodes Lu and Lv can generate the pair-wise key K_{Luv}= fK_{Lv}(ID_{Lu}) . Each node can use these nodes' ID to calculate its neighbor nodes' key, that is, every node can be authenticated by its immediate nodes. If there are some malicious nodes, for example, the adversaries' nodes, they can be distinguished by this approach.

This part of the communication is mainly done to provide initial security. During the above mentioned authentication process, if nodes under a cluster determines presence of malicious node trying to enter cluster, then they will inform cluster head which in turn will update group key PK. The process of rekeying will be discussed in the following sections.

4.4 Key Renewal (referring to figure 4.1)

Since in our scheme sensor nodes are immobile, the two-level key establishment technique does not have to consider deployment knowledge of others before node deployment. When an adversary obtains a sensor node, it is assumed that the node cannot be compromised before time t_{min}. Whenever a node is deployed in a WSN, it requires some minimum time to identify neighbors and establish keys with them, which will be t_{est}. In our scheme we assume that $t_{min} > t_{est}$.

We assume that we have intrusion detection mechanism to detect node compromise. As soon as a node is compromised corresponding cluster head will change all the keys that are known to compromised node (i.e. all the KEK along the path from leaf to root of the B+ tree). The changed keys are distributed securely to existing nodes. For e.g. if say node s4 is compromised, keys k_{4-7} and PK are changed k_{4-7}' and PK'.

For key renewal our scheme follows a simple computation shown below:

F(auxiliary key, new group key) < − (Auxiliary key) XOR (New Group key) (4.1)

For each node we assume that it has the capability to compute a one-way hash function G as discussed in 'N.F.P.180-1. Secure hash standard' (1994) [21] and by R.Rivest.(1992) [22]. We also assume each node is able to update auxiliary keys after getting new group key using the function F as shown in equation (4.1). For nodes not along the path of compromised node needs to refresh only PK to PK' but nodes along the path of the compromised node needs to refresh all the keys that it had in common with the compromised node. Therefore referring to above diagram S5,S6 and S7 needs both PK' and K_{5-7}' and rest will be distributed with only PK'. Then the compromised node s4 will not share any common key with the entire cluster thus rest of the network is secure and remains connected.

We will now elaborate how new group key PK' and K_{5-7}' is distributed to the remaining group members (i.e., nodes) using the simple computation shown in equation (4.1).

At first the compromised auxilary key will be updated to K_{5-7}' .For s5 Cluster Head, CH, computes the hash of key k_5 i.e., $G(k_5)$ and XOR's this with new group key K_{5-7}' which yields K^{s5}<- $(G(k_5))$ XOR (K_{5-7}'). Upon receiving this message nodes s5 (knowing key k5) compute $G(k_5)$ and XOR's with K^{s5} to get new group key K_{5-7}' (i.e., K_{5-7}' <- (K^{s5}) XOR $(G(k_5))$).Similarly the new K_{5-7}' is distributed to s6 and s7.

To communicate new group key PK' to nodes s0, . . . , s3,CH computes the hash of key k_{10} i.e., $G(k_{10})$ and XOR's this with new group key PK' which yields $K^{s0,...,s3}$ <-($G(k_{0-3})$) XOR (PK') .Upon receiving this message nodes s0, . . . , s3 (knowing key k_{0-3}) compute $G(k_{0-3})$ and XOR's with $K^{s0,...,s3}$ to get new group key PK' (i.e., PK' <-($K^{s0,...,s3)}$ XOR ($G(k_{0-3})$)).

Similarly messages sent by CH to existing group members are $K^{s8,...,s11}$<-($G(k_{8-11})$) XOR (PK'), $K^{s12,...,s15}$<- $G(k_{12-15})$ XOR PK' and $K^{s5,...,s7}$ <- ($G(K_{5-7}')$) XOR PK'. The nodes will compute the new group key PK' by XORing received message with the hash of the keys known to them. Group key PK' computed by nodes s8, . . . , s11 is PK' <-($K^{s8,...,s11}$) XOR ($G(k_{8-11})$), for nodes s12, . . . , s15 it is PK'<-($K^{s12,...,s15}$)XOR ($G(k_{12-15})$) and , for node s5 to s7 PK'<- (K^{s5-7}) XOR($G(K_{5-7}')$). The keys that are known to compromised nodes s4 are k4, k4-7 and PK.

