Applications of Combinatorial Designs in Key Pre-distribution in Sensor Networks

Thesis submitted to Indian Statistical Institute

by

Dibyendu Chakrabarti
Applied Statistics Unit
Indian Statistical Institute
September, 2007

Applications of Combinatorial Designs in Key Pre-distribution in Sensor Networks

Thesis submitted to Indian Statistical Institute in partial fulfillment of the requirements for the award of the degree of Doctor of Philosophy

by

Dibyendu Chakrabarti
Applied Statistics Unit
Indian Statistical Institute
203 B. T. Road, Kolkata - 700 108, INDIA
e-mail : dibyendu_r@isical.ac.in

September, 2007

under the supervision of

Professor Bimal Roy
Applied Statistics Unit
Indian Statistical Institute
203 B. T. Road, Kolkata - 700 108, INDIA
e-mail : bimal@isical.ac.in

Dedicated to My Parents

Acknowledgments

A journey is easier when you travel together. Interdependence is certainly more valuable than independence. During the preparation of this thesis, I have been accompanied and supported by many people. It is my pleasure to have the opportunity to express my gratitude to all of them.

The first person I would like to thank is my supervisor Prof. Bimal Roy. I could not have staged a come back to academics after spending several years in industry if it were not for him. Besides being a renowned cryptographer, he is an eminent scholar in combinatorics and mathematics in general. I still keep wondering how he could predict the solvability of apparently very difficult problems with instinctive ease, though I had to struggle through nights to get to the solutions. Above all, he is a nice human being and he always stood by me during my hard times.

My career in research was initiated by Dr. P. PalChoudhury, with whom I had worked in a project before I joined formally as a research scholar. I attended a few courses and I learnt a lot from Prof. K. S. Vijayan and Prof. Rana Barua, apart from Prof. Bimal Roy and Dr. Subhamoy Maitra. I learnt some basic principles of life from Prof. Vijayan. He is a great teacher.

Dr. Subhamoy Maitra is another person with whom I have been interacting throughout my research career. He is a great mathematician, a very good programmer and possesses the virtue of "presence of mind." Trusting his gut feelings always proved to be extremely useful. He is more than a teacher to me.

Prof. Jennifer Seberry of University of Wollongong deserves a special mention. She accommodated me for a month in her department and provided me the opportunity to have a new vista of combinatorics and coding theory. She is not only a great mathematician, but also a very kind and helpful person. My stay in Australia will remain a memorable experience for the rest of my life.

My friends Dr. M. K. Mukherjee and Dr. M. K. Srivastava played a very vital role in inspiring me and going through the manuscript of this thesis. Dr. Srivastava boosted my morale during the troubled hours. My language is not strong enough to express my gratefulness to him.

I owe a lot to Prof. S. R. Chakraborty and Prof. A. Goswami for taking painstaking efforts to check the drafts of my work.

I pestered my colleague Dr. D. K. Dalai whenever I faced any problem with latex. He is so kind that he solved all the problems in no time without any qualms.

My parents were very supportive throughout the duration of my research. My friend Ms P. Choudhury inspired me a lot to get the job done in time. I am indebted to all of them.

Abstract

Key pre-distribution is an important area of research in Distributed Sensor Networks (DSN). Some improved techniques over the existing schemes (employing combinatorial designs) have been proposed in this thesis and detailed mathematical analysis of the schemes has been presented.

At first, combinatorial design followed by randomized merging strategy is applied to key pre-distribution in sensor nodes. Our main target is to get more than one pair of common keys between any pair of nodes to provide a robust network in terms of security under adversarial conditions where some nodes may get compromised. A transversal design is used to construct a (v, b, r, k) configuration and then randomly selected blocks are merged to form the sensor nodes. We have given detailed mathematical analysis of the number of nodes, number of keys per node and the probability that a link gets affected if certain number of nodes are compromised with supporting experimental data. The technique is tunable to user requirements and it also compares favourably with state of the art design strategies. An important feature of our design is the presence of a higher number of common keys between any two nodes. Further we study the situation where properly chosen blocks are merged to form sensor nodes such that there is no intra-node common key. We present a basic heuristic for this approach and show that it provides slight improvement in terms of certain parameters than our basic random merging strategy. Our idea presents a departure from the usual combinatorial design in the sense that the designs are readily obtained according to user requirements. Our merging strategy results into schemes that are not directly available from combinatorial designs.

Next we studied the largest cliques of a DSN based on transversal designs and the probabilistic extension of it (through merging). It is important to analyse the largest subset of nodes in a DSN where each node is connected to every other node in that subset (i.e., the largest clique). This parameter (largest clique size) is important in terms of resiliency and capability towards efficient distributed computing in a DSN. We concentrate on the schemes where the key pre-distribution strategies are based on transversal design and study the largest clique sizes. We show that merging of blocks to construct a node provides larger clique sizes than considering a block itself as a node in a transversal design. We consider the DSNs where the key pre-distribution mechanism evolves from combinatorial design. Such schemes provide the advantage of very low complexity key exchange facility (only inverse calculation in finite fields).

Next, we have proposed a general framework using combinatorial designs which will enable the participating devices to communicate securely among themselves with little memory and power overhead. The scheme caters for different kinds of user requirements and allows the designer to choose different combinatorial designs for different parts or levels of the network. This general framework will find application in all wireless radio technologies, typically WPANs (Wireless Personal Area Networks) and WLANs (Wireless Local Area Networks). This is a hitherto unexplored technique in wireless technologies. Our proposal is perfectly general and fits into networks of any size. Suppose there are several levels depending on the user requirements. The root of the hierarchy tree is assumed to be a central server, S. At the next level, x special nodes $S_1, S_2, \cdots, S_x$ are placed. The leaf level comprises of the sub-networks $NW_1, NW_2, \cdots, NW_x$. One has the freedom to choose different combinatorial designs for different parts of the network. Again, that depends on the specific requirements of the user. For example, if the sub networks are required to form a totally connected network graph, one can choose projective planes. If the sub-networks are very large in size and total connectivity is not a requirement (i.e., if single/multi-hop connectivity is permissible), transversal designs might be a reasonable choice.

Then we study the implementation of a distributed sensor network on a two dimensional grid, where the communication is considered to be secured under the assumption that the position of all the sensor nodes are fixed and the nodes are placed at the grid intersection points. The combinatorial structure used for this purpose is the well known transversal design. In this scenario, the number of keys in each node, the size of the area to be monitored, the sensing/RF (Radio Frequency) radius and the robustness of the network are inter-related and one can consider several design choices based on specific requirements. We present the key pre-distribution strategy and the connectivity analysis of the network.

Next we discuss a novel technique for increasing the number of common keys between the nodes of a sensor network deterministically with the help of combinatorial designs. A general protocol is described for the key agreement. Elegant methods are described to achieve the key agreement by calculating the common key(s) between two nodes when the underlying combinatorial structures are generated from Difference Sets. The extension of this scheme is also presented when the Kronecker Product Design is used.

Finally we pose some open problems and conclude the thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A wireless sensor network consists of a number of inexpensive sensor devices spread across a geographical area. Each sensor is capable of wireless communication using the RF (Radio Frequency). The sensor nodes also have some limited computing capability.

A few applications of sensor networks are as follows:

1. Military networks to detect and gain information about enemy movements, explosions and other phenomena of interest.

2. Networks to detect and characterize Chemical, Biological, Radiological, Nuclear and Explosive (CBRNE) attacks and material.

3. Networks to detect and monitor environmental changes in plains, forests, oceans, etc.

4. Wireless traffic networks to monitor vehicle traffic on highways or in congested parts of a city.

5. Wireless surveillance networks for providing security in shopping malls, parking garages and other facilities.

6. Wireless parking lot networks to determine which spots are occupied and which are free.

The above list suggests that wireless sensor networks offer certain capabilities and enhancements in operational efficiency in civilian applications as well as assist in the national effort to increase alertness to potential terrorist threats.

Two ways to classify wireless ad hoc sensor networks are whether or not the nodes are individually addressable and whether the data in the network is aggregated. The sensor nodes in a parking lot network should be individually addressable, so that one can determine the locations of all the free spaces. This application shows that it may be necessary to broadcast a message to all the nodes in the network. If one wants to determine the temperature in a corner of a room, then addressability may not be so important. Any node in the given region can respond. The ability of the sensor network to aggregate the data collected can greatly reduce the number of messages that need to be transmitted across the network. This function of data fusion is discussed more below.

The basic goals of a wireless ad hoc sensor network generally depend upon the application, but the following tasks are common to many networks:

1. **Determine the value of some parameter at a given location:** In an environmental network, one might want to know the temperature, atmospheric pressure, amount of sunlight and the relative humidity at a number of locations. This example shows that a given sensor node may be connected to different types of sensors, each with a different sampling rate and range of allowed values.

2. **Detect the occurrence of events of interest and estimate parameters of the detected event or events:** In the traffic sensor network, one would like to detect a vehicle moving through an intersection and estimate the speed and direction of the vehicle.

3. **Classify a detected object:** Is a vehicle in a traffic sensor network a car, a mini-van, a light truck, a bus, etc.

4. **Track an object:** In a military sensor network, track an enemy tank as it moves through the geographic area covered by the network.

In these four tasks, an important requirement of the sensor network is that the required data be disseminated to the proper end users. In some cases, there are fairly strict time requirements on this communication. For example, the detection of an intruder in a surveillance network should be immediately communicated to the police so that action can be taken.

Wireless ad hoc sensor network requirements include the following:

1. **Large number of (mostly stationary) sensors:** Aside from the deployment of sensors on the ocean surface or the use of mobile, unmanned, robotic sensors in military operations, most nodes in a smart sensor network are stationary. Networks of 10,000 or even 100,000 nodes are envisioned, so scalability is a major issue.

2. **Low energy use:** Since in many applications the sensor nodes will be placed in a remote area, service of a node may not be possible. In this case, the lifetime of a node may be determined by the battery life, thereby requiring the minimization of energy expenditure.

3. **Network self-organization:** Given the large number of nodes and their potential placement in hostile locations, it is essential that the network be able to self-organize; manual configuration is not feasible. Moreover, nodes may fail (either from lack of energy or from physical destruction) and new nodes may join the network. Therefore, the network must be able to periodically reconfigure itself so that it can continue to function. Individual nodes may become disconnected from the rest of the network, but a high degree of connectivity must be maintained.

4. **Collaborative signal processing:** Yet another factor that distinguishes these networks from MANETs is that the end goal is detection/estimation of some events of interest and not just communications. To improve the detection/estimation performance, it is often quite useful to fuse data from multiple sensors. This data fusion requires the transmission of data and control messages and so it may put constraints on the network architecture.

5. **Querying ability:** A user may want to query an individual node or a group of nodes for information collected in the region. Depending on the amount of data fusion performed, it may not be feasible to transmit a large amount of the data across the network. Instead, various local sink nodes will collect the data from a given area and create summary messages. A query may be directed to the sink node nearest to the desired location.

Sensor types and system architecture: With the coming availability of low cost, short range radios along with advances in wireless networking, it is expected that wireless sensor networks will become commonly deployed. In these networks, each node may be equipped with a variety of sensors, such as acoustic, seismic, infrared, still/motion video-camera, etc. These nodes may be organized in clusters such that a locally occurring event can be detected by most of, if not all, the nodes in a cluster. Each node may have sufficient processing power to make a decision and it will be able to broadcast this decision to the other nodes in the cluster. One node may act as the cluster master and it may also contain a longer range radio using a protocol such as IEEE 802.11 or Bluetooth.

1.1 Sensor Network and Secure Communication

The question of secure communication among the sensor nodes has become very important nowadays. This is a non-trivial task since the sensor devices are severely constrained by their computation and communication capabilities. As a result, public key algorithms are usually not applicable in sensor networks. To minimise the amount of computation and communication during the key agreement phase, one popular approach is to pre-distribute the keys among the sensor nodes so that any two nodes willing to communicate with each other may share one or more common keys. The subsequent communication may take place through some fast stream cipher. Combinatorial design techniques help a lot in the pre-distribution of keys. This thesis is an endeavour in that direction.

1.2 Thesis Plan

This thesis includes the work done in six papers [16, 17, 18, 19, 20, 21]. The contribution of the thesis is discussed in the following subsection. A summary of the chapters appearing in this thesis is presented in the next subsection.

1.2.1 Contribution

In this thesis, one of the objectives is to improve the key pre-distribution in a sensor network with respect to two metrics, viz., the key sharing probability and resilience against node capture, keeping all the other metrics unaltered. This is achieved by the use of combinatorial designs. This will be found in *chapters* 4 and 8. The other objective is the development of different design strategies of a sensor network. This is done in *chapters* 6 and 7. As another important parameter of a sensor network, a study on largest clique size is also included in the thesis (*chapter* 5).

In *chapter* 4, some hybrid techniques (probabilistic extension of combinatorial schemes) are introduced which yielded better results in comparison with that of [65]. Key sharing probability and resilience against node capture are shown to be better at the expense of storing more number of keys per node. In *chapter* 5, the largest clique size resulting from the network of [65] is calculated. This is a new idea altogether and as far as our knowledge goes, this aspect has not been considered so far in the literature. Next, we discussed methods of increasing the largest clique size of a network by our improved design strategy introduced in the previous chapter. In the next chapter (*chapter* 6), we introduced the idea of designing a hierarchical sensor network using different combinatorial designs tailored to the requirement of the user. The application of different combinatorial designs in network planning is also a new idea and not available in the literature. This application facilitates key sharing between different parts of the hierarchical network. In the next chapter (*chapter* 7), we study the connectivity and coverage properties of a sensor network when the deployment is known and a transversal design is used as a tool to construct the key pre-distribution scheme. Detailed theoretical study has been made on the connectivity and coverage properties and experimental results have been included. A design procedure of a sensor network satisfying certain parameters is also discussed. This study may be useful in any practical application of the sensor network. Though some studies on sensor networks with deployment knowledge was made earlier, it is important to note that the cumulative effect of applying a combinatorial design on a specific deployment configuration is one of its kind and is taken up over here. In *chapter* 8, we have established a number of results. We have given key agreement protocols when BIBDs are used as the underlying combinatorial design of the key pre-distribution scheme. Next, "Kronecker Product" designs are used to blow up the design and it has been observed that one obtains a (at most) three associate class PBIBD by taking the "Kronecker Product" of two BIBDs. The application of this result is significant since it allows us to create a very large network with desirable parameters. A clever protocol for key agreement in this case is also devised. This kind of application of combinatorial designs is new and gives deterministic way of designing large sensor networks with several common keys between any two nodes and reasonably high resilience.

1.2.2 Summary

A summary of the chapters appearing in this thesis is given here.

Chapter 1 contains a brief introduction to sensor networks. *Chapter* 2 gives a detailed survey on the existing works on sensor networks and key predistribution. *Chapter* 3 is on the introduction and mathematical preliminaries and provides the background of the rest of the thesis. *Chapters* 4, 5, 6, 7 *and* 8 contain the original contribution and finally *chapter* 9 outlines the open problems and the future works.

Chapter 4 is based on the papers [17, 18, 21]. In *chapter* 4, combinatorial design followed by randomized merging strategy is applied to key pre-distribution in sensor nodes. A transversal design is used to construct a (v, b, r, k) configuration and then randomly selected blocks are merged to form the sensor nodes. We present detailed mathematical analysis of the number of nodes, number of keys per node and the probability that a link gets affected if certain number of nodes are compromised. The technique is tunable to user requirements and it also compares favourably with state of the art design strategies. An important feature of our design is the presence of more number of common keys between any two nodes. Further we study the situation when properly chosen blocks are merged to form sensor nodes such that there is no intra-node common key. We present a basic heuristic for this approach and show that it provides slight improvement in terms of certain parameters than our basic random merging strategy.

Chapter 5 is based on the papers [16, 19]. It discusses the largest clique size of a sensor network based on transversal designs and develops constructions to increase the clique size. It is important to analyse the largest subset of nodes in a DSN (Distributed Sensor Network) where each node is connected to every other node in that subset (i.e., the largest clique). This parameter (largest clique size) is important in terms of resiliency and capability towards efficient distributed computing in a DSN (Distributed Sensor Network). In this chapter, we concentrate on the schemes where the key pre-distribution strategies are based on transversal design and study the largest clique sizes. We show that merging of blocks to construct a node provides larger clique sizes than considering a block itself as a node in a transversal design.

Chapter 6 is based on the paper [20]. In *chapter* 6, we have proposed a general framework using combinatorial designs which will enable the participating devices to communicate securely among themselves with little memory and power overhead. The scheme caters for different kinds of user requirements and allows the designer to choose different combinatorial designs for different parts or levels of the network. This general framework will find application in all wireless radio technologies, typically WPANs and WLANs. This is a hitherto unexplored technique in wireless technologies. For example, if the sub networks are required to form a totally connected network graph, one can choose projective planes. This may be applicable in case of a smart home. If the subnetworks are very large in size and total connectivity is not a requirement (i.e., if single/multi-hop connectivity is permissible), transversal designs might be a reasonable choice.

In *chapter* 7, we study the implementation of a distributed sensor network on a two dimensional grid, where the communication is considered to be secured. The combinatorial structure used for this purpose is the well known transversal design. In this scenario, the number of keys in each node, the size of the area to be monitored, the sensing/RF radius and the robustness of the network are inter-related and one can consider several design choices based on specific requirements. We present the key predistribution strategy and the connectivity analysis of the network. Further we study the robustness of the network when certain number of nodes are compromised by an adversary.

Chapter 8 discusses a novel technique for increasing the number of common keys between any two nodes of a sensor network deterministically with the help of combinatorial designs. It is also shown how to calculate the common key(s) between any two nodes when the underlying combinatorial structures are generated from Difference Sets. The key agreement protocols (based on this computation) are presented. The extension of this scheme is also presented when the Kronecker Product Design is used.

Chapter 9 poses some open problems and concludes the thesis.

Chapter 2

Sensor Networks and Key Pre-distribution: A Survey

2.1 Introduction

Sensor network is a part of a family of networks called "ad hoc networks." There is no rigid definition of the terminology "ad hoc networks." The two most popularly cited examples of ad hoc networks are mobile ad hoc networks (MANETs) and sensor networks. Still it is pointed out in [1] that sensor networks call for special attention because of the following reasons:

1. The number of nodes in a sensor network is much more compared to an ad hoc network.
2. The deployment is dense.
3. The nodes are more likely to fail frequently.
4. The network topology keeps on changing.
5. Usually the nodes use a broadcast approach whereas the ad hoc networks use point-to-point approach.
6. The nodes have constraints on power, memory and computational capacity.
7. Since the nodes are many in number, they often do not have global identification (ID).

2.1.1 Communication in Sensor Networks

In a sensor network, the links are usually formed by radio, infrared or optical media, though most of the current hardware is based on radio frequency. See [89] for an example of μAMPS wireless sensor node. Two other examples of similar kind are described in [80, 101].

Since infrared communication is license-free and resistant to interference from electrical devices, transceivers using infrared technology are cheaper.

Smart Dust motes [53] is an example of autonomous sensing, computing and communication system using the optical medium for transmission.

It may be noted that a "line of sight" between the sender and receiver is a requirement for both infrared and optical technologies.

2.1.2 Network Models

There are usually two different kinds of Wireless Sensor Networks(WSN) viz., Hierarchical Wireless Sensor Networks (HWSN) and Distributed Wireless Sensor Network (DWSN).

In a HWSN, the nodes are of three types: base stations, cluster heads and sensor nodes. In fact, this is the hierarchical ordering among the nodes. Base stations are usually more powerful, have more computational and memory capacity and serve as gateways to other networks. Sometimes they are assumed to be tamper resistant, trusted and act as key distribution centres. Sensor nodes are deployed around single or multiple hop neighbourhood of the base stations. The cluster heads are special nodes, whose job is to collect and merge the data obtained from the sensor nodes and forward it to base stations. The base station has enough transmission power to reach all the nodes, but the converse is not true in general. The data flow in a HWSN may be

1. network-wise (broadcast) from base stations to sensor nodes.
2. pair-wise (unicast) among sensor nodes.
3. group-wise (multicast) within a cluster of sensor nodes.

In a DWSN, there is no fixed infrastructure and network topology is not known prior to deployment. Sensor nodes are usually randomly scattered all over the target area. Once they are deployed, each sensor node scans its radio coverage area and finds out its neighbours. Data flow in DWSN is similar to that of HWSN, with the difference that every node can broadcast.

2.2 Protocols and Restrictions

The protocol stack used by the sink and sensor nodes consists of the following: physical layer, data link layer, network layer, transport layer and application layer. In this section, we shall talk about some of the existing work in different layers of the protocol stack which may serve as a summary of the current knowledge base of present day technologies.

We begin with some examples of data link layer applications/protocols. In contrast to Bluetooth and MANETs, the transmission power and radio range of a sensor node is much less. Node mobility and failure give rise to frequent changes in network topologies. The bottom line is that the sensor networks must conserve power in order to have a prolonged operational duration. That is why the existing MAC(Medium Access Control) protocols can not be used without modifications.

Two versions of MACs have been proposed so far, viz., fixed allocation and random access [92, 101].

In fact, SMACS protocol and EAR algorithm are discussed in [92].

In [101], a MAC scheme based on carrier sense multiple access (CSMA) is introduced and in [89], a hybrid TDMA/FDMA-based centrally controlled MAC scheme is presented.

In [90], a dynamic power management scheme is proposed for wireless sensor networks with five power-saving nodes. This is an example of a work on power saving modes of operation.

In [89], an error control of transmission data is done using FEC (Forward Error Correction). They assumed frequency non selective, slow Rayleigh fading channel and convolutional codes for FEC.

Next we mention some network layer protocols/applications. Routing in a sensor network may be based on several approaches. Among them there are energy efficient routing like maximum power available (PA) route, minimum energy (ME) route, minimum hop (MH) route, maximum minimum PA node route etc. There may be data-centric routing also. Interest dissemination is performed to assign the sensing tasks to the sensor nodes. Interest dissemination may be done in two ways. In [52], the sinks broadcast the interest and in [45], the sensor nodes broadcast an advertisement for the available data and wait for a request from the interested nodes. This kind of routing calls for attribute-based naming as given in [87]. [45] also discusses data aggregation calling it "data fusion." In [44], data aggregation is defined as a set of automated methods of combining the data that comes from many sensor nodes into a set of meaningful information. A protocol is developed in [82] to compute an energy-efficient sub-network called MECN(minimum energy communication network). A new algorithm called SMECN (small MECN) is proposed in [66]. SMECN creates a subgraph of the sensor network that contains the minimum energy path. Both the protocols follow the minimum-energy property.

Flooding is an example of an old technique used for routing in sensor networks. It broadcasts data to all neighbour nodes regardless of whether they have received it previously or not. It has several limitations such as [45]. Among them, implosion, overlap and resource blindness are a few important ones.

A variant of flooding is called gossiping as discussed in [43]. A sensor node randomly selects one of its neighbours to send the data and this process is repeated until the data reaches the destination. Obviously it may incur a long propagation time.

In [45], the deficiency of classic flooding is overcome by negotiation and resource adaptation. In fact, they propose a family of adaptive protocols called Sensor Protocols for Information via Negotiation (SPIN). It sends data to sensor nodes only if they are interested.

In [92], a set of algorithms that perform organisation, management and mobility management operations in sensor networks are proposed. It is called sequential assignment routing. It creates multiple trees where the root of each tree is one hop neighbour from the sink. It selects a tree for data to be routed back to the sink according to the energy resources and additive QoS metric.

Low-Energy Adaptive Clustering Hierarchy (LEACH) is a clustering-based protocol that minimises energy dissipation in sensor networks [44].

The direct diffusion data dissemination paradigm is proposed in [52]. It sets up gradients for data to flow from source to sink during interest dissemination.

In [87], three possible application layer protocols have been discussed, viz., Sensor Management Protocol (SMP), Task Advertisement and Data Advertisement Protocol (TADAP) and Sensor Query and Data Dissemination Protocol (SQDDP). The descriptions of some of the tasks that are expected to be performed by SMP are described in [87]. Also Sensor query and tasking language (SQTL) is proposed in [87].

2.3 Security Consideration in Sensor Networks

Let us point out the fundamental difficulties in providing security to a sensor network.

1. One can not take the advantage of asymmetric cryptography since the sensor devices have constraints in terms of computation, communication, memory and energy resources. RSA algorithm or Diffie-Hellman key agreement protocol are difficult to implement whereas the symmetric solutions like AES block cipher and HMAC-SHA-1 message authentication code are faster and easier to compute for the sensor nodes.

2. The nodes may be physically captured. Usually one should not assume that the hardware in each node is tamper-resistant. Compromised nodes may behave arbitrarily, possibly in collusion with other compromised nodes.

3. Since communication is wireless in sensor networks, eavesdropping and injection of malicious messages is easy.

4. The sensor network security protocols should be amenable to scalability. Usually the network is often required to be scaled up to cater to several sensor nodes.

5. Lack of fixed infrastructure.

6. Unknown network topology prior to deployment.

2.4 Attack Models

2.4.1 Outsider Attacks

If the attacker is not an authorised participant of the network, it is called an outsider attack. For example, a passive eavesdropper, packet spoofer or signal jammer may launch an outsider attack. Also physical destruction of nodes (may be intentional, climatic or resulting from depletion of energy sources) is a form of outsider attack. Benign node failure is to be considered as a security problem since it is indistinguishable from an attack resulting into disabling of a node.

2.4.2 Insider Attacks

Essentially an insider attack means the compromise of a sensor node. A compromised node may run some malicious node to steal some secret from the network and/or disrupt the normal functioning of the sensor network. To communicate with the sensor network, it has a compatible radio. Moreover, it is an authorised participant of the network. If standard encryption and authentication protocols are implemented in the network, the compromised node has to have some valid secret keys (which enable it to join the secret and authenticated communications). In the worst possible scenario, if the compromised node behaves in a totally arbitrary manner, it is said to follow the Byzantine model [62].

2.4.3 Central Trusted Authority in the Form of a Base Station

If the base station is assumed to be a trusted server that is never compromised, the problem of key distribution finds a ready solution. The base station serves as the trusted intermediary and distributes a key to each pair of nodes that need to communicate. However, for a network of very large size, the nodes in the immediate vicinity of the base station will have to continuously relay the key set up messages and very soon deplete the energy source. Also the base station will have to set up $\frac{n(n-1)}{2}$ keys in the worst case and becomes inefficient in case of large n.

2.5 Desiderata of a Secure Sensor Network Protocol

2.5.1 General Consideration

The basic idea is to make the network resistant to outsider attacks and resilient to insider attacks (while maintaining a realistic notion of security).

The former may be achieved by standard cryptographic primitives and maintain some redundancy in the network. The network protocols should be capable of identifying the failed nodes in real time and update themselves according to the updated topology.

For the later, the ideal situation would have been to detect the compromised node and revoke the keys contained therein. It is not always possible and perhaps the way out is to design protocols resilient to node capture so that the performance of the network gracefully degrades with the compromise of a small fraction of nodes.

Depending on the application and sensitivity of the collected data, the security level may be relaxed or beefed up.

2.5.2 Specific Requirements

1. Authentication: It is usually in two forms namely source authentication and data authentication. The verification of the origin of a message/packet is known as source authentication and the condition that the data is unchanged during the transmission is known as data authentication. Though authentication prevents outsider attacks like injecting/spoofing of packets, but a compromised node can authenticate itself to the network since it is in possession of valid secret keys.

2. Secrecy: Using standard cryptographic techniques like AES and shared secret keys between the communicating nodes is not sufficient to maintain secrecy because an

eavesdropper may analyse the network traffic and obtain some sensitive meta data. Access control has to be exercised in order to protect the privacy of the collected data. An insider attack may defeat this purpose since the data may be revealed or the communication between two nodes may be eavesdropped by a compromised node.

3. Availability: Availability means the functioning of the devices for the entire lifetime. Denial of Service (DoS) attacks result in a loss of availability. Both outsider and insider attacks may cause non-availability.

4. Integrity of Service: In the application layer, the protocols may be required to provide service integrity in the face of malfunctioning (compromised) nodes. As an example, the data aggregation service should be able to filter out the erroneous readings provided by the compromised nodes. The other example may be the time synchronisation protocol. The implementation of this protocol in a trusted environment is provided in [33].

2.6 Solutions

1. Secrecy and authentication may be protected from outsider attacks (like packet spoofing/modification and eavesdropping) using standard cryptographic techniques. Key establishment and management and broadcast/multicast authentication are two such solutions.

   (a) Two sensor nodes can set up a secret and authenticated link though a shared secret key. The problem of setting up the secret key between a pair of nodes is known as the key establishment problem. There are various solutions available to this problem. Among them, the most naive one is to use a single master key for the entire network. The moment a single node is compromised, the entire network goes haywire. At the other extreme, if one uses different keys for each pair of nodes, it will be extremely secure. This scheme is not viable because each node has to store several keys, which is not permissible in sensor nodes. This solution does not scale well with the increase in the size of the network. The other solution may be obtained using public key cryptography, though being computationally intensive, it is not suitable for sensor networks even with today's technology. The public key solution is also susceptible to DoS attacks. The probabilistic key pre-distribution has been discussed in [22, 31, 34, 67]. Other techniques of key pre-distribution will be discussed in detail in the later part of this chapter.

   (b) Many sensor network protocols use broadcast and multicast, one can not use digital signatures for the verification of the messages since public key cryptography is difficult in sensor networks. As a possible solution, in [76], the μTesla protocol has been proposed. A notion of asymmetry is introduced into symmetric key cryptography by the use of one-way function key chains and delayed key disclosures.

2. Availability may be disrupted through DoS attacks [102] and may take place in different parts of the protocol stack.

   (a) At the physical layer, jamming may be tried by propagating interfering RF signals. The other form of jamming may be by injection of irrelevant data or wastage of battery power at the reception node. The solution to this problem is discussed in [78], where frequency hopping and spread spectrum communication have been suggested. The jamming may also take place in the link layer by inducing malicious collisions or obtaining an unfair share of the radio resource. It is nothing but a weakness of the MAC protocols and the solution is to design secure MAC protocols as described in [102]. If the jamming is attempted at the networking layer through the injection of malicious data packets, one can use authentication to detect such packets and nonces to detect replayed packets.

   (b) There is another kind of attack called the Sybil attack [29, 74]. In this case, a malicious node claims multiple identities. The affected node can claim a major part of the radio resource. The attacker will succeed in achieving selective forwarding and in creating a sinkhole so that the affected node can capture a large amount of data [55]. The defence mechanisms have been detailed in [74] leveraging the key distribution.

   (c) There may be different other kind of attacks such as denying a message to the intended recipient, dropping of packets and selective forwarding [55]. Multipath routing solves this problem [25, 36]. In [55], some other attacks such as spreading bogus routing information, creating sinkholes or wormholes and Hello flooding have been described.

3. Service integrity may be at stake if the attacker launches a stealthy attack in order to make the network accept a false data value. It may be achieved in different ways such as compromising an aggregator node, a Sybil attack by a compromised node to affect the data value, a DoS attack to legitimate nodes to stop them reporting to the base station etc. In [81], the stealthy attack in data aggregation context and SIA (Secure Information Aggregation) Protocol have been proposed.

2.7 Key Distribution Mechanisms: An Overview of Key Pre-distribution Schemes

In this section, we shall have a look at the existing works on key pre-distribution schemes.

2.7.1 Metrics to Evaluate a Key Distribution Mechanism

(a) **Scalability:** It is the ability to support larger networks (even after deployment).

(b) **Efficiency:**

  i. **Memory:** Amount of memory required to store security credentials.

  ii. **Computation:** Number of processor cycles required to establish a key.

  iii. **Communication:** Number of messages exchanged during a key generation process.

(c) **Key connectivity (probability of key-share):** Probability that two (or more) sensor nodes store the same key or keying material.

(d) **Resilience:** Resistance against node capture.

In general, one can not increase all the above metrics at the same time. Trade-off among these metrics is inevitable and it depends on the specific application scenario.

2.8 Key Distribution in DWSN

Key distribution problem in DWSN is solved in three ways viz.,

(a) **Probabilistic:** Key chains are randomly selected from a key pool and distributed to sensor nodes.

(b) **Deterministic:** Deterministic algorithms are used to design the key-pools and key-chains to provide better connectivity.

(c) **Hybrid:** These are probabilistic extension of deterministic solutions to improve scalability and resilience.

2.8.1 Pair-wise Key Distribution Schemes

Schemes under this category have three phases in general:

(a) key setup prior to deployment

(b) shared-key discovery after deployment

(c) path-key establishment (if two communicating nodes do not have a key in common).

Pair-wise Key Distribution Solutions

Two obvious solutions should be discussed at the outset. One is to put a single master key in all the nodes. This solution is not at all resilient since the compromise of a single node will compromise the security of the entire network. The other one is to store $n-1$ keys in each sensor node for a network of size n, i.e., a different key for each pair of nodes. The very good resilience notwithstanding, this solution does not scale well with the increase in the size of the network.

In [22], a random pair-wise key scheme is introduced. This scheme does not require much storage but gives good resilience. It is based on Erdos and Renyi's work. Every pair of node is connected with probability p and each node stores a random set of Np pair-wise keys. At key setup phase, each node identity is matched with Np other randomly selected node IDs. A pair-wise key is generated for each ID-pairs and is stored in both nodes' key- chain along with the ID of the other node. Thus each node uses $2Np$ amount of memory locations to store its key-chain. At shared-key discovery phase, each node broadcasts its ID, i.e., sends one message and receives one message from each of its neighbours within radio coverage. Neighbour nodes can find out if they share a common . This scheme has very good key resilience, but low key sharing probability. In other words, there is a clear trade-off between storage and key connectivity.

In [68], a closest (location-based) pair-wise key pre-distribution scheme is discussed. It utilises the deployment knowledge to improve the key sharing probability. It is assumed that the sensors are deployed in a two-dimensional plane and the location of the sensor nodes are predictable. Each node is made to share keys with c nearest neighbours. In key setup phase, for each node S_A, a unique key K_A and c nearest neighbours $S_{B_1}, \cdots, S_{B_c}$ are selected. For each pair (S_A, S_{B_i}), a pair-wise key K_{AB_i} $= f(K_{B_i}|ID_A)$ is generated where f is a pseudo random function. Node S_A stores all pair-wise keys, whereas node S_B only stores the key K_{B_i} and the pseudo random function. To store the key chain, only $2c+1$ memory locations are used. So new nodes may be added to the network very easily since the new node may be pre-loaded with the pair-wise keys for c sensor nodes in its expected location. This scheme does not use much memory and gives good key-sharing probability between two nodes provided deployment errors are low. A sensor either uses its CPU to search for a key or generates it with a pseudo random function. Similar to [22], this solution is also resilient and scalable.

In another work [64], ID based one way function scheme (IOS) is introduced. It assumes a connected $r-$regular graph G with edge decomposition into star-like sub-graphs. Pair-wise keys are distributed according to these sub-graphs. A sensor node S_A receives a secret key K_A and secret keys $Hash(K_{B_i}|ID_A)$ if S_A is in the star-like graph centred around node S_B. Node S_B can always generate the secret key $Hash(K_{B_i}|ID_A)$ by using its secret K_B and public $ID(A)$. In an $r-$regular graph G, each sensor node can be the centre of one and leaf of $\frac{r}{2}$ star-like sub-graphs. Thus each sensor uses $r+1$ memory locations to store keys and key IDs. It has very good resilience and two nodes share a key almost certainly in at most two hops.

In [64] itself, the authors proposed multiple IOS to improve scalability of IOS. Every node in graph G corresponds to l nodes $S_A = S_{A_1}, \cdots, S_{A_l}$. Thus sensor nodes S_{A_i} store a common key K_A and a secret hash $Hash(K_B|ID_{A_i})$. Every node S_{B_j} in the class of node S_B, can use common key K_B to generate the secret hash $Hash(K_B|ID_{A_i})$ for node S_{A_i}. Multiple IOS decreases memory usage by a factor of l. It has less resilience since compromise of a class key is equivalent to compromise of the links of l sensor nodes.

Master Key Based Key Pre-distribution Solutions

Broadcast Key Negotiation Protocol (BROSK) [60] is based on a single master key pre-loaded in all the nodes. A pair of sensor nodes (S_i, S_j) exchanges random nonce values and use the master key K_m to establish session key $K_{ij} = f(K_m|RN_i|RN_j)$. Only one memory location is used to store the master key. The compromise of the master key results into the compromise of all the link keys, since they are known once the master key is known. Hence the resilience of this scheme is very low.

The Lightweight Key Management System [32] has slightly better resilience. It uses more than one master key. It assumes a deployment of sensor nodes in generations of size θ. Each node stores a group authentication key bk_1 and a key generation key bk_2. Two nodes of the same generation S_A and S_B authenticate each other by the authentication key bk_1. They exchange random nonce values RN_A and RN_B to generate the session key $K_{A,B} = f(bk_2|RN_A|RN_B)$. If the nodes S_A and S_B belong to two different generations i and j respectively, the authentication is done as follows: Let i be the older generation. S_A stores a random nonce RN_A and a secret S_{Aj} for each new generation j. Secret S_{Aj} is used to authenticate sensor nodes from new generation j. Node S_B of new generation j can authenticate itself by generating the secret $SA_j = f(gk_j|RN_A)$ given RN_A where f is a pseudo random function. Secret gk_j is only known to nodes of new generation j. Once authenticated, both parties use S_{A_j} as the key generation key to generate the pair-wise key K_{AB}. If there are g such generations, each sensor needs at most $2g+4$ memory locations to store the keys. However, if the secrets bk_1, bk_2 and gk_j of generation g are revealed, all the links of nodes in generation j are revealed. Hence the resilience of the scheme is low. Also the attacker may log the messages for future processing when the network is totally compromised.

Random Key-chain Based Key Pre-distribution Solutions

The basic scheme is given in [34] and is known as Basic probabilistic key pre-distribution scheme. It relies on probabilistic key sharing among the nodes of a random graph. In key setup phase, a large key-pool of KP keys and their identities are generated. For each sensor, k keys are randomly drawn from the key-pool without replacement. These k keys and their identities form the key-chain for a sensor node. Thus key-sharing probability between two sensor nodes becomes $p = \frac{((KP-k)!)^2}{((KP-2k)!KP!)}$. In shared-key discovery phase, two neighbour nodes exchange and compare list of identities of keys in their key-chains. Each sensor node broadcasts one message and receives one message from each node within its radio range where messages carry key ID list of size k.

Cluster key grouping scheme [51] proposes to divide key-chains into c clusters where each cluster has a start key ID. Remaining key IDs within the cluster are implicitly known from the start key ID. Only start key IDs for clusters are broadcast during shared-key discovery phase and messages carry key ID list of size c instead of k. There is another scheme called Pair-wise key establishment protocol [103]. Every node is assigned a unique ID which is used as a seed to a pseudo random function. Key IDs for the keys in the key-chain of node S_A are generated by $f(ID_A)$ where f is a pseudo random function. Thus, broadcast messages carry only one key ID. Storage requirement decreases drastically but a sensor node has to compute $f(ID_A)$ for each broadcast message received from a neighbour.

Transmission Range Adjustment Scheme [51] proposes sensor nodes to increase their transmission ranges during shared-key discovery phase. Once the keys are discovered, the nodes return to their original optimal transmission range. The objective is to decrease communication burden in path-key establishment phase and to save energy while still providing a good key connectivity. The key identities may be protected during shared-key discovery using Merkle Puzzle [73]. Solving of this puzzle increases processing and communication load. The node pairs unable to discover a common key apply path-key establishment phase to communicate securely through other nodes. Use of larger key pools lead to improved scalability and resilience, but it may also deteriorate the key sharing probability since key chain size may not increase due to storage limitations. Probability that a link is compromised, when a single node is compromised is $\frac{k}{KP}$ which is very high for small key-pools and indicates low resilience.

There are many key reinforcement proposals as well. Their purpose is to enhance the security of the established keys and improve resilience. If a unique link or path key is generated using established keys, the key is not compromised when one/more nodes are captured. It may be done by increasing the number of overlapped keys in the shared-key discovery phase. This is done in Q-composite random key pre-distribution scheme [22]. It requires q common keys to establish a link key. The link key K_{AB} between the nodes S_A and S_B is calculated as follows: $K_{AB} = Hash(K_1||K_2||\cdots||K_q)$. The probability that a link is compromised because of the capture of a node is $\frac{\binom{k}{q}}{\binom{KP}{q}}$. This is less than $\frac{k}{KP}$ and hence resilience improves. At the same time, the key sharing probability decreases since two nodes have to share q keys instead of one. The other way to do it is to reinforce the established link key. In [22], Multi-path key reinforcement scheme is also discussed. Node S_A generates j random key updates rk_i and sends them through j disjoint secure paths. S_B can generate the reinforced link key $K^r_{AB} = K_{AB} \oplus rk_1 \oplus \cdots \oplus rk_j$ upon receiving all key updates. This approach requires S_A and S_B to send and receive j messages, each carrying a key update. Also each node on the j disjoint paths has to send and receive an extra message. Similar mechanism is proposed by Pair-wise Key Establishment Protocol [103]. It uses threshold secret sharing for key reinforcement. S_A generates a secret key K^r_{AB} and $j-1$ random shares sk_i and $sk_j = K^r_{AB} \oplus sk_1 \oplus \cdots \oplus sk_{j-1}$. S_A sends the shares through j disjoint secure paths and S_B can recover K^r_{AB} upon receiving all the shares. In Co-operative pair-wise key establishment protocol [79], S_A first chooses a set $C = c_1, c_2, \cdots, c_m$ of co-operative nodes. A co-operative node provides a hash $HMAC(K_{c_1,B}, ID_A)$ and the reinforced key is defined as $K^r_{AB} = K_{AB} \oplus (\oplus_{c \in C} HMAC(K_{c,B}, ID_A))$ where K_{AB} and $K_{c,B}$ are the established link keys. Node S_A shares set C with node S_B and thus S_B can generate the same key. In this case, S_A and S_B need to send and receive c more messages and co-operative nodes have to send and receive two extra messages. Apart from the communication overhead, each co-operative node has to calculate $HMAC$ function twice for S_A and S_B. In this approach, both computation and communication complexity increase but the compromise of the key chain does not directly affect the security of any links in the network. Thus resilience is improved. However, the adversary can recover the initial link keys and subsequently the reinforced link keys from the multi-path reinforcement messages.

Practically nodes located far apart do not need to have any key in common. Similar to

[68], location information is also used in [30]. The sensor nodes are divided into $t \times n$ groups G_{ij} and deploys them around (x_i, y_j) for $1 \leq i \leq t$ and $1 \leq j \leq n$ where the points are arranged in a two dimensional grid. The location of the node follows the two dimensional Gaussian distribution with pdf $f_m^{i,j}(x, y|m \in G_{i,j}) = f(x - x_i, y - y_j)$. In key setup phase, key pool KP is divided into $t \times n$ key pools KP_{ij} of size ω_{ij} and this pool is used for the group G_{ij}. Given ω_{ij} and overlapping factors α and β, the key-pool is divided into subsets where

(a) two horizontally and vertically neighbouring key-pools have $\alpha \times \omega_{ij}$ keys in common.

(b) two diagonally neighbouring key-pools have $\beta \times \omega_{ij}$ keys in common.

(c) non-neighbouring key-pools do not share a key.

Basic probabilistic key pre-distribution scheme is applied within each group but how to decide α, β *and* ω_{ij} is still an open question.