In the messages that are sent in clear by CH to group members by using the hash of the encryption keys, none of the messages uses any of the keys that are known to compromised nodes. Hence using the keys of the compromised nodes it is not possible to get any information regarding new group key.

In this method if two L-sensors with two different keys k_x and k_x' receive the same secret key in two individual communications, they can compute the hash of the other. For e.g., a node say Ux having k_x can recover $G(k_x')$ as follows : $G(k_x')$<- (($G(k_x')$) XOR (k_{new}))XOR(k_{new}) where k_{new} is known to the node Ux (secret sent in earlier communication) and '$G(k_x')$ XOR (k_{new})' is eavesdropped. Using this hash, i.e. $G(k_x')$, node Gx can decrypt the secret intended only for nodes having key k_x' that is sent in next communication. To avoid this every key that is hashed in the current communication is incremented by 1 and then used to take next hash value for next communication (i.e., k_x' <- k_x' +1) and then hashed to send next secret key. It is computationally infeasible for a user during the ith application of this method to recover $k_{x'+i}$ even given $G(k_{x'+i})$, . . . $G(k_s')$.

Therefore in order to avoid attackers decrypting any message in the next time interval we perform two operations. First, each remaining node along the

path from the leaving point will compute new auxiliary key using the equation (4.1). Second, every key used to compute the hash value is incremented by 1. In this scheme to communicate new group key securely we are not using any encryption instead all communications are by using hash values and XOR operations which will reduce the communication overhead i.e., rekeying cost is reduced.

Whenever a new node is added to network, the cluster head will find an appropriate position for the new node in the tree and tree is updated (i.e., all the keys along the path including the cluster key are changed). Cluster head will now distribute new PK' to previous nodes and the new node will receive all the keys along the path. In order to distribute the changed keys securely cluster head uses private key of the new node and for other nodes it uses previous cluster key.

So overall this is the scheme that we will like to propose as an enhancement of existing heterogeneous tree based key management schemes.

CHAPTER V
PERFORMANCE ANALYSIS

In this chapter, we have evaluated and compared the performance of our scheme of in terms of key Storage overhead, communication overhead, computation overhead and resiliency to attack.

5.1 KEY STORAGE OVERHEAD

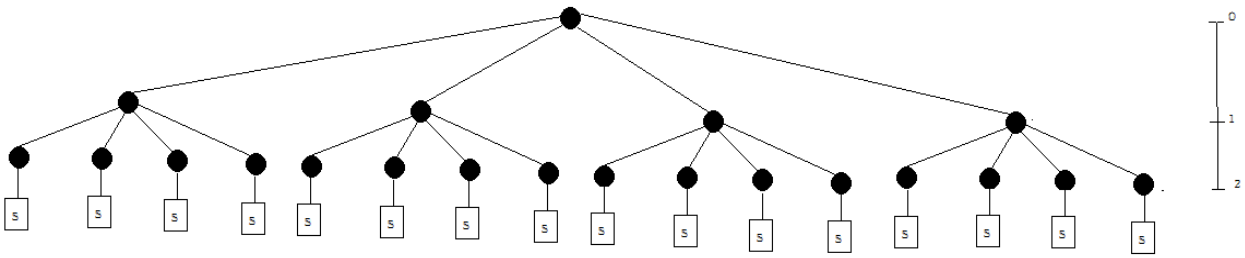

Figure: 5.1:key storage tree

Our proposed Scheme ensures strong authenticity through the key encryption keys we have introduced. The size of the pool of Key Encryption Keys is relatively so large that overhead of storing few functions and few other preloaded keys can be ignored. In our scheme, we define the Key Storage Overhead (KSO) as the number of keys stored by each sensor in the Wireless Sensor Network. Here, the key storage overhead depends on the type of the sensor node.

Since the H-sensors are nodes with higher capability they can store huge number of key's as required by the scheme. At the initial setup phase, after the cluster has been formed, each H sensor builds a B+ tree of KEK's. Storage in our system depends on the degree and height of the B+ tree. For H-sensor nodes each node need to store $\sum_{i=0}^{h} m^i$ number of keys. However, for L-sensor nodes,

each node needs only to store the key encryption key's which are on the path from leaf (position of the L-sensor node in the tree) to node of the B+ tree.