Combinatorial Design Based Solutions

In [12], key pre-distribution techniques based on combinatorial design theory has been introduced. Symmetric and generalised quadrangle design techniques are used and the scheme uses finite projective plane of order n, where n is a prime power. It is a symmetric BIBD with parameters $(n^2 + n + 1, n^2 + n + 1, n + 1, n + 1, 1)$. A network of size $n^2 + n + 1$ may be accommodated using this design. The number of keys is also $n^2 + n + 1$. Every pair of nodes have a single key in common. So key sharing probability is 1. Each key occurs in $n + 1$ nodes and each node contains $n + 1$ keys. Probability of compromise of a link following the capture of a node is $\frac{1}{n+1}$. n being a prime power, all the network sizes can not be supported unless sufficient redundancy is allowed. Generalised quadrangles (GQ) provide more scalable solutions. In GQ, even if two nodes do not have a key in common, there exists a node having a key common with both. Thus the nodes are connected by at most a two-hop path. [12] has examples of different GQs to support networks of order $O(n^3), O(n^5)$ *and* $O(n^4)$ with key-sharing probabilities $\approx \frac{1}{n}, \frac{1}{n^2}$ *and* $\frac{1}{n^{1.5}}$ respectively. Even GQs need n to be a prime power. [12] also proposes a probabilistic extension to the core combinatorial design. It improves scalability and resilience at the cost of decreased key sharing probability. Very similar approach based on transversal designs is proposed in [65]. The key sharing probability is $\frac{k}{r+1}$ for a transversal design $TD(k, r)$ and the probability of compromise of a link consequent upon the failure of s nodes is $fail(s) = 1 - \left(1 - \frac{r-2}{r^2-2}\right)^s$.

Matrix Based Dynamic Solutions

Consider a network of size N. All possible keys may be represented by a $N \times N$ matrix. If it were possible for each pair of nodes to calculate the corresponding entry of the matrix, they may use it as the secret key for communication. In [4], a public $(\lambda+1) \times N$ matrix G and a private $N \times (\lambda + 1)$ matrix D over $\mathbb{F}_{\shortparallel}$. If no more than λ nodes are compromised, this scheme is secure and is known as λ secure. For that, G must have $(\lambda + 1)$ linearly independent columns. The key matrix K is defined as $K = (D.G)^T.G$. Each sensor node S_i stores the i-th column of G of size $(\lambda + 1)$ as public information and i-th row of $(D.G)^T$ as private information. A pair of communicating nodes (S_i, S_j) first exchange their publicly available i-th and j-th columns. The key is then calculated as $K_{ij} = i$-th row $\times j$-th column and $K_{ji} = j$-th row $\times i$-th column. The bottleneck of the scheme is the multiplication of two vectors of size $(\lambda + 1)$, where the elements of the vectors are of the size of the corresponding cryptographic keys. Each node receives and broadcasts one message from each node within its radio coverage where messages carry a vector of size $(\lambda + 1)$.

Multiple space key pre-distribution scheme [31] suggests an improvement of the resilience of citeBlom using a public matrix G and a set of ω private matrices D. These matrices form ω spaces (D_i, G) for $i = 1, \cdots, \omega$. For each node, a set of τ spaces are selected at random from these ω spaces. Similar to Blom's scheme, the keying materials for each selected space are stored in the nodes and thus each node stores $\tau + 1$ vectors, each of size $\lambda + 1$. In shared-key discovery phase, a pair of nodes first agree upon a common space by exchanging an extra message containing τ space IDs. If they do not have a space in common, they use path-key establishment phase to establish a key through intermediate nodes.

Scalability improvement is achieved in Multiple Space Blom's Scheme (MBS) [64]. The nodes are divided into two sets U and V to form a bipartite key connectivity graph. So every pair of nodes do not have a key in common. Also the private matrix D of Blom's scheme need not be symmetric in this case. Secret information $(u\text{-th column})^T$ of D is assigned to each node $S_u \in U$ and (v-th column) of D is assigned to each node $S_v \in V$. The nodes S_u and S_v also store public information u-th column and v-th column respectively. They can exchange their public information to calculate the secret key $(u\text{-th column})^T D(v\text{-th column})$. [64] has also proposed Deterministic Multiple Space Blom's Scheme (DMBS) for supporting larger networks by the use of l copies of strongly regular graphs R of degree r. Each vertex of R has been considered as a class of l nodes $S_u = S_{u_1}, \cdots, S_{u_l}$. An arbitrary direction is assigned to every edge in R and every edge e has a random private matrix D_e which is not necessarily symmetric. Each node S_{u_i} receives its public column vector u-th column of size $\lambda + 1$. For a directed edge $(S_{u_i}, S_{v_j}) \in R$, source node S_{u_i} receives secret information (u-th column)$^T D_{uv}$ of size $\lambda + 1$ and destination node S_{v_j} receives secret information D_{uv}(v-th column) of size $\lambda + 1$. Thus each node stores vectors of size $r(\lambda + 1)$. Nodes S_{u_i} and S_{v_j} can then generate the link key as $K_{u_i v_j}$ = (u-th column)$^T D_{uv}$(v-th column). This scheme increases scalability at the cost of decreased resilience since capture of one node results into compromise of the credentials of $l - 1$ other nodes.

Polynomial Based Dynamic Solutions

Polynomial based key pre-distribution scheme [5] distributes a polynomial share or partially evaluated polynomial to each node such that each pair of node can generate a key. A polynomial $P(x, y)$ symmetric in x and y, i.e., $P(x, y) = P(y, x)$ of degree λ is used. The coefficients of the polynomial P are chosen from a large field $\mathbb{F}_q$. Each node stores a polynomial with $\lambda + 1$ coefficients. The node S_i receives its share $f_i(y) = P(i, y).S_i$. and obtain the key as $K_{ij} = P(i, j)$ by evaluating the shares $f_i(y)$ and $f_j(y)$. This solution is also λ-secure, i.e., the coalition of less than $\lambda + 1$ sensor nodes can not calculate the keys of others. Another scheme of this type [67] is based on the fact that all pairs of nodes do not need to have a key in common. It is a combination of the polynomial based scheme [5] and key pool based schemes like [22, 34] aiming at increasing scalability and resilience. In the key setup phase, a set of λ degree polynomials over $\mathbb{F}_q$ is generated. Let us denote it by F. Each node S_i receives a subset F_i of the polynomial set F where $F_i \subseteq F$. There are many options for selecting polynomial subsets for sensor nodes. The nodes may store list of IDs of the other nodes with which it shares the polynomial. Grid based key pre-distribution may also be used. Let $m = \left\lceil \sqrt{N} \right\rceil$. Consider a $m \times m$ grid with a set of $2 \times m$ column and row polynomials $\{f_i^c(x, y), f_i^r(x, y)\}$ for $i = 0, \cdots, m - 1$ are generated. Each row i in grid is associated with a polynomial $f_i^r(x, y)$ and each column i with a polynomial $f_i^c(x, y)$. Each sensor is assigned a co-ordinate (i, j) on the grid and receives polynomials $\{f_i^c(x, y), f_i^r(x, y)\}$. A pair of nodes only need to check whether their column or row addresses overlap. In shared-key discovery phase, if two sensor nodes have the same polynomial, they can establish a key.

In yet another scheme [68], bivariate polynomials have been used for location based pair-wise keys. The deployment area is divided into R rows and C columns. It is based on [5] scheme. For each cell at cth column and rth row, a unique polynomial $f_{cr}(x, y)$ is generated. Each node stores the polynomial shares of its own cell and the four neighbouring cells. Two nodes exchange their cell co-ordinates to agree on a polynomial share. In [49], the deployment area is divided into cells over which groups of sensor nodes are uniformly distributed.

2.8.2 Group-wise Key Distribution Schemes

The direct approach is to use the existing pair-wise keys to establish group-wise keys. In [32] a lightweight key management system is considered. Group of sensor nodes are deployed in different phases. The proposal is to distribute group-wise keys through the links secured with pair-wise keys. Polynomial based key pre-distribution scheme may also be used to distribute the shares among the group members, who can generate a common group key. In [5], two models have been proposed. One is an interactive model. Initially a polynomial $P(x, y)$ of degree $\lambda + t - 2$ is selected. Each user S_i receives a share $P_i(y) = P(i, y)$. Users $S_{j_1}, \cdots, S_{j_t}$ can calculate the conference key $K_{j_1 \cdots, j_t}$ as follows:

(a) S_{j_t} selects a random key K.

(b) S_{j_t} calculates $K_{j_t j_l} = P_{j_t}(j_l) = P(j_t, j_l)$ for each $l = 1, \cdots, t-1$.

(c) S_{j_t} sends $\chi_l = K_{j_t j_l} \oplus K$ to each S_{j_l} for each $l = 1, \cdots, t-1$.

(d) Each S_{j_l} generates $K_{j_l j_t} = P_{j_l}(j_t)$ and derives the secret $K = \chi_l \oplus K_{j_l j_t}$.

Thus S_{j_t} performs $t-1$ polynomial evaluations and sends $t-1$ messages which carry a single χ value to establish a group-wise key.

The other is a non-interactive model. A random symmetric polynomial $P(x_1, \cdots, x_t)$ in t variables of degree λ is selected (over $\mathbb{F}_q$, where q is large enough to accommodate the key length of the crypto-system under consideration). Each user S_i receives share $P_i(x_2, \cdots, x_t) = P(i, x_2, \cdots, x_t)$. Users $S_{j_1}, \cdots, S_{j_t}$ can generate the conference key $K_{j_1, \cdots, j_t}$ by evaluating their polynomial shares. Each user S_{j_i} can evaluate $P_{j_i}(j_1, \cdots, j_{i-1}, j_{i+1}, \cdots, j_t)$.

2.9 Key Distribution in HWSN

In HWSN, there are one or more powerful nodes called base stations and they often act as key distribution centres. Base stations may share pair-wise keys with all the nodes which may be subsequently used to establish the other keys. This may be also be divided into three categories as follows:

2.9.1 Pair-wise Schemes

IN HWSNs, the communication between base station and sensor node require pair-wise keys. This is easy if the base station shares pair-wise keys with each of the sensor nodes. This kind of solutions have been proposed in Perimeter Protection Scenario [98], Base Station Authentication Protocols [23, 26, 27] and LEAP [103]. The base station can act as an intermediary to establish the pair-wise key between any two sensor nodes. In ESA [63], a scenario is described where the nodes are divided into domains and each domain is supervised by a base station. In SNEP [76], it is proposed that each pair of communicating nodes S_A and S_B will share a master secret key χ_{AB} and a pseudo random function f. S_A and S_B then generate the keys $K_{AB} = f(\chi_{AB}, 1)$ and $K_{BA} = f(\chi_{AB}, 3)$ and MAC keys $K'_{AB} = f(\chi_{AB}, 2)$ and $K'_{BA} = f(\chi_{AB}, 4)$. LEAP [103] proposes that each node can establish pair-wise keys with its immediate neighbours. In the key setup phase, nodes receive a general key K_I. A node S_u can use K_I and one-way hash function H to generate its master key $K_u = H_{K_I}(ID_u)$. In shared key discovery phase, node S_u broadcasts ID_u, RN_u and a neighbour S_v responds with $(ID_v, MAC_{K_v}(RN_u|ID_v))$. Node S_u can then generate the key $K_v = H_{K_I}(ID_v)$ and both nodes S_u and S_v can generate the session key $K_{uv} = H_{K_v}(ID_u)$. To reach the cluster heads, multi-hop pair-wise keys may be required. In that case, node S_u generates secret $K_{u,c}$ and finds m intermediate nodes. It divides the secrets into shares $K_{uc} = sk_1 \oplus \cdots \oplus sk_m$ and sends each share through a separate intermediate node S_{v_i} where $1 \leq i \leq m$. This solution has high communication cost since S_u sends m messages through m intermediate nodes to increase resilience. Security of the system depends on K_I, which may be revealed by the compromise of a sensor node. Once K_I is compromised, all the session keys may also be compromised.

2.9.2 Group-wise Schemes

This kind of schemes are necessary to secure multicast communications. A secure but costly solution is asymmetric cryptography. There are works based on Diffie-Hellman group key transport protocol, like Burmester-Desmedt [11] and IKA2 [94]. Further improvement is made in ID-STAR algorithm [15], which uses identity based cryptography [8, 86], i.e., the public keys of the nodes are derived from their identities. Pair-wise keys can also be used to establish group-wise keys. The base station may act as an intermediary to establish pair-wise keys between any pair of nodes provided the base station shares keys with all the nodes. In LEAP [103], a mechanism is given to generate group-wise keys which follows LEAP pair-wise key establishment phase. If S_u wants to establish a group key with its neighbours $S_{v_1}, \cdots, S_{v_m}$, it first generates a unique group key K_u^g and then sends it to the neighbours S_{v_i} by encrypting it with K_{uv_i}. The security of this scheme is dependent on the security of the pair-wise keys (and it has very low resilience).

2.9.3 Network-wise Schemes

Master Key Based Solutions

In HWSN, base station to sensor node broadcast traffic is secured with network-wise keys. The obvious insecure approach is to pre-distribute a single network-wise key to all sensor nodes. A better approach is proposed by Multi-tiered security solution [91]. The data items are protected to a degree consistent with their value. Three types of data in a WSN are considered: mobile code, locations of sensor nodes and application data. It assumes that sensor nodes are initially loaded with a list of m master keys, a pseudo random function and a seed. They use the pseudo random function with the seed to obtain an index within the list of master keys. The selected key is called active master key and is used to secure communication. $RC6$ is used as encryption algorithm. Three different security levels are defined. In level I, a strong encryption algorithm and active master key is used to secure mobile codes. In level II, sensors are divided into cells. A common location security key is generated within each cell and is used to secure location information. Finally in level III, $MD5$ hash of the active master key is used to secure application data. The drawback of the scheme is that the master key list, the pseudo random function and the seed may be compromised.

TESLA Based Solutions

TESLA stands for Timed Efficient Stream Loss-tolerant Authentication [77]. It is a multicast stream authentication protocol. It uses delayed key disclosure mechanism where the key used to authenticate the ith message is disclosed along with $(i+1)$-th message. SPINS [76] uses μ-TESLA, an adoption of TESLA for HWSNs. SPINS uses base stations as key distribution centres. μ-TESLA provides authentication for data broadcasts and requires that base station and sensor nodes be loosely time synchronised. . Basically, the base station randomly selects the last key K_n of a chain and applies one-way public function H to generate the rest of the chain $K_0, \cdots, K_{n-1}$ as $K_i = H(K_{i+1})$. Given K_i, every node can generate the sequence $K_0, \cdots, K_{i-1}$, but given K_i, one can not generate K_{i+1}. At ith time slot, the base station sends authenticated message $MAC_{K_i}(Message)$. Sensor nodes store the message until the base station discloses the verification key in $(i+1)$th time slot. The nodes can verify the disclosed verification key $K_{i+1}$by using the previous key K_i as $K_i = H(K_{i+1})$. In μTESLA, nodes are required to store a message until the authentication key is disclosed. This operation may create storage problems and encourages DoS types of attacks. An adversary may jam key disclosure messages to saturate storages of sensor nodes. μTESLA requires the sensor nodes to bootstrap from the base station, i.e., the nodes receive the first key of the chain (called the key chain commitment). Bootstrapping requires unicast communication and can be secured with pair-wise keys. Also μTESLA is used in [23, 26, 27]to authenticate message broadcasts from base station, in [93] to authenticate route update broadcasts and in LEAP [103] to update pre-deployed network-wise keys in case of a node compromise. Another variant of $TESLA$ is $TESLA\ Certificate$ [6], where a base station is used as a certificate authority(CA). In this scheme, CA generates certificate $Cert(ID_A, t_{i+d}, \cdots, MAC_{K_i}(\cdots))$ for sensor node S_A at time t_i. It discloses the TESLA key K_i at time t_{i+d} when the certificate expires.

Bootstrapping of key chain commitments in μTESLA causes high volume of packets flowing in WSN and creates scalability problems. μTESLA extensions [69, 70] propose five extensions to address scalability issues. In predetermined key chain commitment, commitment is pre-distributed to sensors before the deployment. In this solution, key chain must cover lifetime of sensor nodes to prevent bootstrapping requirements. This can be achieved by either using long chains or large time intervals. A new node has to generate the whole key chain from the beginning to authenticate recently disclosed key. Thus, long key chain means excessive processing for sensor nodes joining later. Large time interval means increased number of messages to store because sensor nodes have to store incoming messages until the authentication key is disclosed. Two-level key chain scheme tries to address these problems. There is a high level key chain with long enough time interval to cover the life time of sensor nodes and multiple low-level key chains with short enough time intervals. High level key chain is used to distribute and authenticate randomly generated commitments of low-level key chains. The nodes are initialised with the commitment of high-level chain, time intervals of high-level and low-level key chains and one way functions of high and low-level chains. Low-level keys are not chained together and loss of a low level key disclosure can only be recovered with a key which is disclosed later within the same interval. Loss of a low level key commitment may also mean loss of entire interval. An adversary may take advantage of this and may jam disclosure of low-level key commitments. Fault tolerant two level key chains scheme is proposed to address these issues. In this scheme, the commitments of low level key chains are not randomly generated, but obtained from high-level keys by using another one way function. The low level key commitments are periodically broadcast; however, an adversary may still recover the commitment period and can jam disclosure of low level key commitments. Fault tolerant two level key chains with random commitments scheme uses a random process to broadcast the low-level commitments. Finally multi-level chains scheme is proposed to provide smaller time intervals and shorter key chains.

2.10 Ongoing Research Areas

Handling compromised nodes is one of the most difficult problems in the security of sensor networks.

(a) The memory content of a compromised node is bound to be different from that on a legitimate node and thus code attestation through hardware or software solves this problem. Though there are examples of hardware solutions, little research has been done so far in software attestation techniques.

(b) Detection of secure misbehaviour detection and node revocation is another important area of research. In [22], this is implemented through a voting scheme.

(c) Examples of secure routing is given in [46, 52, 56], though [46] discusses the routing protocols for ad hoc networks. These protocols may not always be suitable for sensor networks and more specific research is needed in this area.

(d) Secure localisation becomes important because of two reasons: a node should be able to determine its own location and a malicious nose should not be able to claim a false position. The first part is studied in [13] and the second part is discussed in [14, 83]. This is helpful in preventing Sybil attack and wormhole attack [47].

(e) Cryptographic techniques have been discussed in [54, 76]. Use of asymmetric cryptography is discussed in [40].

(f) In secure multi-party communication systems, group key is required and members join in and leave from party frequently. So group key materials of all members of the network must be updated. An efficient group key distribution method using M-ary coding for a key message without using FEC (Forward Error Correction) and an ARQ (Automatic Repeat reQuest) and transmitting this in parallel with the non-key message is discussed in [75].

Chapter 3

Preliminaries

3.1 Basics of Combinatorial Design

In this section, we give explicit definitions of some of the designs we will use in this thesis. For more detailed account of difference sets and combinatorial designs, please refer to [9, 10, 24, 28, 96, 97].

Let B be a finite set of subsets (also known as blocks) of a finite set X.

Definition 1 *A* set system *or* design *is a pair* (X, B).

Definition 2 *The degree of a point* $x \in X$ *is the number of subsets containing the point* x.

If all subsets/blocks have the same size k, then (X, B) is said to be uniform of rank k. If all points have the same degree r, (X, B) is said to be regular of degree r.

Definition 3 *A regular and uniform set system is called a* (v, b, r, k)*-1 design, where* $|X| = v, |B| = b$*,* r *is the degree and* k *is the rank.*

Definition 4 *A* (v, b, r, k)*-1 design is called a* (v, b, r, k) *configuration if any two distinct blocks intersect in zero or one point.*

Definition 5 *A* (v, b, r, k, λ) *BIBD is a* (v, b, r, k)*-2 design in which every pair of points occurs in exactly* λ *many blocks. We note that* $bk = vr$ *and* $\lambda(v-1) = r(k-1)$.

Definition 6 *A* $BIBD(v, b, r, k, \lambda)$ *where* $v = b$ *and* $r = k$ *is called an* $SBIBD(v, v, k, k, \lambda)$*-2 design or simply an* $SBIBD(v, k, \lambda)$ *design. We note that* $\lambda(v-1) = k(k-1)$.

Definition 7 *A pairwise balanced design (PBD) is a design in which each pair of points occurs in λ blocks, for some constant λ, called the index of the design.*

Definition 8 *A linked design is a design in which each pair of blocks intersect in μ points, for some constant μ, sometimes called the linkage of the design.*

Definition 9 *A variety and a block are said to be incident if the variety belongs to the block. One convenient way to represent a design is by means of an incidence matrix. For a design (X, B) with v varieties and b blocks, the incidence matrix is a $v \times b$ matrix, $A = [a_{ij}]$, such that*

$$\begin{aligned} a_{ij} &= 1 \quad \textit{if variety } i \textit{ belongs to block } j, \\ &= 0 \quad \textit{otherwise.} \end{aligned}$$

Definition 10 ***Starter blocks and developed blocks***

Let v be an integer and consider the set of integers modulo v, i.e., $\mathbb{Z}_v = \{0, 1, 2, \cdots, v-1\}$, whose addition and multiplication are denoted by the usual symbols $+$ and $\cdot$ respectively. Let $B = \{i_1, i_2, \cdots, i_k\}$ be a subset of $\mathbb{Z}_v$ consisting of k integers. For each integer $j \in \mathbb{Z}_v$, we define $B + j = \{i_1 + j, i_2 + j, \cdots, i_k + j\}$ to be the subset of $\mathbb{Z}_v$ by adding mod *v the integer j to each of the integers in B. The v sets $B = B + 0, B + 1, B + 2, \cdots, B + v - 1$ so obtained are called the blocks developed from the block B and B is called the starter block.*

Definition 11 ***Difference set*** mod v

Let B be a subset of k integers in $\mathbb{Z}_v$. Then B is called a difference set mod *v, provided each non-zero integer in $\mathbb{Z}_v$ occurs the same number of times among the $k(k-1)$ differences among distinct elements of B (in both order): $x - y$ where $x, y \in B$; $x \neq y$.*

Since there are $v - 1$ non-zero integers in $\mathbb{Z}_v$, each non-zero integer in $\mathbb{Z}_v$ must occur $\lambda = \frac{k(k-1)}{v-1}$ times as a difference in a difference set.