In Figure 5.1 the B+ tree is a 4-ary tree. The root of the B+ tree is the Cluster key which is shared by all the L-sensors under this cluster. At level 0 the number of KEK is 4^0 =1 key (Cluster key, PK). Then at level 1, number of KEK is 4^1 =4 keys and at Level 2 the number is 4^2 =16.So total no of KEK is 1+4+16=21.Thus generalizing the concept, total number of KEK is

$$m^0 + m^1 + m^0 + m^2 = \sum_{i=0}^{h} m^i \quad (5.1)$$

Therefore total number of Key storage overhead for H-sensor is,

$$KEK_{total} = m^0 + m^1 + m^0 + m^2 = \sum_{i=0}^{h} m^i \quad (5.2)$$

From our previous discussion in our scheme we know that every L-sensor node will store all the keys along the path from leaf to root of the B+ tree. Therefore total number of Key storage overhead for each H-sensor is,

$$KEK_{total} = \text{height of the tree} = \log_m n. \quad (5.3)$$

At first, we compared our scheme with few basic pre-distribution schemes like random pairwise in [23], trivial pairwise in [24], closest pairwise in [25] schemes. From figure: 5.2, we can see that for a network with 600 nodes, KSO for random pairwise scheme is 198.For trivial solution its 566 and for closest pairwise scheme its 200 keys per node. Where in case of our scheme for minimum height h=3 and m=8 the H-sensor needs to store only 73 and the L sensor only 3 keys.

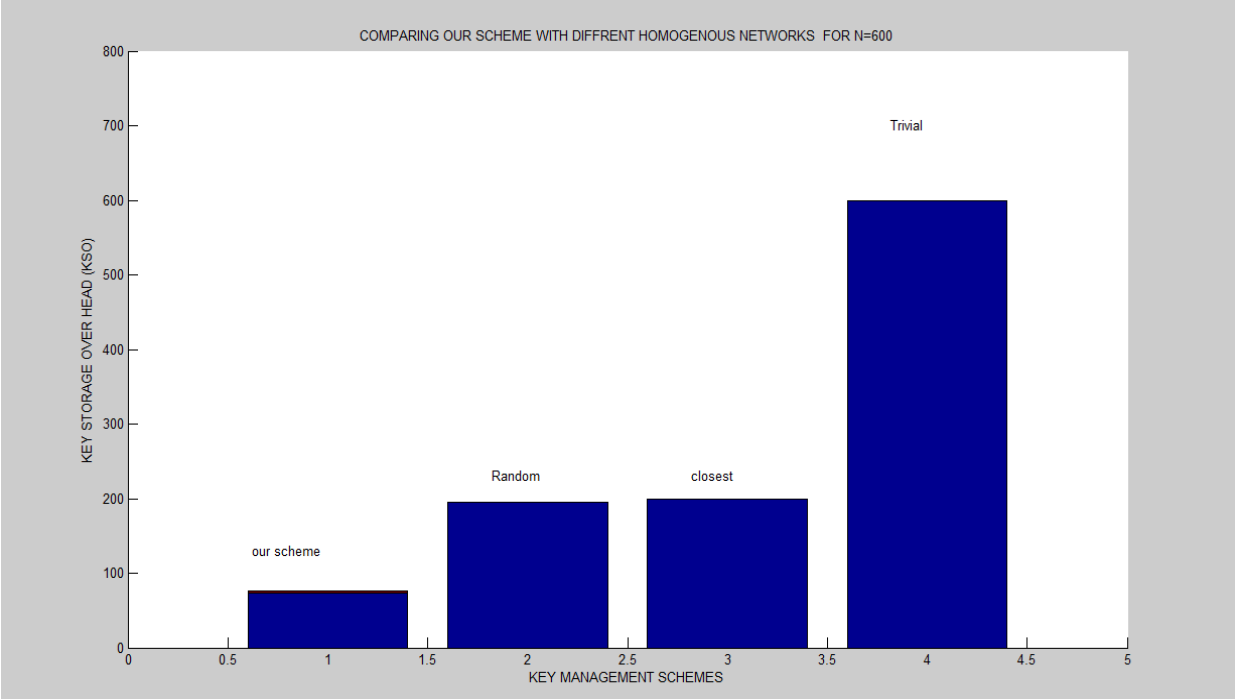

Figure5.1:Comparing our scheme with different homogeneous network for N=600

Now if we compare our scheme to other schemes which we have studied as our related work like Sajid et.al. Scheme in [16] for a network of 1000 L-sensor and 10 H-sensor, we observe that in Sajid et.al. , for a key sharing probability of 0.5 each L-sensor stores 2 generation keys and H-sensor stores approximately 100 generation keys. When the key sharing probability is 0.8 L-sensor stores 5 generation keys and H-sensor approximately 250 generation keys. For the Asymmetric pre-distribution the key Storage overhead is 500.