Theorem 1 *Let B be a subset of $k < v$ elements of $\mathbb{Z}_v$ which forms a difference set* mod *v. Then the blocks developed from B as a starter block form a SBIBD with index $\lambda = \frac{k(k-1)}{v-1}$.*

Definition 12 *An association scheme with m associate classes on a v-set X is a family of m-many symmetric, anti-reflexive binary relations on X such that:*

1. *any two distinct elements of X are i-th associates for exactly one value of i, where $1 \leq i \leq m$.*

2. *each element of X has n_i i-th associates, $1 \le i \le m$.*

3. *for each i, $1 \le i \le m$, if x and y are i-th associates, then there are p^i_{jl} elements of X which are both j-th associates of x and l-th associates of y.*

The numbers v, n_i $(1 \le i \le m)$ and p^i_{jl} $(1 \le i, j, l \le m)$ are called the parameters of the association scheme. We see that $p^i_{jl} = p^i_{lj}$.

Definition 13 *A partially balanced incomplete block design with m associate classes $(PBIBD(m))$ is a design based on a v-set X, with b blocks, each of size k and with replication number r, such that there is an association scheme with m classes defined on X, where, if elements x and y are i-th associates, $1 \le i \le m$, then they occur together in precisely λ_i blocks. The numbers $v, b, r, k, \lambda_i (1 \le i \le m)$ are called the parameters of the $PBIBD(m)$. We denote such a design by $PB[k, \lambda_1, \cdots, \lambda_m; v]$.*

Definition 14 *Let X be a set of v varieties such that*

$$X = \bigcup_{i=1}^{m} G_i, \ |G_i| = n \ for \ 1 \le i \le m, G_i \bigcap G_j = \phi \ for \ i \ne j.$$

The G_i's are called groups (though they are not groups in the usual algebraic sense) and an association scheme defined on X is said to be group divisible if the varieties in the same group are first associates and those in different groups are second associates.
Designs in which the underlying association scheme is group divisible are called group divisible (GD) designs.

Definition 15 *A transversal design with k groups of size n and index λ, denoted by $T[k, \lambda; n]$, is a triple $(X, \mathcal{H}, \mathcal{A})$ where:*

1. *X is a set of kn varieties;*

2. *$\mathcal{H} = \{H_1, H_2, \cdots, H_k\}$ is a family of k n-sets (or groups) which form a partition of X.*

3. *$\mathcal{A}$ is a family of k-sets (or blocks) of varieties such that each k-set in $\mathcal{A}$ intersects each group H_i in precisely one variety and any pair of varieties which belong to different groups occur together in precisely λ blocks in $\mathcal{A}$.*

Remark 1 *It may be noted that the transversal designs are group divisible designs with $\lambda_1 = 0$ and $\lambda_2 = \lambda$.*

If $\lambda = 1$, we simply denote the transversal design by $TD(k, n)$.

Let us now describe the construction of a transversal design [65].

Construction 1 *Let p be a prime and $2 \leq k \leq p$.*

1. *Define $X = \mathbb{Z}_k \times \mathbb{Z}_p$.*
2. *For $0 \leq x \leq k-1$, define $H_x = \{x\} \times \mathbb{Z}_p$ and $\mathcal{H} = \{H_x : 0 \leq x \leq k-1\}$.*
3. *For every ordered pair $(i, j) \in \mathbb{Z}_p \times \mathbb{Z}_p$, define a block $A_{i,j} = \{(x, (ix + j) \bmod p) : 0 \leq x \leq k-1\}$ and $\mathcal{A} = \{A_{i,j} : (i, j) \in \mathbb{Z}_p \times \mathbb{Z}_p\}$.*

It can be shown that $(X, \mathcal{H}, \mathcal{A})$ is a $TD(k, p)$.

We shall present a detailed example of the aforesaid construction, which will be subsequently used in chapter 4.

Example 1 *Let $k = 3$ and $p = 5$.*

1. *Here, $X = \mathbb{Z}_3 \times \mathbb{Z}_5 = \{(0,0), (0,1), (0,2), (0,3), (0,4), (1,0), (1,1), (1,2), (1,3), (1,4), (2,0), (2,1), (2,2), (2,3), (2,4)\}$.*
2. *$H_0 = \{(0,0), (0,1), (0,2), (0,3), (0,4)\}$,*
   $H_1 = \{(1,0), (1,1), (1,2), (1,3), (1,4)\}$ and
   $H_2 = \{(2,0), (2,1), (2,2), (2,3), (2,4)\}$.
   Also, $\mathcal{H} = \{H_0, H_1, H_2\}$.
3.

$$A_{0,0} = \{(0,0), (1,0), (2,0)\}$$
$$A_{0,1} = \{(0,1), (1,1), (2,1)\}$$
$$A_{0,2} = \{(0,2), (1,2), (2,2)\}$$
$$A_{0,3} = \{(0,3), (1,3), (2,3)\}$$
$$A_{0,4} = \{(0,4), (1,4), (2,4)\}$$

$$A_{1,0} = \{(0,0), (1,1), (2,2)\}$$
$$A_{1,1} = \{(0,1), (1,2), (2,3)\}$$
$$A_{1,2} = \{(0,2), (1,3), (2,4)\}$$

$$A_{1,3} = \{(0,3),(1,4),(2,0)\}$$
$$A_{1,4} = \{(0,4),(1,0),(2,1)\}$$

$$A_{2,0} = \{(0,0),(1,2),(2,4)\}$$
$$A_{2,1} = \{(0,1),(1,3),(2,0)\}$$
$$A_{2,2} = \{(0,2),(1,4),(2,1)\}$$
$$A_{2,3} = \{(0,3),(1,0),(2,2)\}$$
$$A_{2,4} = \{(0,4),(1,1),(2,3)\}$$

$$A_{3,0} = \{(0,0),(1,3),(2,1)\}$$
$$A_{3,1} = \{(0,1),(1,4),(2,2)\}$$
$$A_{3,2} = \{(0,2),(1,0),(2,3)\}$$
$$A_{3,3} = \{(0,3),(1,1),(2,4)\}$$
$$A_{3,4} = \{(0,4),(1,2),(2,0)\}$$

$$A_{4,0} = \{(0,0),(1,4),(2,3)\}$$
$$A_{4,1} = \{(0,1),(1,0),(2,4)\}$$
$$A_{4,2} = \{(0,2),(1,1),(2,0)\}$$
$$A_{4,3} = \{(0,3),(1,2),(2,1)\}$$
$$A_{4,4} = \{(0,4),(1,3),(2,2)\}$$

Let us relabel the elements of the blocks as shown in table 3.1.
Finally,

$$A_{0,0} = \{K_0, K_1, K_2\}$$
$$A_{0,1} = \{K_3, K_4, K_5\}$$
$$A_{0,2} = \{K_6, K_7, K_8\}$$
$$A_{0,3} = \{K_9, K_{10}, K_{11}\}$$
$$A_{0,4} = \{K_{12}, K_{13}, K_{14}\}$$

$$A_{1,0} = \{K_0, K_4, K_8\}$$

Actual value	Label
$(0,0)$	K_0
$(1,0)$	K_1
$(2,0)$	K_2
$(0,1)$	K_3
$(1,1)$	K_4
$(2,1)$	K_5
$(0,2)$	K_6
$(1,2)$	K_7
$(2,2)$	K_8
$(0,3)$	K_9
$(1,3)$	K_{10}
$(2,3)$	K_{11}
$(0,4)$	K_{12}
$(1,4)$	K_{13}
$(2,4)$	K_{14}

Table 3.1: Labels

$$A_{1,1} = \{K_3, K_7, K_{11}\}$$
$$A_{1,2} = \{K_6, K_{10}, K_{14}\}$$
$$A_{1,3} = \{K_9, K_{13}, K_2\}$$
$$A_{1,4} = \{K_{12}, K_1, K_5\}$$

$$A_{2,0} = \{K_0, K_7, K_{14}\}$$
$$A_{2,1} = \{K_3, K_{10}, K_2\}$$
$$A_{2,2} = \{K_6, K_{13}, K_5\}$$
$$A_{2,3} = \{K_9, K_1, K_8\}$$
$$A_{2,4} = \{K_{12}, K_4, K_{11}\}$$

$$A_{3,0} = \{K_0, K_{10}, K_5\}$$
$$A_{3,1} = \{K_3, K_{13}, K_8\}$$
$$A_{3,2} = \{K_6, K_1, K_{11}\}$$
$$A_{3,3} = \{K_9, K_4, K_{14}\}$$

$$A_{3,4} = \{K_{12}, K_7, K_2\}$$

$$A_{4,0} = \{K_0, K_{13}, K_{11}\}$$
$$A_{4,1} = \{K_3, K_1, K_{14}\}$$
$$A_{4,2} = \{K_6, K_4, K_2\}$$
$$A_{4,3} = \{K_9, K_7, K_5\}$$
$$A_{4,4} = \{K_{12}, K_{10}, K_8\}$$

Now let us relate a $(v = kr, b = r^2, r, k)$ configuration (resulting from the $TD(k, r)$) with sensor nodes and keys. This will also be useful as a background for section 3.4. X is the set of $v = kr$ number of keys distributed among $b = r^2$ number of sensor nodes. The nodes are indexed by $(i, j) \in \mathbb{Z}_r \times \mathbb{Z}_r$ and the keys are indexed by $(i, j) \in \mathbb{Z}_k \times \mathbb{Z}_r$. Consider a particular block $A_{\alpha,\beta}$. It will contain k keys $\{(x, (x\alpha + \beta) \bmod r) : 0 \leq x \leq k - 1\}$. Here $|X| = kr = v$, $|\mathcal{H}_x| = r$, the number of blocks in which the key (x, y) appears for $y \in \mathbb{Z}_r$, $|A_{i,j}| = k$, the number of keys in a block.

Note that if r is a prime power, we will not get an inverse of $x \in \mathbb{Z}_r$ when $\gcd(x, r) > 1$. This is required for key exchange protocol (see Section 4.4). So basically we should consider the field $GF(r)$ instead of the ring $\mathbb{Z}_r$. However, there is no problem when r is a prime by itself. In this chapter, we generally use $\mathbb{Z}_r$ since in our examples we consider r to be prime.

Definition 16 *A projective plane consists of a set of lines and a set of points, and a relation between points and lines called incidence, having the following properties:*

1. *Given any two distinct points, there is exactly one line incident with both of them.*
2. *Given any two distinct lines, there is exactly one point incident with both of them.*
3. *There are four points such that no line is incident with more than two of them.*

The second condition means that there are no parallel lines. The last condition simply excludes some degenerate cases.

A projective plane is therefore a symmetric $(n^2 + n + 1, n + 1, 1)$ block design or $SBIBD(n^2 + n + 1, n + 1, 1)$.

A finite projective plane exists when the order n is a power of a prime, i.e., for $n = p^a,\ a \geq 1$. It is conjectured that these are the only possible projective planes, but proving this remains one of the most important unsolved problems in combinatorics.

The smallest finite projective plane is of order $n = 2$ and consists of the configuration known as the Fano plane. The remarkable Bruck-Ryser-Chowla theorem [97] says that if a projective plane of order n exists and $n = 1 \pmod 4$ *or* $2 \pmod 4$, then n is the sum of two squares. This rules out $n = 6$. Lam [61] showed, using massive computer calculations on top of some mathematics, that there are no finite projective planes of order 10. The status of the order 12 projective plane remains open.

The projective plane of order 2 is denoted by PG(2, 2). It has incidence matrix

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Every row and column contains three 1s and any pair of rows/columns has a single 1 in common.

3.2 Incidence Matrix of a Sensor Network

We propose to model our sensor network using a (0,1) incidence matrix, N, called the *incidence matrix of a sensor network.*

Definition 17 ***The incidence matrix of a sensor network***
Let $N = (N_{ij})$ be a $v \times b$ matrix with entries 0 or 1 where the v rows denote the sensor nodes in a sensor network and the b blocks denote the keys in the sensor network. $n_{ij} = 1$ if key "j" is in sensor "i" and 0 otherwise, that is, $(i, j) = 1$ denotes that node "i" contains key "j". N will have r_i elements 1 in row "i" and k_j elements 1 in column "j".

Lemma 1 *The incidence matrix of a sensor network with v keys and b sensors contains*

$$\sum_{i=1}^{v} r_i = \sum_{j=1}^{b} k_j$$

ones.

We note some general properties we would like to have for a sensor network as indicated by the incidence matrix N:

1. v, the number of nodes should be large. We should be able to enlarge v to easily allow scalability.

2. b, the number of keys should not be too large to enable manageability.

3. Since each node/sensor has limited power and space resources, $\max\limits_i r_i$ should be as small as possible.

4. Since we wish to have as much resilience as possible we would like each key to be widely distributed so the compromise of a node does not adversely affect the reliability of the network. At the same time we want to limit the distribution of each key so that if a key is compromised the network will still be able to function. Thus if key "i" is compromised in all nodes/sensors which contain it, then $\min\limits_i k_i$ gives a lower bound on the number of sensors compromised in the network. Note this is not the minimum for the whole network.

5. $(i_1, j) = (i_2, j)$ means key j is in both sensors/nodes i_1 and i_2.

6. $(i, j_1) = (i, j_2)$ means keys j_1 and j_2 are both in node i. If key j_1 is compromised, the network will continue to function as is, provided removal of column j_1 from N leaves the network connected. (A rank argument on the matrix N at any stage allows us to determine connectivity).

7. In (v, b, r, k) configurations (as traditionally used by combinatorialists), λ or λ_j, the number of common keys between a pair of nodes, or *pair property*, is extensively studied. This is studied by the inner product of any pair of rows of N. Early sensor network studies using (v, b, r, k) configurations implicitly used λ or λ_j as 0 or 1. In this thesis we will not impose that restriction.

8. In sensor networks, the number of common keys that occur between different pairs of nodes, is equivalent to the *linkage property*, which is studied by considering the inner product of the rows of N, has received far less study. We will use μ or μ_j to denote the linkage of N. The linkage parameter gives us a metric for determining the number of nodes/sensors that might be compromised if a node is compromised.

Remark 2 *We note that an SBIBD has both the pair property and the linkage property. A BIBD*(v, b, r, k, λ) *configuration has the pair property while its transpose, a* (b, v, k, r) *configuration has the linkage property. Hence we shall prefer to consider the rows of the incidence matrix as the sensor nodes and the columns as the keys.*

3.3 Hopping Between Levels in an HWSN

If two sensor nodes share a common key, they are able to communicate directly with each other. We call it a single-hop communication or direct communication.

Similarly one can think of communication between two sensor nodes as a sequence of several single-hop communications. We call it multi-hop communication.

When the incidence matrix of a $SBIBD(v, k, 1)$ is used as the incidence matrix of the sensor network, any two sensor nodes can communicate in a single hop. This direct communication takes place between any two sensor nodes at the expense of storing large number of keys in each sensor node.

When the incidence matrix of a complete transversal design, with $\lambda = 1$ or 0 is used as the incidence matrix of the sensor network, every sensor node can reach every other sensor node in at most two hops with almost certainty.

Example 2 *As shown in example 1, the sensor nodes* $A_{0,0} = \{K_0, K_1, K_2\}$ *and* $A_{1,2} = \{K_6, K_{10}, K_{14}\}$ *do not share a common key. However,* $A_{0,0}$ *can reach* $A_{1,2}$ *via* $A_{2,0} = \{K_0, K_7, K_{14}\}$*, or* $A_{3,0} = \{K_0, K_{10}, K_5\}$*.*

3.4 Lee-Stinson Approach [65]

Consider a (v, b, r, k) configuration (which is in fact a (rk, r^2, r, k) configuration). There are $b = r^2$ many sensor nodes, each containing k distinct keys. Each key is repeated in r many nodes. Also v gives the total number of distinct keys in the design. One should note that $bk = vr$ and $v - 1 > r(k - 1)$. The design provides 0 or 1 common key between two nodes. The design $(v = 1470, b = 2401, r = 49, k = 30)$ has been used as an example in [65]. The important parameters of the design are as follows:

1. **Expected number of common keys between two nodes:** This value is $p_1 = \frac{k(r-1)}{b-1} = \frac{k}{r+1}$. In the given example, $p_1 = \frac{30}{49+1} = 0.6$.

2. **Consider an intermediate node:** There is a good proportion of pairs (40%) with no common key and two such nodes will communicate through an intermediate node. Assuming a random geometric deployment, the example shows that the expected proportion such that two nodes are able to communicate either directly or through an intermediate node is as high as 0.99995.

3. **Resiliency:** Under adversarial situation, one or more sensor nodes may get compromised. In that case, all the keys present in those nodes cannot be used for secret

communication any longer, i.e., given the number of compromised nodes, one needs to calculate the proportion of links that cannot be used further. The expression for this proportion is

$$fail(s) = 1 - \left(1 - \frac{r-2}{b-2}\right)^s,$$

where s is the number of nodes compromised. In this particular example, $fail(10) \approx 0.17951$. That is, given a large network comprising as many as 2401 nodes, even if only 10 nodes are compromised, almost 18% of the links become unusable.

Chapter 4

Key Pre-distribution Schemes for Wireless Sensor Networks: Merging Blocks in Combinatorial Designs

4.1 Introduction

Recently secure communication among sensor nodes has become an active area of research [12, 22, 31, 34, 64, 65, 67]. One may refer to [53] for broader perspective in the area of sensor networks. Based on the architectural consideration, wireless sensor networks may be broadly classified into two categories viz. (i) Hierarchical Wireless Sensor Networks (HWSN) and (ii) Distributed Wireless Sensor Networks (DWSN).

In HWSN, there is a pre-defined hierarchy among the participating nodes. There are three types of nodes in the descending order of capabilities: (a) base stations, (b) cluster heads and (c) sensor nodes.

The sensor nodes are usually placed in the neighbourhood of the base station. Sometimes the network traffic (data) is collected by the cluster heads which in turn forward the traffic to the base station. There may be three different modes of data flow as follows: Unicast (sensor to sensor), multicast (group wise), broadcast (base station to sensor). However, it may be pointed out that the HWSN is best suited for applications where the network topology is known prior to deployment. On the other hand, there is no fixed infrastructure in the case of a DWSN and the network topology is unknown before the deployment. Once the nodes are scattered over the target area, the nodes scan their radio coverage area and find out their neighbours. In this case also, the data flow may be divided into three categories (as discussed above) with the only difference that the broadcast might take place between any two nodes. Unless mentioned otherwise, we shall always talk about DWSNs. Hence all the nodes are equal in their capabilities.

Consider a scenario where N number of sensor nodes are dropped from an airplane in the battlefield. Thus the geographical positioning of the nodes cannot be decided a priori. However, any two nodes in radio frequency range are expected to be able to communicate securely. One option is to maintain different secret keys for each of the pairs. Then each of the nodes needs to store $N-1$ keys. Given (i) the large number of sensor nodes generally deployed, (ii) the memory constraint of the sensor nodes, this solution is not practical. On the other hand, on-line key exchange is not very popular till date since implementation of public key framework demands processing power at the higher end. Very recently implementations of ECC and RSA on 8-bit CPUs have been proposed [40]. Still a closer scrutiny of [36, Table 2, Section 3.3] reveals that the algorithms execute in seconds (the range being 0.43s to 83.26s); whereas the key pre-distribution just involves the calculation of inverse of an integer modulo a prime number, which is bound to be much faster than the former.

Hence key pre-distribution to each of the sensor nodes before deployment is a thrust area of research and the most used mathematical tool for key pre-distribution is combinatorial design. Each of the sensor nodes contains M many keys and each key is shared by Q many nodes, (thus fixing M and Q) such that the encrypted communication between two nodes may be decrypted by at most $Q-2$ other nodes if they fall within the radio frequency range of the two communicating nodes. Similarly one node can decrypt the communication between any two of at most $M(Q-1)$ nodes if it lies within the radio frequency range of all the nodes who share a key with it.

Let us present an exact example from [65]. Take $N = 2401, M = 30, Q = 49$. The parameters are obtained using a transversal design (for a basic introduction to transversal designs, refer to [97, Page 133]). It has been shown that two nodes share either 0 or 1 key. In this case, $M(Q-1)$ gives the number of nodes with which one node can communicate. The expected number of keys that is common between any two nodes is $\frac{M(Q-1)}{N-1} = 0.6$, (in [65], this is called the probability that two nodes share a common key). Further, it can be checked that if two nodes do not share a common key, then they may communicate via another intermediate node. Let nodes ν_i, ν_j do not share a common key, but ν_i, ν_k share a common key and ν_k, ν_j share a common key, i, j, k are all distinct. Hence the secret communication between ν_i and ν_k needs a key (encrypted by ν_i, decrypted by ν_k) and that between ν_k and ν_j needs another secret key (encrypted by ν_k, decrypted by ν_j). It has been shown in [65] that the communication between two nodes is possible in almost 0.99995 proportion of cases (this is based on some assumptions on the geometric distribution of nodes, which we do not use for our analysis). However, the following problems are immediate:

1. Communication between any two nodes in 60% of the cases will be in one step (no involvement of any other node), but the communication between any two of them

needs two steps for the rest 40% of the cases, making the average of 1.4 steps in each communication. This is an overhead. Thus we need a design where we can guarantee that there is a common key between any two nodes.

2. The direct communication between any two nodes can be decrypted by at most $Q-2$ other nodes. However, if one takes the help of a third intermediate node, then the communication can be decrypted by at most $2(Q-2)$ nodes. Thus any communication can be decrypted by at most $1.4(Q-2)$ many nodes on an average.

3. In an adversarial situation, if s many nodes are compromised, it has been shown that $1-(1-\frac{Q-2}{N-2})^s$ proportion of links becomes unusable. In this specific design, for $s = 10$, out of 2401 nodes, the proportion of unusable links becomes as high as 17.95%.

The solution to all these problems is based on the fact that we need to increase the number of common keys between any two nodes. The issues at this point are as follows:

1. The number of keys to be stored in each node will clearly increase. So one needs to decide the availability of storage space. In [65, Page 4], it has been commented that storing 150 keys in a sensor node may not be practical. On the other hand, in [31, Page 47], [64, Section 5.2], scenarios have been described with 200 many keys. If one considers 4 Kbytes of memory space for storing keys in a sensor node, then choosing 128-bit key (16 byte), it is possible to accommodate 256 many keys.

2. It is not easy to find out combinatorial designs with pre-specified number of common keys (say for example 5) among any two nodes for key pre-distribution [24, 96]. Consider the following technique. Generally a sensor node corresponds to a block in combinatorial design [12, 65]. Here we "merge" (defined below) a few blocks to get a sensor node. Thus the key space at each node gets increased and the number of common keys between any two nodes can also be increased to the desired level. It will be shown that this technique provides a much better control over the design parameters in key pre-distribution algorithms.

3. Further it is also shown that by this random merging strategy, one gets more flexible parameters than [65].

Definition 18 *Merging of two blocks: Two blocks (or sets of keys) P and Q are said to be merged to form a new sensor node R when the new node R contains the multiset of keys contained in either of the blocks P and Q, with repetition of element (key) instances allowed.*

Thus the goal in this chapter is to present a randomized block merging based design strategy that originates from transversal design. We differ from the existing works where it is considered that any two nodes will have either 0 or 1 common key and motivate a design strategy with more number of common keys. This is important from resiliency consideration in an adversarial framework since if certain nodes are compromised, the proportion of links that becomes unusable can be kept low, i.e., the connectivity of the network is less disturbed.

The computation to find out a common key is also shown to be of very low time complexity under this paradigm as explained in Section 4.4. Note that Blom's scheme [4] has been extended in recent works for key pre-distribution in wireless sensor networks [31, 64]. The problem with these kinds of schemes is the use of several multiplication operations (as example see [31, Section 5.2]) for key exchange.

The randomized key pre-distribution is another strategy in this area [34]. However, the main motivation is to maintain the connectivity (possibly with several hops) in the network. As example [34, Section 3.2], a sensor network with 10000 nodes has been considered and to maintain the connectivity, it has been calculated that it is enough if one node can communicate with only 20 other nodes. Note that the communication between any two nodes may require a large number of hops. However, as we discussed earlier, only the connectivity criterion (with too many hops) can not suffice in an adversarial condition. Further in such a scenario, the key agreement between two nodes requires exchange of the key indices.

The use of combinatorial and probabilistic design (also a combination of both – termed as hybrid design) in the context of key distribution has been proposed in [12]. In this case also, the main motivation was to have low number of common keys as in [65]. On the other hand we propose the idea of good number of common keys between any two nodes. The novelty of our approach is to start from a combinatorial design and then apply a probabilistic extension in the form of random merging of blocks to form the sensor nodes and in this case there is good flexibility in adjusting the number of common keys between any two nodes.

Note that in our approach, we first consider the block merging strategy in a completely randomized fashion. In such a case there is a possibility that the constituent blocks (which are merged to get a sensor node) may share common keys among themselves. This is a loss in terms of the connectivity in the designed network as no shared key is needed since there is no necessity for 'intra-node communication'. Thus we further consider a merging strategy towards minimizing the number of common keys among the blocks that are being merged. We present a heuristic for this and it works better than our initial random merging strategy. The scheme is a hybrid one as combinatorial design is followed by a heuristic.

The first section introduces the topic. In the next section, the different definitions and results from Combinatorial Design are summarised. The main idea of [65] is also presented in this section. The next section discusses our merging strategy and presents the important mathematical results and theorems. After that, some heuristic improvements are suggested along with extensive experimental results. A comparison between all these schemes is also given. The key exchange protocol is described next followed by the conclusion and future directions of research.

4.2 Our Strategy: Merging Blocks in Combinatorial Design

4.2.1 Probability Model

Before we present our strategy, a few words on the probability model are in order.

Consider two nodes N_1 and N_2, each formed by merging z blocks. Each of these blocks contain k keys. Since these blocks are taken from the $TD(k, r)$, any two of these blocks have either 0 or 1 key in common. Also note that the probability of any two blocks sharing a common key is $\frac{k}{r+1}$. Now let us concentrate on the number of common keys between N_1 and N_2. It is easy to see that in one extreme case, each of the z blocks in N_1 may have a key in common with each of the z blocks in N_2 and thus there could be totally z^2 common keys between N_1 and N_2. In the other extreme case, none of the z blocks in N_1 may have a key in common with any of the z blocks in N_2 and thus there could be 0 common keys between N_1 and N_2.

Let B_{ij} be a discrete random variable defined as follows:

$B_{ij} = 1$ if the ith block of N_1 shares a common key with the jth block of N_2
$\quad = 0$ otherwise.

Also note that $P(B_{ij} = 1) = \frac{k}{r+1}$ for all i, j where $0 \leq i, j \leq z$.

We define another random variable X denoting the number of common keys between N_1 and N_2. It is defined as $X = \displaystyle\sum_{i=0}^{z}\sum_{j=0}^{z} B_{ij}$.

Hence $X \sim Bin(z^2, \frac{k}{r+1})$.

4.2.2 Merging

We use the concept of merging blocks to form a sensor node. Initially we do not specify any merging strategy and consider that blocks will be merged randomly. In this direction we shall present a technical result. However, it is important to note that

Theorem 2 *Consider a (v, b, r, k) configuration (resulting from the $TD(k, r)$) with $b = r^2$. We merge z many randomly selected blocks to form a sensor node. Then*

1. *There will be $N = \lfloor \frac{b}{z} \rfloor$ many sensor nodes.*
2. *The probability that any two nodes share no common key is $(1-p_1)^{z^2}$, where $p_1 = \frac{k}{r+1}$.*
3. *The expected number of keys shared between two nodes is $z^2 p_1$.*
4. *Each node will contain M many distinct keys, where $zk - \binom{z}{2} \leq M \leq zk$. The average value of M is $\hat{M} = zk - \binom{z}{2}\frac{k}{r+1}$.*
5. *The expected number of links in the merged system is*
$$\hat{L} = \left(\binom{r^2}{2} - \left\lfloor \frac{r^2}{z} \right\rfloor \binom{z}{2}\right) \frac{k}{r+1} - (r^2 \bmod z)k.$$
6. *Each key will be present in Q many nodes, where $\lceil \frac{r}{z} \rceil \leq Q \leq r$. The average value of Q is*
$\hat{Q} = \frac{1}{kr}\left(\lfloor \frac{b}{z} \rfloor\right)\left(zk - \binom{z}{2}\frac{k}{r+1}\right)$.

Proof : The first item is easy to see.

Since the blocks are merged randomly, any two sensor nodes will share no common key if and only if none of the keys in z blocks constituting one sensor node are available in the z blocks constituting the other sensor node. Thus there are z^2 many cases where there are no common keys. As we have considered random distribution in merging z blocks to form a node, under reasonable assumption (corroborated by extensive simulation studies), all these z^2 events are independent. Note that p_1 is the probability that two blocks share a common key. Hence the proof of the second item.

The number of common keys between two nodes follows binomial distribution. The probability that two nodes share i many common keys is given by $\binom{z^2}{i} p_1^i (1-p_1)^{z^2-i}$, $0 \leq i \leq z^2$. Thus the mean of the distribution is $z^2 p_1$ which proves the third item.

For the fourth item, note that each block contains k many distinct keys. When z many blocks are merged, then there may be at most $\binom{z}{2}$ common keys among them. Thus the number of distinct keys M per sensor node will be in the range $zk - \binom{z}{2} \leq M \leq zk$. The average number of common keys between two nodes is $\frac{k}{r+1}$. So the average value of M is $zk - \binom{z}{2}\frac{k}{r+1}$.

Consider that z blocks are merged to form a node, i.e., given a $(v = rk, b = r^2, r, k)$ configuration (resulting from the $TD(k, r)$) we get $\lfloor \frac{r^2}{z} \rfloor$ many sensor nodes. The total number of links was $\binom{r^2}{2}\frac{k}{r+1}$ before the merging of blocks. For each of the nodes (a node is z many blocks merged together), $\binom{z}{2})\frac{k}{r+1}$ many links become intra-node links and totally, there will be a deduction of $\lfloor \frac{r^2}{z} \rfloor \binom{z}{2}\frac{k}{r+1}$ links (to account for the intra-node links) on an average. .

s	1	2	3	4	5	6	7	8	9	10
$fail(s)$	0.019591	0.038799	0.057631	0.076093	0.094194	0.111940	0.129338	0.146396	0.163119	0.179515
$Fail(s)$	0.020408	0.040408	0.060000	0.079184	0.097959	0.116327	0.134286	0.151837	0.168980	0.185714
Expt.	0.020406	0.040609	0.059986	0.078376	0.096536	0.117951	0.135109	0.151639	0.165508	0.184885

Table 4.1: Calculation of *fail(s)* and *Fail(s)*.

Further as we use $\lfloor \frac{r^2}{z} \rfloor$ many sensor nodes, we discard $(r^2 \bmod z)$ number of blocks, which contribute to $(r^2 \bmod z)k$ many links. There will be a deduction for this as well. Thus the expected number of links in the merged system is

$$\left(\binom{r^2}{2} - \left\lfloor \frac{r^2}{z} \right\rfloor \binom{z}{2}\right) \frac{k}{r+1} - (r^2 \bmod z)k.$$

This proves the fifth item.

Note that a key will be present in r many blocks. Thus a key may be exhausted as early as after being used in $\lceil \frac{r}{z} \rceil$ many sensor nodes. On the other hand a key may also be distributed to a maximum of r many different nodes. Hence the number of distinct nodes Q corresponding to each key is in the range $\lceil \frac{r}{z} \rceil \leq Q \leq r$. Now we try to find out the average value of Q, denoted by $\hat{Q}$. Total number of distinct keys in the merged design does not change and is also kr. Thus $\hat{Q} = \frac{N\hat{M}}{kr} = \frac{1}{kr} \left(\lfloor \frac{b}{z} \rfloor\right) \left(zk - \binom{z}{2} \frac{k}{r+1}\right)$. This proves the sixth item. ■

4.2.3 Calculating *fail(s)* when a block is considered as a node (no merging)

The expression *fail(s)*, the probability that a link become unusable if s many nodes are compromised, has been approximately calculated in the following way in [65]. Consider that there is a common secret key between the two nodes N_i, N_j. Let N_h be a compromised node. Now the key that N_i, N_j share is also shared by $r-2$ other nodes. The probability that N_h is one of those $r-2$ nodes is $\frac{r-2}{b-2}$. Thus the probability that compromise of s many nodes affect a link is approximately $1 - (1 - \frac{r-2}{b-2})^s$. Given the design $(v = 1470, b = 2401, r = 49, k = 30)$ and $s = 10$, $fail(10) \approx 0.17951$.

We calculate this approximate expression in a little different manner, which is better than the one described above. Given $b = r^2$ many nodes, the total number of links is $\binom{r^2}{2} \frac{k}{r+1}$. Now compromise of one node reveals k many keys. Each key is repeated in r many nodes, i.e., it is being used in $\binom{r}{2}$ many links. Thus if one key is revealed, it disturbs the following proportion of links:

$$\frac{\binom{r}{2}}{\binom{r^2}{2} \frac{k}{r+1}} = \frac{1}{kr}.$$

Now s many nodes contain $ks - \binom{s}{2}\frac{k}{r+1}$ many **distinct** keys on an average. This is because there are $\binom{s}{2}$ many pairs of nodes and a proportion of $\frac{k}{r+1}$ of them will share a common key. Thus, in our calculation, on an average

$$Fail(s) = \frac{ks - \binom{s}{2}\frac{k}{r+1}}{kr} = \frac{s}{r}(1 - \frac{s-1}{2(r+1)}).$$

Note that to distinguish the notation we use *Fail(s)* instead of *fail(s)* in [65]. Considering the design $(v = 1470, b = 2401, r = 49, k = 30)$, we tabulate the values of *fail(s)*, *Fail(s)* and experimental data (average of 100 runs for each s) regarding the proportion of links that cannot be used after compromise of s many nodes. The results look quite similar. However, it may be pointed out that our approximation is in better conformity with the experimental values (see Table 4.1) than that of [65], which looks a bit underestimated.

4.2.4 Calculation of *Fail(s)* when more than one blocks are merged

Let N_a and N_b be two given nodes. Define two events E and F as follows:

1. E: N_a and N_b are disconnected (i. e., N_a and N_b do not share a valid common key) after the failure of s nodes,
2. F: N_a and N_b were connected before the failure of those s nodes.

The sought for quantity is

$$Fail(s) = P(E|F) = \frac{P\left(E \bigcap F\right)}{P\left(F\right)}.$$

Let X be the random variable denoting the number of keys between N_a and N_b and following the proof of Theorem 2(2), we assume that X follows $B\left(z^2, \frac{k}{r+1}\right)$. Thus,

$$P\left(F\right) = P\left(X > 0\right) = 1 - P\left(X = 0\right) = 1 - \left(1 - \frac{k}{r+1}\right)^2.$$

Next define two sets of events:

1. E_{1i}: i number of keys (shared between N_a and N_b) are revealed consequent upon the failure of s nodes,
2. E_{2i} : i number of keys are shared between N_a and N_b.

Let $E_i = E_{1i} \bigcap E_{2i}$ for $i = 1, 2, \ldots, z^2$. So, $E_i \bigcap E_j = \emptyset$ for $0 \leq i \neq j \leq z^2$. As $E \bigcap F = \bigcup_{i=1}^{z^2} E_i$, we have $P\left(E \bigcap F\right) = P\left(\bigcup_{i=1}^{z^2} E_i\right)$

$= \sum_{i=1}^{z^2} P(E_i) = \sum_{i=1}^{z^2} P(E_{1i}|E_{2i})P(E_{2i})$ and also

$P\left(E_{2i}\right) = \binom{z^2}{i}\left(\frac{k}{r+1}\right)^i\left(1 - \frac{k}{r+1}\right)^{z^2-i}$.

Now we estimate $P(E_{1i}|E_{2i})$ by hypergeometric distribution. Consider the population (of keys) of size kr and γ number of defective items (the number of distinct keys revealed). We shall draw a sample of size i (without replacement) and we are interested in the event that all the items drawn are defective.

Note that γ is estimated by the average number of distinct keys revealed, i.e., $\gamma = szk\left(1 - \frac{sz-1}{2(r+1)}\right)$. So $P(E_{1i}|E_{2i}) = \frac{\binom{\gamma}{i}}{\binom{kr}{i}}$, $i = 1, 2, \ldots, z^2$.

Finally $P(E|F) = \frac{P(E \bigcap F)}{P(F)}$

$$= \frac{\sum_{i=1}^{z^2} \frac{\binom{\gamma}{i}}{\binom{kr}{i}} \binom{z^2}{i} \left(\frac{k}{r+1}\right)^i \left(1 - \frac{k}{r+1}\right)^{z^2-i}}{1 - \left(1 - \frac{k}{r+1}\right)^2}.$$

The estimate γ is a quadratic function of s and hence is not an increasing function (though in reality, it should be an increasing function of s $\forall s$). That is why *Fail(s)* increases with s as long as γ increases with s. Given $\gamma = szk\left(1 - \frac{sz-1}{2(r+1)}\right)$, it can be checked that γ is increasing for $s \leq \frac{2r+3}{2z}$. As we are generally interested in the scenarios where a small proportion of nodes are compromised, this constraint on the number of compromised nodes s is practical.

s	1	2	3	4	5	6	7	8	9	10
Fail(s) (Th 4)	0.020408	0.040408	0.060000	0.079184	0.097959	0.116327	0.134286	0.151837	0.168980	0.185714
Fail(s) (Th 3)	0.022167	0.044369	0.066527	0.088560	0.110385	0.131917	0.153069	0.173756	0.193891	0.213388
Expt.(random)	0.022987	0.045345	0.068904	0.090670	0.114853	0.135298	0.158633	0.181983	0.203342	0.222167
Expt.(heuristic)	0.022595	0.044146	0.067136	0.091243	0.112162	0.133693	0.157884	0.178895	0.200226	0.219273

Table 4.2: Calculation of *Fail(s)* in case of nodes which are merging of more than one blocks.

Based on the above discussion, we have the following theorem.

Theorem 3 *Consider a (v, b, r, k) configuration resulting from the $TD(k, r)$. A node is* *created by random merging of z many blocks. For $s \leq \frac{2r+3}{2z}$,*

$$Fail(s) \approx \frac{\sum_{i=1}^{z^2} \frac{\binom{\gamma}{i}}{\binom{kr}{i}} \binom{z^2}{i} \left(\frac{k}{r+1}\right)^i \left(1 - \frac{k}{r+1}\right)^{z^2-i}}{1 - \left(1 - \frac{k}{r+1}\right)^2},$$

where $\gamma = szk\left(1 - \frac{sz-1}{2(r+1)}\right)$.

It may be mentioned that while estimating $P(E_{1i}|E_{2i})$ by $\frac{\binom{\gamma}{i}}{\binom{kr}{i}}$, we are allowing a higher quantity in the denominator. The number of distinct keys revealed is under the restriction that the keys are distributed in s distinct blocks. However, the denominator is the expression for choosing i number of distinct keys from a collection of kr keys without any restriction. As a consequence, the resulting probability values will be under estimated, though the experimental results reveal that the difference is not significant at all (see Table 4.2).

Note that in Theorem 3, there is a restriction on s. Next we present another approximation of *Fail(s)* as follows where such a restriction is not there. However, the approximation of Theorem 4 is little further than that of Theorem 3 from the experimental results (see Table 4.2).

Theorem 4 *Consider a $(v = kr, b = r^2, r, k)$ configuration resulting from the $TD(k,r)$. A node is formed by merging $z > 1$ blocks. Then in terms of design parameters, $Fail(s) \approx$*

$$\frac{1}{1 - (1 - \frac{k}{r+1})^{z^2}} \sum_{i=1}^{z^2} \binom{z^2}{i} (\frac{k}{r+1})^i (1 - \frac{k}{r+1})^{z^2-i} \pi^i,$$

where, $\pi = szk(1 - \frac{sz-1}{2(r+1)})\frac{\hat{Q}(\hat{Q}-1)}{2\hat{L}}$.

Proof : Compromise of one node reveals $\hat{M}$ many keys on an average. Thus there will be $s\hat{M}$ many keys. Further, between any two nodes, $z^2\frac{k}{r+1}$ keys are common on an average. Thus we need to subtract $\binom{s}{2}z^2\frac{k}{r+1}$ many keys from $s\hat{M}$ to get the number of distinct keys. Thus the number of distinct keys in s many merged nodes is $= s\hat{M} - \binom{s}{2}z^2\frac{k}{r+1} = s(zk - \binom{z}{2}\frac{k}{r+1}) - \binom{s}{2}z^2\frac{k}{r+1} = szk(1 - \frac{sz-1}{2(r+1)})$.

We have $N = \lfloor \frac{b}{z} \rfloor$ many sensor nodes and $\hat{L} = (\binom{r^2}{2} - \lfloor \frac{r^2}{z} \rfloor \binom{z}{2})\frac{k}{r+1} - (r^2 \bmod z)k$ many average number of total links. Each key is repeated in $\hat{Q}$ many nodes on an average, i.e., it is being used in $\frac{\hat{Q}(\hat{Q}-1)}{2}$ many links. Thus if one key is revealed that disturbs $\frac{\hat{Q}(\hat{Q}-1)}{2\hat{L}}$ many links on an average. Hence compromise of 1 key disturbs $\frac{\frac{Q(Q-1)}{2}}{\hat{L}}$ proportion of links. Hence, compromise of s nodes disturbs $\pi = szk(1 - \frac{sz-1}{2(r+1)})\frac{\hat{Q}(\hat{Q}-1)}{2\hat{L}}$ proportion of links on an average. Thus we can interpret π as the probability that one link is affected after compromise of s many merged nodes.

Now the probability that there are i many links between two nodes given at least one link exists between them is $\frac{1}{1-(1-\frac{k}{r+1})^{z^2}}\binom{z^2}{i}(\frac{k}{r+1})^i(1-\frac{k}{r+1})^{z^2-i}$. Further the probability that all those i links will be disturbed due to compromise of s nodes is π^i. Hence *Fail(s)*

$$= \frac{1}{1-(1-\frac{k}{r+1})^{z^2}} \sum_{i=1}^{z^2} \binom{z^2}{i} (\frac{k}{r+1})^i (1 - \frac{k}{r+1})^{z^2-i} \pi^i. \qquad \blacksquare$$

The following example illustrates our approximations vis-a-vis the experimental results. Consider a $(v = 101 \cdot 7, b = 101^2, r = 101, k = 7)$ configuration (resulting from the $TD(7, 101)$) and merging of $z = 4$ blocks to get a node. Thus there will be 2550 many nodes. In such a situation we present the proportion of links disturbed if s many $(1 \leq s \leq 10)$ nodes are compromised, i.e., this can also be seen as the probability that two nodes get disconnected which were connected earlier (by one or more links). In Table 4.2 we present the values that we get from Theorem 4, Theorem 3 and also experimental results which are the average of 100 runs.

4.2.5 Comparison with the Lee-Stinson Approach

Comparison	our Section 4.2.2	our Section 4.3	[65]
Number of nodes	2550	2550	2401
Number of keys per node	≤ 28	28	30
Prob(two nodes don't share a common key)	0.320555	0.309916	0.4
Fail(s)	0.222167	0.219273	0.185714

Table 4.3: Comparison with an example presented in [65]

In the example presented in [65], the design $(v = 1470, b = 2401, r = 49, k = 30)$ has been used to get $N = 2401, M = 30, Q = 49, p_1 = 0.6, 1 - p_1 = 0.4$.