Similarly, we compared our scheme with another related work called LEAP in [19] for a 1000 nodes and found that for d=10 (d is the number of neighbors) where LEAP needs to store 203 KEK per node While in case of our scheme with a minimum height of 3 of the B+ tree and m=10, the cluster head needs to store $\sum_{i=0}^{3} m^{i}$=111 KEK each L sensor needs to store only $\log_8 1000$ =3 KEKs. We can see the differences from between the compared schemes from figure: 5.3.

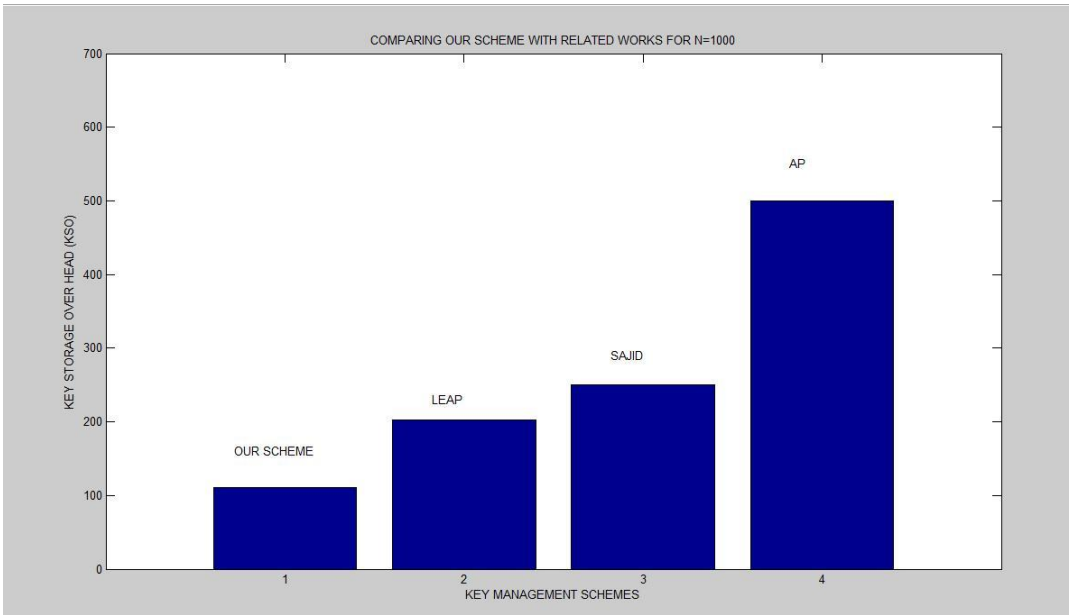

Figure: 5. 2: Comparing our scheme with related works for N=1000

Thus it is clear that as per key storage overhead our scheme has less storage.

5.2 Communication

Communication cost is measured in terms of number of messages needed to be exchanged in order to update the existing keys as a result of events like: addition of new node, node compromise and key refresh at regular intervals. For events like node addition and node compromise, numbers of key changes varies from 1 to $\log_m n$.

The changed cluster key and other intermediate keys are encrypted using the appropriate keys. The number of messages constructed and communicated varies from one to $\log_m n$. Therefore, communication at H-sensor vary from one to $\log_m n$ transmit operations. Similarly each L-sensor performs either one or $\log_m n$ receives operations.

For key refresh at regular intervals the new cluster key is encrypted using old cluster key and is sent to other sensors. Hence each H-sensor performs one transmit operations and L-sensor one receive operation in order to update the cluster key.

The communication cost incurred for the scheme as Sajid's scheme [16] is as follows: to collect neighboring node information each node communicates with every neighbor node. Each node performs p receive operations if p nodes are in the communication range. After collecting neighboring node information each L-sensor transmits it to H-sensor, in turn H-sensor selects a key and transmits it to L-sensor individually; for a group of n L-sensors, number of transmit and receive operations performed by H-sensor are 2n. L-sensors required to perform p+1 receive operations (p to collect neighbor node information and one to receive selected key from H-sensor) and two transmit operations (one initially sent to neighboring nodes and second transmit to send the collected information to H-sensor). Fig: 5.4 show us the difference between these two schemes in terms of communication cost.

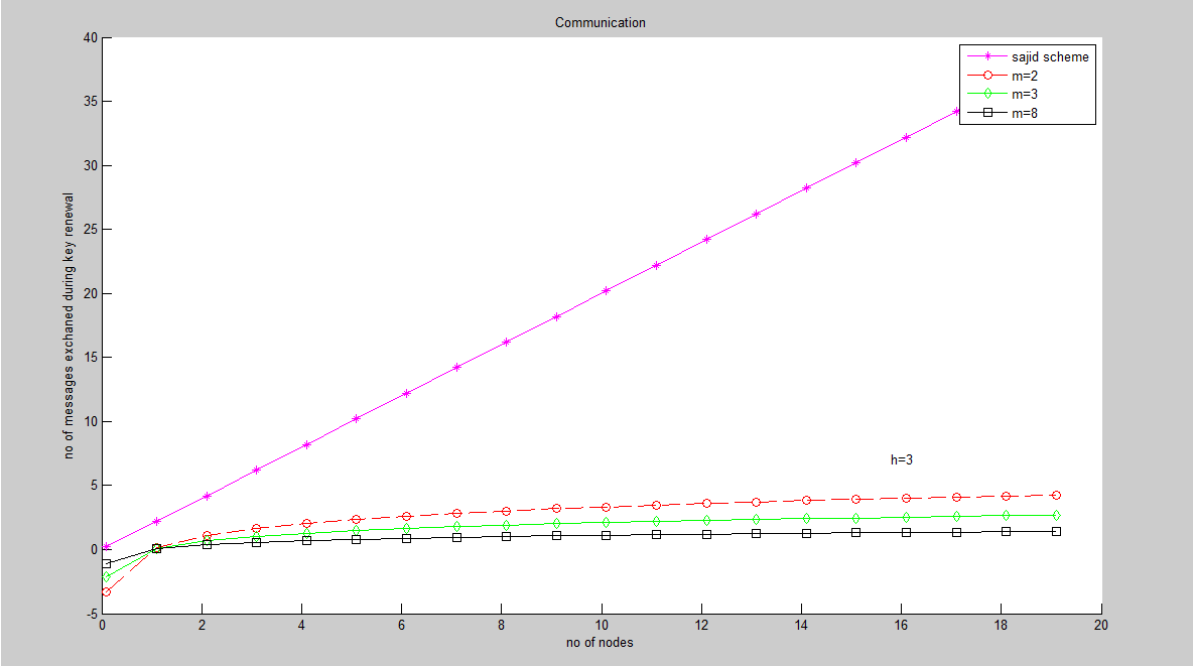

Figure : 5.4: Comparing communication overhead

5.3 Computation overhead

Computation costs are measured in terms of number of encryptions required to change the keys in the event of node compromise and node addition. Here we consider computation cost at both H-sensor and L-sensors for node addition and node compromise separately. When a node is added H-sensor has to update the tree by changing all the keys along the path from joining point till the root. After updating the tree new set of keys are encrypted using appropriate keys in order to distribute changed keys to the existing nodes as well as new node securely. Number of encryptions performed when a node is added is: to distribute new cluster key PK' to all the nodes except new node it is encrypted using old cluster key PK. New keys along the path from joining point till the root are encrypted using respective old intermediate keys and distributed to respective set of nodes. To distribute all the keys along the path to the new node the keys are encrypted using node's private key. Total number of encryptions performed by cluster head (H-node) in case of node addition is m*(h-1) where h is the tree height. For node addition computation with respect to L-sensor not in the path of the joining node is one i.e., decrypting the new cluster key which is encrypted by cluster head using old cluster key. For the L-sensor in the path of joining node computation is equal to (h-1) decryptions.

When a single node is compromised, keys along the path are changed and distributed securely to other nodes by the cluster head which maintains the tree. For the L-sensors along the path of the compromised node intermediate keys as well as cluster key will change where as for other L-sensors only cluster key is changed. Total number of encryptions required in case of node compromise is dependent on the position of the compromised node in the B+ tree. From analysis we have calculated that maximum number of encryption is required when only one node is compromised in each sub array. As the number of compromised node increases in each sub-array, number of encryption required to refresh the keys reduces. The maximum number of encryption required for key renewal when only one node is compromised in the network is:

$$(N-1)+(m-1)(h-2) \tag{5.4}$$

Where N is the total number of node sunder a cluster, m is degree of the tree and h is the height of the B+ tree.