Now we consider the design $(v = 101 \cdot 7 = 707, b = 101^2 = 10201, r = 101, k = 7)$. Note that in this case $p_1 = \frac{k}{r+1} = \frac{7}{102}$. We take $z = 4$. Thus $N = \lfloor\frac{10201}{4}\rfloor = 2550$. Further the probability that two nodes will not have a common key is $(1 - \frac{7}{102})^{16} = 0.32061$. Note that this is considerably lesser (better) than the value 0.4 presented in [65] under a situation where the number of nodes is greater ($2550 > 2401$) and number of keys per node is lesser ($28 < 30$) in our case. Thus our strategy is clearly more efficient than that of [65] in this aspect. On the other hand, the *Fail(s)* value is worse in our case than what has been achieved in [65]. In Table 4.3, for our approaches, we present the experimental values which are average over 100 runs. For the time being let us concentrate on the comparison between our contribution in this section (Section 4.2.2) and the idea presented in [65]. In the next section (Section 4.3), we will present a better idea and the result of that is also included in Table 4.3 for brevity.

The comparison in Table 4.3 is only to highlight the performance of our design strategy with respect to what is described in [65] and that is why we present a design with average number of common keys between any two nodes ≤ 1. However, we will present a practical scenario in the next subsection where there are more number (≥ 5) of common keys (on an average) between any two nodes and consequently the design achieves much less *Fail(s)* values.

One more important thing to mention is that we consider the average case analysis for our strategy. The worst case situation will clearly be worse than the average case, but that is not of interest in this context as we will first try to get a merging configuration which is close to the average case. As this is done in preprocessing stage, we may go for more than one attempts for the configuration and it is clear that in a few experiments, we will surely get a configuration matching the average case result. On the other hand, it is very important to identify the best case as this will provide a solution better than the average case. However, this is open at this point of time.

The strength of our scheme is in the presence of several common keys between two nodes, which in fact makes it more resilient. Of course, this is at the cost of an obvious increase in number of keys in each node by a factor of z. The example presented in Subsection 4.2.5 and Subsection 4.2.6 illustrate this fact. In Subsection 4.2.5, we deliberately allowed a very low number of common keys (so that the node size is comparable to that of [65]) and hence the negative resiliency measure *Fail(s)* increased slightly. In what follows, we demonstrate that with an increase in the node capacity, the negative resiliency measure *Fail(s)* assumes a negligible value.

4.2.6 A Practical Design with More than one Key (on Average) Shared Between two Nodes

We start with the idea that a node can contain 128 keys and as we like to compare the scenario with [65], we will consider the number of sensor nodes ≥ 2401, as it has been used in the examples in [65].

Consider a $(v = rk, b = r^2, r = 101, k = 32)$ configuration resulting from the $TD(32, 101)$.

If one merges $z = 4$ blocks (chosen at random) to construct a node, the following scheme is obtained (refer to Theorem 2, 3).

1. There will be $\left\lfloor \frac{10201}{4} \right\rfloor = 2550$ sensor nodes.
2. The probability that two nodes do not share a common key is approximately $\left(1 - \frac{32}{102}\right)^{16} = 0.0024$.
3. Expected number of keys shared between two nodes $= \frac{16 \cdot 32}{102} \geq 5$.
4. Each node will contain on an average $\hat{M} = 4 \times 32 - \binom{4}{2} \frac{32}{102} \approx 126$ many distinct keys and at most 128 many keys.
5. $Fail(10) = 0.019153 \approx 2\%$ and $Fail(25) = 0.066704 \approx 7\%$.

This example clearly uses more keys ($\leq$ 128) per sensor node than the value 30 in the example of [65]. Note that directly from a (v, b, r, k) configuration (resulting from the $TD(k, r)$), it is not possible to have $k > r$. However, in a merged system that is always possible. Moreover, the average number of keys shared between any two nodes is ≈ 5. It is not easy to get a combinatorial design [97] to achieve such a goal directly. This shows the versatility of the design proposed by us.

4.3 A Heuristic: Merging Blocks attempting to minimize the number of intra node common keys

So far we have used the concept of merging blocks to form a sensor node without any constraints on how the blocks will be chosen to form a node. Now we add the constraint that the blocks that will be merged to form a node will not have any common key among themselves. For this we present the following heuristic. Before we present the heuristic, we shall need another definition.

Define a **move** as follows:

1. From the list of pairs of nodes sharing maximum number of common keys, select one pair of nodes randomly. Call them a and b.
2. From the list of pairs of nodes sharing no common key, select one pair of nodes randomly. Call them c and d.

3. Select one block each from a and b and remove them such that the removed blocks intersect each other and a and b are still connected upon their removal. Let the removed blocks be α and β respectively.
4. select one block each from c and d and remove them. Let the removed blocks be γ and δ respectively.
5. Put γ in a, δ in b, α in c and β in d.
6. Update the adjacency matrix and record the increase in connectivity.
7. Undo the above changes.

Heuristic :

```
1. $flag = true$; $count = 0$; all the blocks are marked as unused;
2. an array $node[\ldots]$ is available, where each element of the array can store $z$ many blocks;
3. while($flag$){
   (a) choose a random block, mark it as used and put it in $node[count]$;
   (b) for $(i = 1; i < z; i++)${
         i. search all the unused blocks in random fashion and put the first available one
            in $node[count]$ which has no common key with the existing blocks already in
            $node[count]$;
        ii. mark this block as used;
       iii. if such a block is not available then break the for loop and assign $flag = false$;
   (c) } (end for)
   (d) if $flag = true$ then $count = count + 1$;
4. } (end while)
5. report that $count$ many nodes are formed such that there is no intra node connectivity.
6. for rest of the $(r^2 - count \cdot z)$ many blocks, merge $z$ blocks randomly to form a node
   (they may have intra node connectivity) to get $(\lfloor \frac{r^2}{z} \rfloor - count)$ many extra nodes. This
   constitutes the initial configuration.
7. Calculate the adjacency matrix.
```

8. Make 1000 **moves** in succession, choose the one that gives rise to the maximum increase in connectivity and make the corresponding change in the configuration. Call it an **iteration**.

9. Perform 1000 such iterations.

In one of the earlier experiments, we have considered only up to step 5 of Heuristic 4.3 [18]. It is very clear that given (v, b, r, k) configuration (resulting from the $TD(k, r)$) with $b = r^2$, if one merges z many blocks to get each node then the maximum possible nodes that are available could be $N \leq \lfloor \frac{b}{z} \rfloor$. However, it is not guaranteed that given any configuration one can really achieve the upper bound $\lfloor \frac{b}{z} \rfloor$ with the constraint that the blocks constituting a node can not have any common key among themselves. Using Heuristic 4.3 up to step 5, one can use all the blocks in some cases, but sometimes it may not be possible also.

The following example illustrates the experimental results and we show that using this technique we get better (lower) *Fail(s)* value than Section 4.2.2 as evident from the last row of Table 4.2. Consider a $(v = 101 \cdot 7, b = 101^2, r = 101, k = 7)$ configuration (resulting from the $TD(7, 101)$) and merging of $z = 4$ blocks to get a node. Thus there will be 2550 many nodes. In such a situation we present the proportion of links disturbed if s many $(1 \leq s \leq 10)$ nodes are compromised, i.e., this can also be seen as the probability that two nodes get disconnected which were connected earlier (by one or more links).

4.3.1 Experimental Results with this Heuristic

Let us refer to Table 4.3 for the comparison. As usual, we consider the $(v = 101 \cdot 7 = 707, b = 101^2 = 10201, r = 101, k = 7)$ configuration (resulting from the $TD(7, 101)$) to attain a comparable design after merging. Note that in this case $p_1 = \frac{k}{r+1} = \frac{7}{102}$. We take $z = 4$. Thus $N = \lfloor \frac{10201}{4} \rfloor = 2550$. Considering the binomial distribution presented in Theorem 2(3), the theoretical probability that two nodes will not have a common key is $(1 - \frac{7}{102})^{16} = 0.32061$. Experimentally with 100 runs we find the average value as 0.309916 which is less (better) than the theoretically estimated value and also the experimental value 0.320555 as explained in Section 4.2.2 under the same experimental set up. Note that this is considerably lesser than the value 0.4 presented in [65]. The average number of common keys between any two nodes is $z^2 p_1 = z^2 \frac{k}{r+1} = 16 \frac{7}{102} = 1.098039$. Experimentally with 100 runs we get it as 1.098362 on an average which is a higher (improved) value than the theoretical estimate and also the experimental value 1.098039 as given in Section 4.2.2 under the same experimental set up. Further note that the last row of the Table 4.2 provides better (lesser) *Fail(s)* values available from the heuristic than the random search.

s	1	2	3	4	5	6	7	8	9	10
Fail(s)	0.000724	0.001763	0.003050	0.004643	0.006612	0.008495	0.011694	0.013063	0.017261	0.019339

Table 4.4: Experimental *Fail(s)* values.

4.3.2 More Keys Shared Between Two Nodes

As in Subsection 4.2.6, consider a $(v = rk, b = r^2, r = 101, k = 32)$ configuration (resulting from the $TD(32, 101)$). If one merges $z = 4$ blocks to construct a node according to Heuristic 4.3, the following scheme is obtained.

1. There are $\lfloor \frac{10201}{4} \rfloor = 2550$ many sensor nodes.
2. The probability that two nodes do not share a common key is approximately $\left(1 - \frac{32}{102}\right)^{16} = 0.002421$. The experimental value on an average is 0.002094 with 100 runs which is lesser (better) than the theoretically estimated value.
3. Expected number of keys shared between two nodes $= \frac{16 \cdot 32}{102} \geq 5.019608$. The experimental value with 100 runs is 5.021080 on an average, little better than the theoretically estimated value.

In Table 4.4 we present the experimental value for *Fail(s)*, where we take the average over 100 runs for each s.

4.4 Key Exchange

In this section, we present the key exchange protocol between any two nodes. First we present the key exchange protocol (as given in [65]) between two blocks N_a, N_b having identifiers (a_1, a_2) and (b_1, b_2) respectively. We take a $(v = kr, b = r^2, r, k)$ configuration (resulting from the $TD(k, r)$). Thus the identifier of a block is a tuple (a_1, a_2) where $a_1, a_2 \in \{0, \ldots, r-1\}$ and the identifier of a key is a tuple (k_1, k_2) where $k_1 \in \{0, \ldots, k-1\}, k_2 \in \{0, \ldots, r-1\}$.

Algorithm 1

1. *Consider two blocks N_a, N_b having identifiers (a_1, a_2) and (b_1, b_2) respectively.*
2. *if $a_1 = b_1$ (and hence $a_2 \neq b_2$), then N_a and N_b do not share a common key.*
3. *else $x = (b_2 - a_2)(a_1 - b_1)^{-1} \bmod r$. If $0 \leq x \leq k-1$, then N_a and N_b share the common key having identifier $(x, a_1x + a_2)$. If $x \geq k$, then N_a and N_b do not share a common key.*

They can independently decide whether they share a common key in $O(\log_2^2 r)$ time as inverse calculation is used [95, Chapter 5].

In the proposed system, a node comprises of z number of blocks. Since each block has an identifier (which is an ordered pair $(x, y) \in Z_r \times Z_r$), a node in the merged system has z number of such identifiers which is maintained in a list.

Algorithm 2

```
1. for the tth block in the node $N_a$, $t = 1, \ldots, z$
    (a) send the identifier corresponding to the tth block to the other node $N_b$;
    (b) receive an identifier corresponding to a block in $N_b$;
    (c) compare the received identifier from $N_b$ with each of the $z$ identifiers in it (i.e., $N_a$) using Algorithm 1;
    (d) if a shared key is discovered acknowledge $N_b$ and terminate;
    (e) if an acknowledgment is received from $N_b$ that a shared key is discovered then terminate;
2. report that there is no shared key;
```

Since N_a and N_b participate in the protocol at the same time, the above algorithm is executed by N_a and N_b in parallel. There will be $O(z)$ amount of communications between N_a and N_b for identifier exchange and the decision whether they share a common key. At each node at most z^2 many inverse calculations are done (each identifier of the other node with each identifier of the node), which gives $O(z^2 \log_2^2 r)$ time complexity.

4.5 Conclusion and Future Research

In this chapter, we first present a randomized block merging strategy in proposing a key pre-distribution scheme for secure communication among the sensor nodes. Our idea presents a departure from the usual combinatorial design in the sense that the designs are readily available according to user requirements. Our merging strategy results into schemes that are not directly available from combinatorial designs.

Our main target is to get more than one common keys among any pair of nodes that provides a robust network in terms of security under adversarial conditions where some nodes may get compromised. We present detailed mathematical analysis in presenting our results with supporting experimental data.

Next we present a heuristic improvement of the basic randomized block merging strategy. In this case we present a strategy for merging blocks in a (v, b, r, k) configuration (resulting from the $TD(k, r)$) in such a manner that the blocks constituting a node will not share any common key among themselves. This provides better parameters than our basic design.

It will be interesting to regularize the key pre-distribution after random merging. In the strategy presented in this chapter, the number of common keys between any two nodes follow binomial distribution. Thus, there is a probability (though very low) that there may be no common key between two nodes (for the time being, to get around this difficulty, two nodes can always communicate via an intermediate node with almost certainty). It looks promising to apply more sophisticated heuristic re-arrangement of blocks among the nodes available after the merging so that the number of common keys between any two nodes becomes more or less constant and always ≥ 1.

Chapter 5

Clique Size in Sensor Networks with Key Pre-distribution Based on Transversal Design

5.1 Introduction

A sensor node is a small, inexpensive and resource constrained device that operates in RF (radio frequency) range. It has limitations in different aspects such as communication, computation, power and storage. A DSN (distributed sensor network) is an ad-hoc network consisting of sensor nodes. The sensor nodes are often deployed in an uncontrolled environment where they are expected to operate unattended. In many situations, the DSN is also very large. In either case, though one might try to control the density of deployment, the only deployment option is to randomly scatter the nodes to cover the target area. The consequence is that the location or topology is not available prior to deployment.

Given the various limitations, the security of the DSN hinges on efficient key distribution techniques. Even with the present day technology, public key crypto-systems are considered too computation intensive for DSNs and typically a DSN establishes a secure network by the use of pre-distributed keys. The following four metrics are often used to evaluate key pre-distribution solutions.

1. **Scalability:** The distribution must allow post-deployment increase in the size of network.
2. **Efficiency:**
   (a) **Storage:** Amount of memory required to store the keys.

(b) **Computation:** Number of cycles needed for key establishment.

(c) **Communication:** Number of messages exchanged during the key generation/agreement phase.

3. **Key Connectivity (probability of key share):** The probability that two nodes share one/more keys should be high.

4. **Resilience:** Even if a number of nodes are compromised, i.e., the keys contained therein are revealed, the complete network should not fail, i.e., only a part of the network should be affected.

One of the challenges in DSNs is to find efficient algorithms to distribute the keys to sensor nodes before they are deployed. The solutions may be categorized as follows:

1. **Probabilistic:** The keys are randomly chosen from a given collection of keys and distributed to the sensor nodes.

2. **Deterministic:** The key distribution is obtained as the output of some deterministic algorithm.

3. **Hybrid:** A combination of deterministic and probabilistic approaches.

A trivial (and obvious) deterministic solution to the problem is to put the same key in all the nodes. However, the moment a single node is compromised, the network fails. To guard against such a possibility, one can think of using distinct keys for all possible pair of nodes in the DSN. The very good resilience notwithstanding, the solution is not viable for even networks of moderate size due to the limited storage capacity of the nodes. If there are N nodes, then there will be $\binom{N}{2}$ keys in total and each node must have $N-1$ many keys. It is not possible to accommodate $N-1$ many keys in a node given the current memory capacity of sensor hardware when N is moderately large, say ≥ 500.

Let us now briefly refer a few state of the art key pre-distribution schemes. The well known Blom's scheme [4] has been extended in recent works for key pre-distribution in wireless sensor networks [31, 64]. The problem with these kinds of schemes is the use of several multiplication operations (as example see [31, Section 5.2]) for key exchange. The randomized key pre-distribution is another strategy in this area [34]. However, the main motivation is to maintain a connectivity (possibly with several hops) in the network. As an example [34, Section 3.2], a sensor network with 10000 nodes has been considered and to maintain the connectivity, it has been calculated that it is enough if one node can communicate with only 20 other nodes. Note that the communication between any two nodes may require a large number of hops. However, only the connectivity criterion (with too many hops) may not suffice in an adversarial condition. Further in such a scenario, the key agreement between two nodes requires exchange of the key indices. The use of combinatorial and probabilistic design (also a combination of both – termed as hybrid design) in the context of key distribution has been proposed in [12]. In this case also, the main motivation was to have low number of common keys.

In [65] transversal design (see Subsection 3.4 for more details) has been used where the blocks correspond to the sensor nodes. In our recent works [17, 18], we have proposed to start from a combinatorial design and then apply a probabilistic extension in the form of random merging of blocks to form the sensor nodes and in this case there is good flexibility in adjusting the number of common keys between any two nodes. In our earlier works [17, 18], we dealt with the cases of (i) unconstrained random merging of blocks and (ii) random merging of blocks with the restriction that the nodes are composed of disjoint blocks (do not share common keys among themselves). The computation to find out a shared key under this framework is of very low time complexity [17, 18, 65], which basically requires calculation of the inverse of an element in a finite field. That is the reason this kind of design becomes popular for application in key pre-distribution.

In the domain of distributed computing, the nodes forming a complete graph is an "ideal situation." As mentioned earlier, one gains a lot in terms of resilience. Moreover, the communication complexity decreases because fewer messages are exchanged between the nodes in order to generate/agree upon a key. In such a scenario, there is no question of "multi-hop" paths and since there is a unique key shared between any two nodes, the computational complexity decreases as well.

Thus, in a DSN, it is important to study the subset of nodes (clique, in graph theoretic terminology) that are connected to each other. By connectivity of two nodes we mean that the nodes share one or more common secret key(s) for secure communication. In this chapter, we study the basic combinatorial designs [65] and their extensions in terms of merging [17, 18] to estimate the cliques of maximum size. We show that if one uses a $(v = rk, b = r^2, r, k)$ configuration (resulting from the $TD(k, r)$), where each block corresponds to a node [65], then the maximum clique size is $r = \sqrt{b}$. We also study the extension of the basic design where a few blocks are merged to get a node [17, 18] and show that in such a strategy the clique size becomes considerably larger than what is available in the basic design [65].

5.2 Analysis of Clique Sizes

First we study the maximum clique size where the $(v = rk, b = r^2, r, k)$ configuration (resulting from the $TD(k, r)$) is used and each block in the design corresponds to a sensor node, which is the idea proposed in [65].

Theorem 5 *Consider a DSN with b many nodes constructed from a $(v = rk, b = r^2, r, k)$ configuration (resulting from the $TD(k, r)$). The maximum clique in this case is of size r.*

Proof : First we prove that there is a clique of size r. It is known that a key is repeated in r many different blocks. Fix a key. Thus, there are r many distinct blocks which are connected to each other by the fixed key. Hence there is a clique of size r.

Now we prove that there is no clique of size $r + 1$, because that will rule out the possibility of cliques of larger size. Let there be a clique of size $r + 1$. Note that the (v, b, r, k) configuration results from $TD(k, r)$ (see Subsection 3.4). In this case each block is identified by two indices (i, j), $0 \leq i, j \leq r - 1$. Further two blocks having same value of i (i.e., in the same row) can't have a common key. The moment one chooses $r + 1$ blocks, at least two of the blocks must be from the same row (by pigeon hole principle as there are at most r many rows) and are disjoint, which is a contradiction to the basic assumption of a clique having size $r + 1$. ∎

It should be observed that the clique size r is exactly the square-root of the number of nodes $b = r^2$. Note that in such a case two nodes/blocks either share a common secret key or not. Consider the graph with b^2 many nodes/vertices where each block corresponds to a node. Now two vertices are connected by an edge if they share a common secret key, otherwise they are not connected. Now a block contains k many distinct keys. For each key, a clique of size r is formed. Thus a vertex/node in this graph participates in k many cliques each of size exactly r.

Given two keys, which never occur together in the same block, will form cliques which are completely disjoint. On the other hand, two keys may occur together at most in a single block. In such a case, the two different cliques generated by them can intersect on a single node/vertex corresponding to the block that contains both the keys.

5.2.1 The Merging Approach

To overcome certain restrictions in the strategy provided in [65] (explained in the previous subsection), we have provided a strategy to merge certain blocks to construct a sensor node in chapter 4. The basic idea is to start from a $(v = rk, b = r^2, r, k)$ configuration (resulting from the $TD(k, r)$). Then we merge z many blocks to form a single sensor node. Thus the maximum number of sensor nodes available in such a strategy is $\lfloor \frac{r^2}{z} \rfloor$. We have studied a random merging strategy in section 4.2.2, where randomly chosen z many blocks are merged to get a sensor node. In such a scenario, we found that the number of common keys among any two nodes approximately follows the binomial distribution $\mathcal{B}(z^2, \frac{k}{r+1}))$. The expected number of common secret keys among any two nodes is $\frac{z^2 k}{r+1}$ (see theorem 2). It has been shown that this strategy provides favourable results compared to [65]. Note that in section

4.2.2, the blocks are merged randomly. So it may happen that the blocks being merged may have common secret key(s) among themselves. This is actually a loss, since we really do not need a common key among the blocks that are merged to get a single node. Hence, in section 4.3, we improved the strategy such that only disjoint blocks are merged to construct node. This provides little better parameters compared to what is given in section 4.2.2. In this chapter, we will show that our strategy given in section 4.2.2 and section 4.3 provides better clique size than that of the design presented in [65].

Now we concentrate on the cliques where blocks are merged to get a node as described in sections 4.2.2 and 4.3. It is worth mentioning that the number of blocks is $\left\lfloor \frac{r^2}{z} \right\rfloor$ in this case. From [17, Theorem 1], each key will be present in Q many nodes, where average value of Q is $\hat{Q} = \frac{1}{kr}\left(\left\lfloor \frac{b}{z} \right\rfloor\right)\left(zk - \binom{z}{2}\frac{k}{r+1}\right) \approx r$. So cliques of size $\approx r$ are available in the design where merging strategy is employed.