As the value of m increases total number of encryption for key renewal reduces. We have compared the communication overhead of our scheme with sajid's scheme [16].

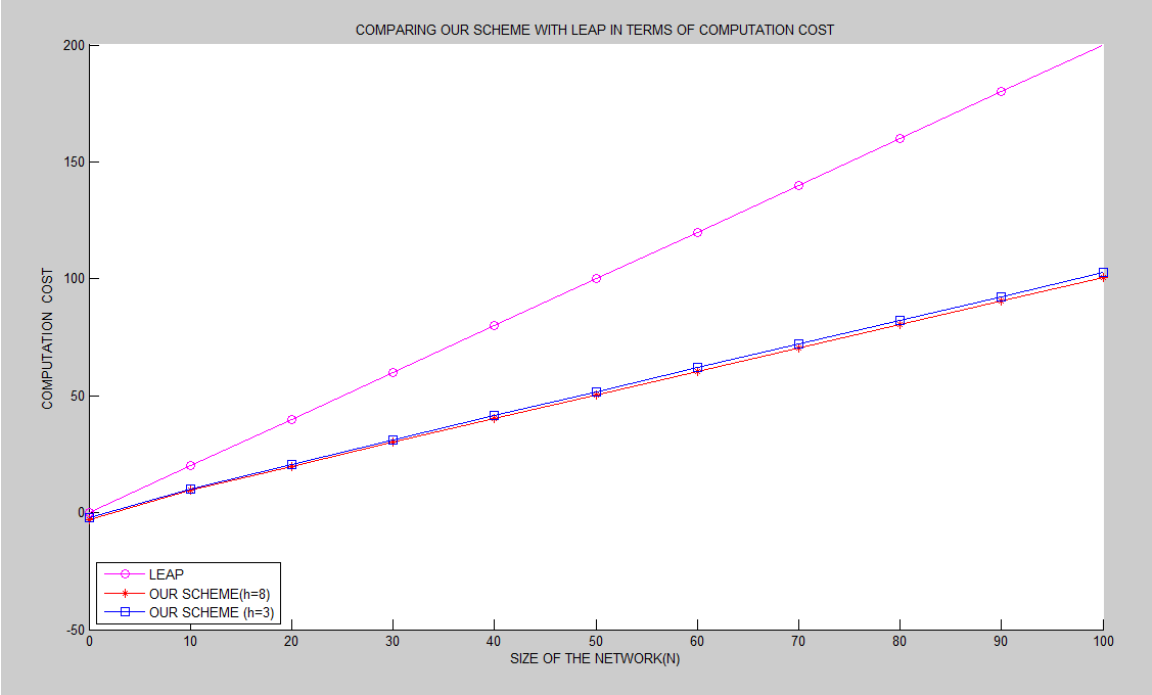

Figure: 5.5 Comparing computation cost with leap

When we compared the computation cost of our scheme which is maximum when one node is compromised with Leap [19] and we found that for Leap where the cost is 2*n for refreshing the group key and cluster key , in our scheme the maximum cost is much less. We can see the difference from the figure 5.5

5.4 Security Analysis

In our previous chapters we have talked about the major security attacks in wireless sensor networks. In this section we have shown that our scheme is an efficient scheme for key establishment that resists many types of attacks on the network, including the Denial of service, Sybil attack, sinkhole attack, wormhole, and so on. LEAP in [19] also provides efficient schemes for node revocation and key updating in WSNs

Denial of Service:

DoS attack tries to exhaust the resources available to the victim sensor node by sending extra unnecessary packets and prevents. The mechanisms to prevent DoS attacks include payment for network resources, pushback, strong authentication and identification of traffic. Our scheme can prevent DoS attack as provides Strong authentication at every stage of network communication through the key encryption keys. Before any kind of communication between two nodes a secured channel is established through authentication. Moreover each packet is send or received through key encryption and decryption for verification.

An L-sensor in our network communicates with only one H-sensor which it selects as its cluster head .It uses the pre-deployed common key for authentication while joining the cluster. And again when exchanging messages between them it uses the key encryption keys provided by the Cluster head. On the other hand, an L-sensor only communicates to those other L sensor nodes within its cluster to which it shares a common cluster key and can establish a pairwise key K_{uv} .

We can see that sensor nodes do not receive and process packets from nodes which are not trusted. Thus the exhaustion of network by processing of extra unnecessary packets cannot be done here.

Attack on Information in transit:

Wireless communication is vulnerable to eavesdropping. Any attacker can monitor the traffic flow and get into action to interrupt, intercept, modify or fabricate packets and provide wrong information to the base stations or sinks.

In our scheme, while communicating between L-sensors authentication is done by function f and PK and messages are encrypted using $K_{uv.}$ While Communication is between H-L-sensors, any message is encrypted $\log_m n$ times by KEK.

Sybil Attack:

In our scheme even if Id of a node is obtained by malicious node it cannot communicate with other nodes without obtaining PK or f as, thus Sybil attack cannot be performed. Instead the nodes will inform the H-sensor about the presence of malicious node and overall key renewal will take place in order to reinforce security.

Black hole or sinkhole attack:

In the sinkhole attack, a compromised node attracts packets by advertising information like high battery power, etc., and then later drops all the packets.

In our scheme, H-L sensor communication is strongly protected by $\sum_{i=0}^{h} m^i$ no. of KEK. Here H-sensor only communicates with those L-sensor that are in its cluster .Thus it will not trust any other nodes outside its network or with malicious behavior for best path or highest quality .Moreover if a node is compromised all the key encryption keys are refreshed and communication channels between L sensors are re-established using new cluster key PK . As a result it would not be possible for a compromised node to affect the network with sinkhole attack.

Hello Flood Attack:

Our scheme can also prevent a HELLO attack in which an adversary attacks the network by repeatedly transmitting HELLO messages and thereby depletes the network's resources. This attack is averted since the nodes in our scheme accept packets only from authenticated neighbors. In our scheme both H-sensor and L-sensors are pre-deployed with key K_i .Initial broadcast by H-sensors are encrypted this key K_i .When an L-sensor gets HELLO packet from any other sensor node it can verify whether the node is authenticated using this pre-deployed key. So malicious hello broadcast can be detected ted by L-sensors thus Hello Flood Attack cannot occur.

Wormhole attack:

In the wormhole attack an adversary launches two nodes in the network, one near the target of interest and the other near the base station. The adversary then convinces the nodes near the target, which would generally be multiple hops away from the base station, that they are only two hops away thereby creating a sinkhole. Also, nodes that are far away think that they are neighbors because of the wormhole created. In our scheme an adversary cannot launch a wormhole attack after key establishment as at that point every node has knowledge about its neighbors so it is not easy to convince a node that it is near a particular compromised node.

Comparison with other schemes:

Table: 5.1

Security analysis

Different Attacks	Sajid's scheme [16]	TLA [14]	Erik's scheme [15]	LEAP [19]	Our scheme
Dos	√	√	√	√	√
Eavesdropping	×	×	×	√	√
Hello flood	×	√	√	√	√
Worm-hole	√	√	√	√	√
Sink hole	√	√	√	√	√
Sybil	√	×	√	√	√
Clone	×	×	×	×	√

From the above table 5.1 we can see that our scheme can prevent most of the major attacks and better than the other schemes we have taken as our related work in terms of security.

5.5 Conclusion

In various applications, WSNs enable the monitoring of the target system or area. Especially in areas such as military, commercial and privacy applications, ensuring security is the most important issue. In this paper, we propose an enhanced heterogeneous B+ tree based key management scheme for wireless sensor networks that ensures security and survivability. In our scheme at first we design a B+ tree that stores keys to protect the node-to-cluster Heads communication and similar tree is generated by Base Station to secure the link of Cluster Head-to-Base Station . Then we utilize random key predistribution to initiate the security of WSN, which mainly support node-to-node security and a mutual trust authentication mechanism. In this scheme security is fortified by using hash functions and XOR operations to renew keys at regular interval or whenever a compromised node. In contrast to other similar security solutions like those of Sajid et. al, TLA and LEAP, the salient advantage of this work is that we addressed challenging security issues of runtime phase by real time rekey, which can efficiently protect the network against attacks of eavesdropping or captured nodes compromise and so on. Also our scheme performance excels in terms of communication, computation and storage when compared the above mentioned schemes. We hope our work is accepted and implemented for betterment of humanity. There is still lot of scope for future works on generation of keys and on re-key message update cycles. We propose to seamlessly integrate WSN security with a promising protocol that provides more security and energy efficiency.

LIST OF REFERENCES

[1] Lai B, Kim S, Verbauwhede I. Scalable session key construction protocol for wireless sensor networks. In: Proceedings of the IEEE workshop on Large Scale Real-time and Embedded Systems LARTES, December 2002.