We like to highlight that the value of z is much less than r (as example, $r = 101, z = 4$) though it is not a serious restriction in the proof of our results in the following discussion.

Thus we like to point out the following improvement in the merging strategy over the basic technique.

1. In the basic design, there are r^2 many nodes (each block corresponds to a sensor node) and the maximum clique size is r.

2. Using the merging strategy, there are $\left\lfloor \frac{r^2}{z} \right\rfloor$ many nodes (z many blocks are merged to get a sensor node) and the maximum clique size is $\approx r$. Thus there is an improvement by a factor of $\sqrt{z}$ in the size of clique.

Let us present some examples to illustrate the comparison. The design ($v = 1470, b = 2401, r = 49, k = 30$) has been used as an example in [65]. Hence there are 2401 nodes and the largest clique size is 49. Now consider a ($v = 101 \cdot 7, b = 101^2, r = 101, k = 7$) configuration (resulting from the $TD(7, 101)$) and merging of $z = 4$ blocks to get a node. Thus there will be 2550 (we take this value as it is comparable to 2401) many nodes. We have cliques of size ≈ 101 on an average, which shows the improvement.

Next we provide a more improved result by increasing the clique size beyond r. We present a merging strategy where one can get a clique of size $r + z - 1 \geq r$ for $z \geq 1$. The result is as follows.

Theorem 6 *Consider a (v, b, r, k) configuration (resulting from the $TD(k, r)$) with $b = r^2$. We merge z many blocks to form each node in achieving a DSN having $N = \lfloor \frac{b}{z} \rfloor$ many sensor nodes. Then there exists an initial merging strategy which will always provide a clique of size $r + z - 1$.*

Proof : Let's denote the nodes by $\nu_1, \nu_2, \ldots$. Initially choose the first column of the $TD(k, r)$ and place the r blocks (indexed by $(i, 0)$ for $0 \leq i \leq r-1$) successively to fill up the first slot (out of the z slots) of the first r nodes $\nu_1, \nu_2, \ldots, \nu_r$. That will obviously yield a clique of size r as any two blocks in a specific column always share a common key.

The rest of the available blocks will always be traversed in column-wise manner. That is the next available block is now the one indexed by $(0, 1)$. Let us refer to the next available block by (i, j) for the rest of the present discussion. Once a block is used, we apply the update function on its index to get the next available node. Update (i, j) to $((i+1) \bmod r, j+\delta)$, where $\delta = 0$, if $i < r-1$ and $\delta = 1$ when $i = r-1$.

We go on adding new nodes for $t = 1$ to $z-1$ to generate a clique of size $r+z-1$ at the end.

To add a new node ν_{r+t}, proceed as follows. Choose the first available block (i, j) and put it in ν_{r+t}. Place the next available blocks in $\nu_1, \nu_2, \ldots, \nu_k$ as long as $i \leq r-1$. After using the last element of current column, the update function provides the first block of the next column. In that case, we add this new block $(0, j)$ to the node ν_{r+t}. Then again the next available blocks are put into the nodes $\nu_{k+1}, \nu_{k+2}, \ldots,$ in the similar manner. Once the blocks in that column gets exhausted, we again add the first block of the next column to ν_{r+t} and the following blocks to the nodes as long as we reach ν_{r+t-1}. Thus it is clear that all the nodes $\nu_1, \ldots, \nu_{r+t-1}$ are connected to ν_{r+t} increasing the size of the clique by 1.

In this strategy, the value of t is bounded above by $z-1$ as otherwise the number of blocks in a node will increase beyond z . The remaining blocks will be arranged randomly to have z blocks in each node to get $\left\lfloor \frac{r^2}{z} \right\rfloor$ many nodes in completing the merging strategy. ■

Now we present an example corresponding to the strategy presented in Theorem 6.

Example 3 *Consider the $TD(k, r = 5)$. Let $z = 2$. Consider the 5^2 blocks of the TD arranged in the form of a 5×5 matrix. If we adopt the strategy outlined in the proof of Theorem 6, initially, the following clique is obtained: $\nu_1 \rightarrow \{(0,0)\}$, $\nu_2 \rightarrow \{(1,0)\}$, $\nu_3 \rightarrow \{(2,0)\}$, $\nu_4 \rightarrow \{(3,0)\}$, $\nu_5 \rightarrow \{(4,0)\}$. Next $(0,1)$ is put in the new node ν_6 and then $(1,1)$ is added to ν_1, $(2,1)$ is added to ν_2, $(3,1)$ is added to ν_3, $(4,1)$ is added to ν_4. As the second column gets exhausted, $(0,2)$ is added to the new node ν_6 and then $(1,2)$ is added to ν_5. Thus we get, $\nu_1 \rightarrow \{(0,0),(1,1)\}$, $\nu_2 \rightarrow \{(1,0),(2,1)\}$, $\nu_3 \rightarrow \{(2,0),(3,1)\}$, $\nu_4 \rightarrow \{(3,0),(4,1)\}$, $\nu_5 \rightarrow \{(4,0),(1,2)\}$, $\nu_6 \rightarrow \{(0,1),(0,2)\}$ and they form a clique of size 6.*

Next we observe that the clique size we present in Theorem 6 is not the maximum achievable one. One can indeed find a different merging strategy that provides a clique of larger size. Here is an example.

Example 4 *Taking a different arrangement compared to Example 3, we get a clique of* *size* 7 *as follows:* $\nu_1 \to \{(0,0),(2,1)\}$, $\nu_2 \to \{(1,0),(3,1)\}$, $\nu_3 \to \{(2,0),(4,1)\}$, $\nu_4 \to \{(3,0),(0,2)\}$, $\nu_5 \to \{(4,0),(1,2)\}$, $\nu_6 \to \{(0,1),(2,2)\}$, $\nu_7 \to \{(1,1),(3,2)\}$.

Thus it will be interesting to devise a merging strategy which will provide the largest clique size when the (v,b,r,k) configuration (resulting from the $TD(k,r)$) and z are fixed.

Theorem 7 *Consider a (v,b,r,k) configuration (resulting from the $TD(k,r)$) with $b = r^2$. We merge z many blocks to form each node in achieving a DSN having $N = \lfloor \frac{b}{z} \rfloor$ many sensor nodes. Then there exists an initial merging strategy which will always provide a clique of size $2r - \left(\lceil \frac{r}{z} \rceil\right)$.*

Proof : Let the nodes forming the clique be $\nu_1, \nu_2, \cdots, \nu_{r+t}$. Each node has z number of empty slots. These slots are to be filled up by the properly chosen blocks. The TD may be considered as a $r \times r$ matrix filled with the values 0 to r^2-1. The value at the (i,j)th position of the matrix is $i \cdot r + j$ and that entry is used as an identification of the corresponding block in the TD. We try to form a clique with $r+t$ nodes, $t < r$. The value of t will be clearer as we go through the proof.

A column in the matrix (corresponding to the TD) is chosen first and the r blocks are placed one by one in r blank nodes, viz., $\nu_1, \nu_2, \cdots, \nu_r$. As all the blocks in the same column share the same secret key, the nodes $\nu_1, \nu_2, \cdots, \nu_r$ form a clique.

Then another column is chosen and the blocks are placed in the next t nodes, one each. In other words, the blocks are put in the nodes $\nu_{r+1}, \nu_{r+2}, \cdots, \nu_{r+t}$ and they form a clique among themselves. The rest $r-t$ blocks are added in the first $r-t$ nodes, viz., $\nu_1, \nu_2, \cdots, \nu_{r-t}$ (in the second slot). Thus each of $\nu_1, \nu_2, \cdots, \nu_{r-t}$ gets connected to each of $\nu_{r+1}, \nu_{r+2}, \cdots, \nu_{r+t}$.

In a similar fashion, the third column is chosen and the blocks are placed in $\nu_{r+1}, \nu_{r+2}, \cdots$, ν_{r+t}, one each (in the second slot). The rest $r-t$ blocks are placed in $\nu_{r-t+1}, \nu_{r-t+2}, \cdots$, $\nu_{r-t+r-t}$ (in the second slot). Thus each of $\nu_{r-t+1}, \nu_{r-t+2}, \cdots, \nu_{r-t+r-t}$ gets connected to each of $\nu_{r+1}, \nu_{r+2}, \cdots, \nu_{r+t}$.

We will continue the above process as long as the second slots of the first r nodes are eventually filled up. However, the continuation can be performed at most z many times as a node may accommodate at most z blocks. Each time $r-t$ new nodes are connected out of a target of r nodes. Thus, in order to complete the above process, we must have $r \leq z(r-t)$, $t \leq r - \lceil \frac{r}{z} \rceil$. ■

Example 5 *The technique in Theorem 7 outputs cliques of size $r+t$ where $t = r - (\lceil \frac{r}{z} \rceil)$. Let us consider $r = 5$ and $z = 2$ as in the previous examples. This technique outputs a clique of size $5+t$ where $t = 5 - (\lceil \frac{5}{2} \rceil) = 2$, i.e., we get a clique of size 7.*

The technique constructs the clique as follows:

$\nu_1 \rightarrow \{(0,0),(3,1)\}$, $\nu_2 \rightarrow \{(1,0),(4,1)\}$, $\nu_3 \rightarrow \{(2,0),(3,2)\}$,
$\nu_4 \rightarrow \{(3,0),(4,2)\}$, $\nu_5 \rightarrow \{(0,1),(0,2)\}$, $\nu_6 \rightarrow \{(1,1),(1,2)\}$,
$\nu_7 \rightarrow \{(2,1),(2,2)\}$,

Note that in the basic (v, b, r, k) configuration (resulting from the $TD(k, r)$) or after our merging strategy, the size of cliques are not dependent on the number of keys in each block/node. It is clear that the connectivity of the DSN increases with the increasing number of keys in each node. However, increasing the number of keys is constrained by the limited memory capacity of a sensor node. It is a nice property that the clique size does not increase with number of keys in each node (using our strategy) as otherwise one may be tempted to obtain cliques of larger sizes by increasing the number of keys in each node (i.e., by increasing the edges in the graph).

5.2.2 Configurations Having Complete Block Graphs: Projective Planes

Since we are talking about cliques, we should also revisit the designs where the entire DSN forms a clique. In [65, Theorem 11, 12], it has been pointed out that the block graph of a set system is a complete graph if and only if the set system is the dual design of a BIBD and in particular, there exists a key pre-distribution scheme for a DSN having $q^2 + q + 1$ nodes, in which every node receives exactly $q + 1$ keys and in which any two nodes share exactly one key. It is also stated that such designs are not recommendable as a key pre-distribution scheme in large DSNs because of storage limitation in each sensor node. We like to point out that even if the storage space is not a limitation, then also this scheme is not suitable. The reason is as follows.

In this design any two nodes share a common key. However, for better resiliency one may like to have more common keys among any two nodes (this is one important motivation for our merging strategy [17, 18]). Even if one maintains multiples keys against each identifier, the projective planes does not help because compromise of a single node results in discarding the identifiers contained in each node (block) and all the corresponding keys for each identifier also get discarded. Thus the resiliency measure *fail(s)*, (the probability that a given link is affected due to the compromise of s number of randomly chosen nodes) does not improve (i.e., does not reduce).

5.3 Conclusion and Future Research

In this chapter, we consider the DSNs where the key pre-distribution mechanism evolves from combinatorial design. Such schemes provide the advantage of very low complexity key exchange facility (only inverse calculation in finite fields). In terms of distributed computing and communication among the sensor nodes, it is important to study the subset of nodes that are securely connected to each other (clique). In this chapter, we have studied that in details. We studied the cliques corresponding to the (v, b, r, k) configuration (resulting from the $TD(k, r)$) where each block corresponds to a node. Further we study the scenario when more than one blocks are merged to generate a node. We show that the clique size gets improved in such a scenario. An interesting future work in this area is to implement a merging strategy such that one can get cliques of maximum size after the merging.

Chapter 6

Combinatorial Structures for Design of Wireless Sensor Networks

6.1 Introduction

Combinatorial designs are very effective tools for managing keys in an infrastructure where power and memory are two major constraints. None of the present day wireless technologies takes advantage of combinatorial designs. In this chapter, we propose a general framework using combinatorial designs which will enable the participating devices to communicate securely among themselves with little memory and power overhead. The scheme proposed caters for different kinds of user requirements and allows the designer to choose different combinatorial designs for different parts or levels of the network. A few examples of WLAN technologies are IEEE 802.11a/b/e/g/h/i, HiperLAN/2, HomeRF etc. and on the other hand, Bluetooth, ZigBee, UWB etc. are examples of WPAN technologies.

Very recently it is reported that two researchers have been successful in cracking the Bluetooth PIN [85]. The other wireless LAN technology protocol 802.11x also suffers from several security loopholes: insertion attacks, interception and monitoring wireless traffic, misconfiguration, jamming and client to client attacks are a few of the important ones. In the following, we shall introduce the desiderata of wireless technologies.

6.1.1 Wireless Technologies: How the Properties of Radio Waves Affect Networking Capabilities

An ideal radio wave for wireless technologies should have high speed, travel far distances and consume little energy. Had such radio waves existed, it would have been possible for us to transfer information very rapidly at any distance using little battery power. Unfortunately, real radio waves do not behave like that. The high speed and long range of a radio wave demands more energy. That is why the designers of the wireless technologies try to optimise certain parameters under a given condition. As a direct consequence, we find wireless area networks of different orders (e.g., personal, local, metropolitan, global, etc.) and each of them is suitable for a particular application or usage.

As an example, in wireless local area network (WLAN), the power consumption is less important compared to range/speed whereas the design of a wireless personal area network (WPAN) demands low power in preference to high speed or long range.

6.1.2 Our Proposal: An Uncharted Territory

However, an unexplored area in the security of wireless technologies is the use of combinatorial designs. Our proposal is an endeavour to propose security solutions in a wireless network using combinatorial designs. The method is not restricted to smart homes only and may also find application in Hierarchical Sensor Networks where the deployment of the sensor nodes may be made in a more or less controlled manner. One can think of other scenarios where a hierarchical structure may be deemed fit. As an extreme example, suppose the different countries of the world are divided into a few groups (possibly based on their geographical locations) and a multinational company operates globally, setting up branches in different countries. However, the management may decide to delegate the authority to each of the branch offices in an hierarchical structure. That structure may easily be translated to our model. In the following, we shall talk about two specific application areas viz., smart homes and sensor networks, though we have a common set of objectives in mind:

1. The entire communication in the network will take place securely.
2. The protocol will be as simple as possible.
3. The network will comprise of several logical parts. The network will be resilient to such an extent that the other parts will continue to function even if one/more parts of the network are compromised.

6.1.3 Smart Homes

A smart home or building is a home or building, usually a new one, that is equipped with special structured wiring to enable occupants to remotely control or program an array of automated home electronic devices by entering a single command. For example, a homeowner on vacation can use a Touchtone phone to arm a home security system, control temperature gauges, switch appliances on or off, control lighting, program a home theater or entertainment system and perform many other tasks. The field of home automation is expanding rapidly as electronic technologies converge. The home network encompasses communications, entertainment, security, convenience, and information systems.

Suppose we want to install the network in such a building. Naturally each of the rooms of the building forms a "logical part" of the network. The natural user requirement would be that the devices in one room should function independently of the devices of any other room. If one room has is cut off from the network, by accident, intention or malice, the other parts of the building should still be able to function unhindered. One can use same/different combinatorial designs to model the different parts of the network.

6.1.4 Key Pre-distribution in General: Our Proposal

One possible solution is to have a situation where every node is guaranteed to have a common key with every other node with which it needs to communicate. For a very large network, this is not possible, as explained earlier. We propose to divide the network into certain logical sub networks. Intra sub network nodes always share keys with each other. For each sub network, we earmark a particular node as a special node. Inter sub network communication takes place by the communication between the special nodes of the respective sub networks.

The issues at this point are as follows:

1. One has to have some control over the deployment of the nodes.
2. For the special nodes, the number of keys to be stored in each node will clearly increase. So one needs to decide the availability of storage space. In [65, Page 4], it has been commented that storing 150 keys in a sensor node may not be practical. On the other hand, in [31, Page 47], [64, Section 5.2], scenarios have been described with 200 keys. If one considers 4 Kbytes of memory space for storing keys in a sensor node, then choosing 128-bit key (16 byte), it is possible to accommodate 256 keys.

Thus the goal in this chapter is to present a scheme that aims at failsafe connectivity all-over the network. We differ from the existing works where it is considered that any two nodes will have either 0 or 1 common keys all over the network. Our motivation is to have a design strategy where the entire network is divided into a number of subnetworks. Any two nodes of a particular subnetwork share a common key. The special nodes of different subnetworks share more than one common keys. This is important from resiliency consideration in an adversarial framework since even if a certain subnetwork is compromised, the other parts of the network, i.e., the other subnetworks may function without any disturbance. Moreover, even if one or more special nodes are compromised, the other special nodes can still communicate among themselves. In other words, the connectivity of the network is not disturbed at all.

The rest of the chapter is organised as follows: We give examples using combinatorial designs for which any two nodes will have either 0 or 1 common keys. We then conclude with the future research proposals.

6.2 Key Pre-distribution in General: Our Approach

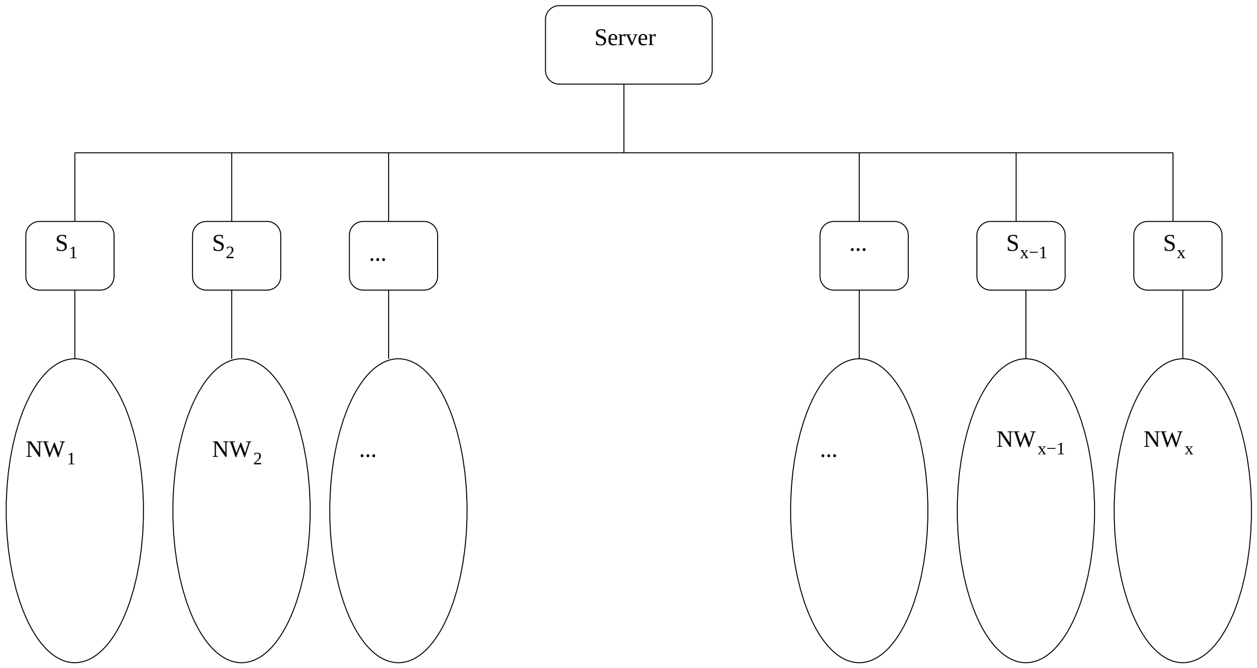

Figure 6.1: The Network

6.2.1 The Correspondence Between a Combinatorial Design and a Sensor Network

The blocks of the combinatorial design corresponds to a sensor node and the elements present in a block represent the keys present in a sensor node.

6.2.2 The Method

Lee and Stinson [65], have shown that using a transversal design, there is direct connectivity between two nodes in 60% of the cases. Overall, any two nodes can communicate either directly or through an intermediate node (i.e., a two-hop path) with almost certainty. For a large network, the compromise of even 10 nodes will render 18% of the nodes unusable.

Our approach is very different from the approach of [65]. In the diagram, we have shown a network with only two levels of hierarchy. There may be more levels depending on the user requirements. Our proposal is perfectly general and fits into networks of any size. The root of the hierarchy tree is assumed to be a central server, S. At the next level, x special nodes $S_1, S_2, \cdots, S_x$ are placed. The leaf level comprises of the subnetworks $NW_1, NW_2, \cdots, NW_x$.

One has the freedom to choose different combinatorial designs for different parts of the network. Again, that depends on the specific requirements of the user. For example, if the sub networks are required to form a totally connected network graph, one can choose projective planes. This may be applicable in case of a smart home. If the subnetworks are very large in size and total connectivity is not a requirement (i.e., if single/multi-hop connectivity is permissible), transversal designs might be a reasonable choice.

Let us assume that we are using only projective planes in all the parts of the network. We know that a projective plane of order n (n is a prime power) has $n^2 + n + 1$ number of blocks and each block contains $n + 1$ keys. If we use a projective plane of order n, we can accommodate a network of $n^2 + n + 1$ nodes with $n + 1$ keys per node.

Let us assume that $\max_i |NW_i| = \alpha$ (for $i = 1, 2, \cdots, x$), i.e., the subnetwork size is at most α, so that a projective plane of order $\geq \left\lceil \sqrt{\alpha - \frac{3}{4}} - \frac{1}{2} \right\rceil$ may be used to model the subnetwork.