[2] Perrig A, Szewczyk R, Wen V, Cullar D, Tygar JD. SPINS: security protocols for sensor networks. In: Proceedings of the 7th annual ACM/IEEE international conference on mobile computing and networking, July 2001. p. 189–99.

[3] Chan H, Perrig A. PIKE: peer intermediaries for key establishment in sensor networks. In: Proceedings of the 24th annual joint conference of the IEEE computer and communications societies (INFOCOM '05), Miami, FL, USA, March 2005. p. 524–35.

[4] L. Eschenauer, V.D. Gligor, A key management scheme for distributed sensor networks, in: Proceedings of the 9th ACM Conference on Computer and Communication Security.

[5] H. Chan, A. Perrig, D. Song, Random key predistribution schemes for sensor networks, in: Proceedings of the 2003 IEEE Symposium on Security and Privacy, May 11–14, pp. 197– 213.

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

[7] A Pairwise Key Predistribution Scheme for Wireless Sensor Networks by Wenliang Du,Jing Deng,Yunghsiang S. Han † and Pramod K. Varshney

[8] R. Blom. An optimal class of symmetric key generation systems. *Advances in* Cryptology: Proceedings of EUROCRYPT 84 (Thomas Beth, Norbert Cot, and Ingemar Ingemarsson, eds.), Lecture Notes in Computer Science, Springer-Verlag, 209:335–338, 1985.

[9] Lee J, Stinson DR. Deterministic key predistribution schemes for distributed sensor networks. In: Proceedings of ACM symposium on applied computing 2004, Lecture notes in computer science, vol. 3357, 2005, Waterloo, Canada, 2004. p. 294–307.

[10] Wallner D, Harder E, Agee R. Key management for multicast: issues and architectures, June 1999, RFC 2627.

[11] Di Pietro R, Mancini LV, Law YW, Etalle S, Havinga P. LKHW: a directed diffusion- based secure multicast scheme for wireless sensor networks. In: First international workshop on wireless security and privacy (WiSPr 03), 2003.

[12] Lazos L, Poovendran R. Secure broadcast in energy-aware wireless sensor networks. In: IEEE international symposium on advances in wireless communications (ISWC 02), 2002.

[13] Intanagonwiwat C, Govindan R, Estrin D. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In: Mobile computing and networking, 2000. p. 56–67.

[14] Boushra Maala, Hatem Bettahar, Hatem Bettahar. TLA: A Tow Level Architecture for Key Management in Wireless sensor Networks. The Second International Conference on Sensor Technologies and Applications

[15] Michael Conrad, Erik Oliver Blaß and Martina Zitterbart
A Tree Based Approach for Secure Key Distribution in Wireless Sensor Networks

[16] Sajid Hussain, Firdous Kausar and Ashraf Masood. An Efficient Key Distribution Scheme for Heterogeneous Sensor Networks

[17] X. Du, Y. Xiao, M. Guizani, and H.-H. Chen. An effective key management scheme for heterogeneous sensor networks.
Ad Hoc Networks, 5(1):24–34, 2007.

[18] L. Eschenauer and V. D. Gligor. A key management scheme for distributed sensor networks. In ACM CCS
2002, 2002.

[19] S. Zhu, S. Setia, S. Jajodia, LEAP: efficient security mechanisms for large-scale distributed sensor networks, in: Proceedings of The 10th ACM Conference on Computer and Communications Security (CCS '03), Washington D.C., October, 2003.

[20] I.Chang, R.Engel, D.Kandlur, D.Pendarakis and D.Daha."Key management for secure internet multicast using Boolean function minimization technique". ACM SIGCOMM'99, March 1999.

[21] N.F.P.180-1. Secure hash standard. Draft, NIST, May 1994.

[22] R.Rivest. The MD5 message-digest algorithm. RFC 1321, April 1992.

[23] S. Camtepe and B. Yener. Key distribution mechanisms for wireless sensor networks: a survey. Technical report, PRI,TR-05-07, 23 March 2005.

[24] D. Liu and P. Ning. Location-based pairwise key establishments for static sensor networks. In *SASN '03: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, pages 72–82, Fairfax, Virginia, USA, 2003.

[25] R. Pietro, L. Mancini, and A. Mei. Random key-assignment for secure wireless sensor networks. In *SASN '03*, pages 62–71, Fairfax, Virginia, USA, 2003.