In fact, we should choose the sub network size $n^2 + n$ instead of $n^2 + n + 1$ because we shall have to include the special node S_i(at the next highest level) corresponding to each sub network NW_i. The corresponding projective plane is of order $\left\lceil \sqrt{\alpha + \frac{1}{4}} - \frac{1}{2} \right\rceil$.

If we have x such sub networks, we have also x corresponding projective planes. They may or may not be of the same order depending on the same/different sizes of the various sub networks. One can use different projective planes for different sub networks NW_i simply by replacing α by NW_i in the above expression.

Note that each of the subnetworks NW_i including the special node S_i, i.e., $S_i \bigcup NW_i$ (for $i = 1, 2, \cdots, x$) forms a complete network graph. Since we are using a projective plane to distribute the keys in the underlying nodes, this property is guaranteed. In other words, any two nodes of $NW_i \bigcup S_i$ for $i = 1, 2, \cdots, x$ share a common key with each other.

Had we used a transversal design $TD(k, r)$ instead of a projective plane, every pair of nodes would not have been connected in a single hop. However, a constant fraction of the total number of pairs would have been connected (i.e., would have shared a common key). It is easy to see that the value of the fraction is $\frac{k}{r+1}$. Out of r^2 blocks of the $TD(k,r)$, a particular block shares keys with $kr - k = k(r-1)$ blocks. Excepting that particular block, there are $r^2 - 1$ blocks in the $TD(k,r)$. So the fraction is $\frac{k(r-1)}{r^2-1} = \frac{k}{r+1}$.

At the next stage, we would like to have several common keys between any two special nodes S_j and S_k. In order to achieve that, we may again choose projective planes. A projective plane of order $m \geq \left\lceil \sqrt{x + \frac{1}{4}} - \frac{1}{2} \right\rceil$ will suffice to connect all the S_is for $i = 1, 2, \cdots, x$ and also the root server S may be included as the $(x+1)$th node. Using multiple copies (say t copies) of the projective plane of order m and labelling them differently, we easily obtain t common keys between any two nodes of $\left(\bigcup_{i=1}^{x} S_i \right) \bigcup S$.

The special nodes/devices (which may be the cluster head in the case of a sensor network) should have more storage capacity in comparison with the other nodes in order to accommodate $t(m+1)$ keys.

6.2.3 An Example Using Projective Planes

Let us continue our discussion apropos of the previous network diagram, i.e., a network with only two levels of hierarchy. The root of the hierarchy tree is the central server, S. At the next level, $x = 18$ and special nodes $S_1, S_2, \cdots, S_{18}$ are placed.

The leaf level comprises of the subnetworks $NW_1, NW_2, \cdots, NW_{18}$. Let us use only projective planes all over the network.

Let us assume that $\max_i |NW_i| = 900$, i.e., the subnetwork size is at most 900, or, $\alpha = 900$.

The corresponding projective plane is of order $\geq \left\lceil \sqrt{900 + \frac{1}{4}} - \frac{1}{2} \right\rceil \geq 30$.

The next highest prime being 31, let us choose a projective plane of order 31.

Since we have 18 such sub networks, we have also 18 corresponding projective planes. They may or may not be of the same order depending on the same/different sizes of the various sub networks. One can use different projective planes for different sub networks NW_i simply by replacing 900 by $|NW_i|$ in the above expression.

Note that each of the subnetworks NW_i including the special node S_i, i.e., $S_i \bigcup NW_i$ forms a complete network graph. Since we are using a projective plane to distribute the keys in the underlying nodes, this property is guaranteed. In other words, any two nodes of $NW_i \bigcup S_i$ share a common key with each other (because the keys were distributed using a projective plane).

At the next stage, we would like to have several common keys between any two special nodes S_j and S_k. In order to achieve that, we may again choose projective planes. A projective plane of order $m \geq \left\lceil \sqrt{18 + \frac{1}{4}} - \frac{1}{2} \right\rceil \geq 4$ will suffice to connect all the S_is (for $i = 1, 2, \cdots, 18$) and also the root server S may be included as the 19th node. Let us choose $m = 4$. Using multiple copies (say 4 copies) of the projective plane of order m and labelling them differently, we readily have 4 common keys between any two nodes of $\left(\bigcup_{i=1}^{x} S_i \right) \bigcup S$.

The special nodes/devices (which may be the cluster head in the case of a sensor network) need more storage capacity than the other nodes in order to accommodate $4(4 + 1) = 20$ keys.

6.2.4 Another Example Using Projective Planes and Transversal Designs

Suppose we have a different kind of requirement. The sub networks are very large, say each subnetwork may be of size 2500 and hence multi-hop communication is permissible.

Again let us assume that the network has only two levels of hierarchy, the root of the hierarchy tree is the central server, S. At the next level, $x = 25$ and special nodes $S_1, S_2, \cdots, S_{25}$ are placed. The leaf level comprises of the subnetworks $NW_1, NW_2, \cdots, NW_{25}$.

At the sub network level, we do not have the requirement that any two nodes should be able to communicate directly. So we may use transversal designs at this level. However, since all the special nodes should be able to communicate directly among themselves and need an enhanced level of security by having multiple keys shared between any two nodes, we prefer to use projective planes at this level. In this example any two nodes at the sub network level can communicate in at most two hops.

Since the sub network may have 2500 nodes, we should choose a transversal design accordingly. We know that a $TD(k, r)$ has r^2 blocks. We also know that if r is prime and $2 \leq k \leq r$, then there exists a $TD(k, r)$ (see construction 1).

Since $\sqrt{2500} = 50$, we choose the next highest prime 53 as our r. Now we can choose k for our own convenience. We choose $k = 36$.

As mentioned earlier, the key sharing probability between any two nodes of the sub network $= \frac{k}{r+1} = \frac{36}{53+1} = 0.667$.

Note that each of the subnetworks NW_i including the special node S_i, i.e., $S_i \bigcup NW_i$ (for $i = 1, 2, \cdots, 25$) does not form a complete network graph. Since we are using a transversal design to distribute the keys in the underlying nodes, any two nodes of $NW_i \bigcup S_i$ share a common key with each other with probability 0.667.

At the next stage, we would like to have several common keys between any two special nodes S_j and S_k. In order to achieve that, we may again choose projective planes. A projective plane of order $m \geq \left\lceil \sqrt{25 + \frac{1}{4}} - \frac{1}{2} \right\rceil \geq 5$ will suffice to connect all the S_is for $i = 1, 2, \cdots, 25$ and also the root server S may be included as the 26th node. Let us choose $m = 5$. Using multiple copies (say 4 copies) of the projective plane of order m and labelling them differently, we readily have 4 common keys between any two nodes of $\left(\bigcup_{i=1}^{x} S_i \right) \bigcup S$.

The special nodes/devices (which may be the cluster head in the case of a sensor network) need more storage capacity than the other nodes in order to accommodate $4(5 + 1) = 24$ keys.

6.3 Conclusion and Future Research

We shall further investigate networks where "users" have differing resources and capacity requirements. One case involves a large network with large, mostly self-contained sub-networks. Another case involves networks which need more robustness at different levels of application. For example, at the second level of hierarchy (i.e., the level containing the special nodes), one may need to have different number of common keys shared between two given nodes. It will be an interesting combinatorial problem to find out a design having such a property. One may even look for better alternatives, in special circumstances, to the use of copies of projective planes at this level.

Chapter 7

Application of Transversal Design to Implement a Secure Grid of Distributed Wireless Sensor Network

7.1 Introduction

We have been considering random deployment of the nodes so far. Let us consider a slightly different area of application of the sensor network. Often there are practical applications of sensor networks to monitor an area that may be divided into a square grid. The points of intersection are accessible and the sensor devices may be placed at those points with reasonable precision. An example may be a factory floor or a warehouse. It will be interesting to setup a sensor network where the deployment pattern is in the form of a grid.

Transversal designs have an inherent structure so that the blocks may be considered to form a square matrix. Thus one can often choose a transversal design of appropriate size in order to cover a grid of sensor network and the blocks of the transversal design and the points of intersection of the physical grid are in 1-1 correspondence.

In this chapter, we assume that the keys contained in the sensor nodes are pre-distributed according to a transversal design and the sensor nodes are placed on a square grid.

Suppose the keys are pre-distributed in the sensor nodes in accordance with a transversal design $TD(k, r)$ and the transversal Design is exactly mapped to the deployment grid of size $r \times r$. The computation to find out a common key between two nodes is also of very low time complexity in this case. The connectivity and coverage of the network is examined when a number of nodes fail or get compromised. Given the number of compromised nodes, we study whether the network is still connected and covers the entire region under observation.

Such a configuration may help in considering different design choices like the number of keys per node, the sensing/RF radius of the sensor nodes and the robustness, $R(s)$, which is the probability that the network of sensor nodes is connected and covers the entire grid despite the failure of s nodes.

7.1.1 Related Works

In a random deployment model, the scaling laws for connectivity of nodes placed at random over a unit area is introduced in [38]. The same authors found the data carrying capacity of the network in [39]. In [37], the capacity of a network with n mobile nodes is studied. It has been shown that capacity of the network increases with node mobility and also end-to-end delay. In [41], the coverage problem when objects of various shape are dropped over an infinite plane is studied and in [72], the connectivity problems or percolation problems for a random node-replacement model is studied where the node placement points originate from a two-dimensional spatial Poisson process.

Bernoulli graph is another network model and is studied in [7]. It is fundamentally different from the radio-graph model as in a Bernoulli graph, two nodes may be separated by a large distance but still be connected whereas in radio-graph model, one can not travel an arbitrary distance in a single hop. In [57, 58], the authors discussed scaling results for reliable communications in gossip networks and developed a self-organising, peer-to-peer protocol that converges to reliably support a gossip network. They used Bernoulli graphs with directed arcs.

Quite a few problems of related nature have been studied in the context of sensor networks where the deployment pattern is known. The solutions usually discuss the connectivity, coverage, network lifetime, sleeping model of the sensor nodes. In [50], the authors proposed a sentry-based power management scheme. Here "sentry" means a live node. The dynamic sentry selection was not discussed in this work. However, [42] gives a scheme for sentry selection. In this scheme, a back-off interval inversely proportional to remaining energy is chosen by a node. Then the node informs the neighbouring nodes of its intention to become a sentry. This scheme manages energy consumption well, though it does not give any method to calculate the probability of a sentry selection if the expected lifetime of the network is known. A solution to this problem is offered in [59]. It discusses Random Independent Sleeping (RIS) where each node decides independently whether to sleep or remain alive for any given time period. This happens probabilistically with a fixed probability of remaining alive. RIS also does not compel a node that intends to become a "sentry" to communicate with its neighbours.

Apart from the goal of minimising power consumption, the principal goal is to ensure the coverage of the entire area under surveillance. If sufficient number of nodes are not deployed, it will not be possible to have enough nodes so that all the control points are covered by at least one node (known as 1-coverage). However, [42] can not provide 1-coverage in this case. Similarly, k-coverage is defined where each of the control points is covered by k nodes. Though [48, 100] present algorithms for deciding the sleeping strategy in this case, they do not talk about the minimum number of nodes to be deployed to ensure k-coverage.

Transversal design based key pre-distribution schemes for distributed wireless sensor network have been discussed in [16, 18, 21, 64], where it was assumed that the nodes are deployed at random and no a priori deployment knowledge is available. We noted that the inherent combinatorial structure of the transversal design enables to map it directly onto a two-dimensional grid of sensor networks. We analysed the network parameters under this assumption with the restriction that two nodes can communicate with each other only when they are within the RF communication range to each other and they share a common secret key.

7.2 Preliminaries

7.2.1 The Correspondence Between the Combinatorial Design and the Deployment Grid

Out of several parameters associated with a sensor node, two different parameters are "sensing radius" and "RF radius." The radius within which a sensor node is capable of gathering data is said to be the sensing radius of the sensor node. On the other hand, the radius within which a sensor node is capable of communicating to any other node using the radio frequency (RF) is said to be the RF radius of the sensor node. Though these two radii are unrelated, we may either assume them to be the same or choose the minimum of the two for the purpose of our analysis in this chapter and denote it by ρ, the "radius."

Consider a $(v = rk, b = r^2, r, k)$ configuration (resulting from the $TD(k, r)$) [97]. There are $b = r^2$ number of blocks which are actually considered as the sensor nodes, each containing k distinct keys. Each key occurs in r nodes. The parameter $v = rk$ gives the total number of distinct keys in the design. One should note that $bk = vr$ and $v - 1 > r(k - 1)$. The design provides 0 or 1 common key between any two nodes.

These r^2 number of nodes may be arranged in the form of a $r \times r$ matrix. Consider a square grid (mesh) with unit length u, i.e., a grid of size $(r-1)u \times (r-1)u$ so that the nodes are placed at all the r^2 number of points of intersection as shown in figure 7.1.

We can often make our life easier by putting $u = 1$ without any loss of generality.

In a transversal design $TD(k, r)$, there are r^2 blocks and the blocks are indexed as if

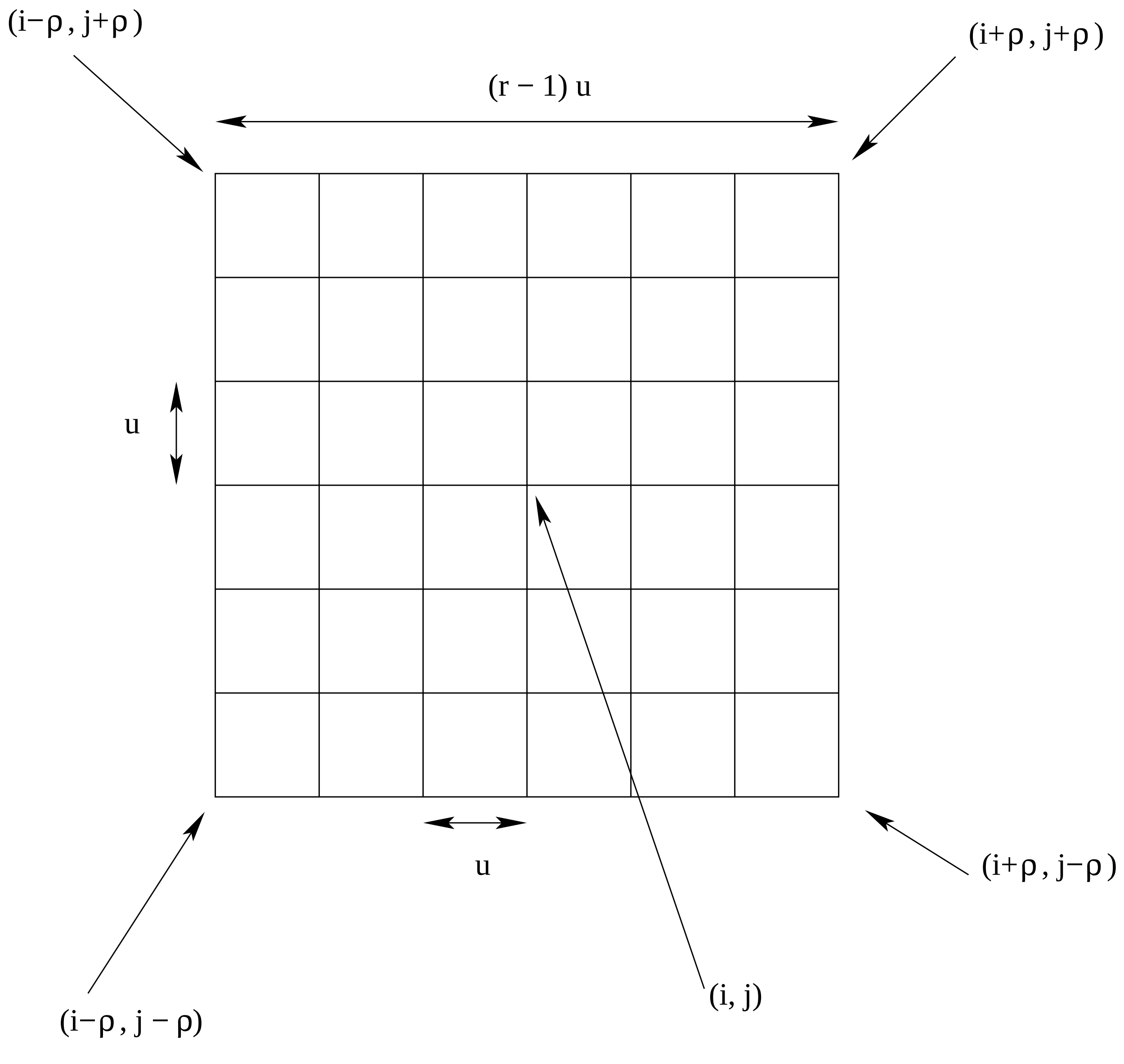

Figure 7.1: The Two-dimensional Grid (with $\rho = \frac{r-1}{2}u$)

they are arranged in the form of a matrix of size $r \times r$. If such a transversal design is used to construct a sensor network as described above and deployed on a grid of size $r \times r$, one can easily associate the keys in each sensor node with its positional co-ordinates, thus establishing a natural correspondence between the transversal design and the square grid.

Definition 19 Physical neighbour: *For a given node α located at (i, j) and a pre-specified radius $\rho = t \cdot u$, $t > 0$ integer, we define all the nodes β located at (i', j') satisfying* $\max(|i - i'|, |j - j'|) \leq \rho$ *as the ρ-physical neighbours of α.*

Definition 20 Key-sharing neighbour: *For a given node α located at (i, j) and a pre-specified radius ρ, we define all the nodes γ located at (i', j') satisfying* $\max(|i-i'|, |j-j'|) \leq \rho$ *and having a common key with α as the ρ-key-sharing neighbours of α.*

As ρ increases, more number of nodes are considered to be neighbours and both kinds of neighbours of a given node should increase. Strictly speaking, one should consider a circular region of radius ρ around each node for the radio frequency coverage. However, we are considering the grid only in order to facilitate the treatment. If we assume the circumscribing circle of a square of radius $\rho, 2\rho, \ldots$ for each of the nodes, the correspondence between the square grid and the circular area of coverage is immediate.

Definition 21 Network graph: *A sensor network may be described as an undirected graph $G = (V, E)$ where V is the set of vertices (sensor nodes in our case) and E is the set of edges. We draw an edge between two vertices (sensor nodes) if they share a common key.*

Definition 22 Connectivity of the network: *If there exists a path between any two nodes α and β of the network, the network is said to be connected.*

Definition 23 Coverage of a node: *For a given ρ, if a node is located at (i, j), it is said to cover the complete square having the four corner points (i', j') satisfying* $\max(|i-i'|, |j-j'|) = \rho$. *Here, (i, j) is an internal node. Had it been a boundary node, we would have considered only the part of the square lying inside the grid.*

Definition 24 Coverage of the network: *If any area under observation in the complete grid is covered by at least one node, the network is said to be covered.*

One needs to ensure at any point of time, the network is covered and connected.

Since we need to calculate ρ-key-sharing neighbours of a given node, we must find out whether two nodes (blocks) have a key in common. The blocks are constructed from a transversal design $TD(k, r)$ as described earlier.

Let the blocks be $A_{i,j}$ and $A_{i',j'}$ where $A_{i,j} = \{x, (ix + j) \bmod r : 0 \leq x \leq k - 1\}$ and $A_{i',j'} = \{x, (i'x + j') \bmod r : 0 \leq x \leq k - 1\}$, where $0 \leq i, i', j, j' < r$.

So it boils down to check whether $A_{i,j}$ and $A_{i',j'}$ have a *valid* intersection. Solving for x, we get $(ix + j) \bmod r = (i'x + j') \bmod r$, or, $x = (j' - j)(i - i')^{-1} \bmod r$.

The value of x is a valid value if and only if the inverse exists in the above expression and also the value of x lies in the admissible range, i.e., $i \neq i'$ and $0 \leq x \leq k - 1$.

Let us call the interval $[0, k - 1]$ the "connectivity interval."

7.3 Connectivity Analysis

Consider a square of size $2\rho \times 2\rho$, centred around the node A at (i, j). The square contains $(2\rho + 1)^2 - 1$ sensor nodes excluding (i, j). Thus the "number of nodes that are physical neighbours of A" is $(2\rho + 1)^2 - 1 = 4\rho(\rho + 1)$.

Here, we assume r to be a prime so that $\mathbb{Z}_r$ becomes a field and every non-zero element of this field has an inverse modulo r.

Consider a sensor node at (i, j). For any square of side 2ρ, centred at (i, j), consider the sensor nodes in that square at the points (i', j') which are $(i+I, j+J), (i-I, j+J), (i+I, j-J)$ and $(i - I, j - J)$, where $0 < I \leq \rho$ and $0 \leq J \leq \rho$. Now consider the four cases.

1. Let $i' = i + I, j' = j + J$. Then $x = (j' - j)(i - i')^{-1} \bmod r = J(-I)^{-1} \bmod r = -JI^{-1} \bmod r$.

2. Let $i' = i - I, j' = j + J$. Then $x = (j' - j)(i - i')^{-1} \bmod r = J(I)^{-1} \bmod r = JI^{-1} \bmod r$.

3. Let $i' = i + I, j' = j - J$. Then $x = (j' - j)(i - i')^{-1} \bmod r = -J(-I)^{-1} \bmod r = JI^{-1} \bmod r$.

4. Let $i' = i - I, j' = j - J$. Then $x = (j' - j)(i - i')^{-1} \bmod r = -J(I)^{-1} \bmod r = -JI^{-1} \bmod r$.

We record the above facts as

Lemma 2 *Both the points $(i + I, j + J), (i - I, j - J)$ either share a key with (i, j) when $-JI^{-1} \leq k - 1$ or do not share a key when $-JI^{-1} > k - 1$. Similarly both the points $(I - i, J + j), (I + i, J - j)$ either share a key with (i, j) when $JI^{-1} \leq k - 1$ or do not share a key when $JI^{-1} > k - 1$. Here, $0 < I \leq \rho$ and $0 \leq J \leq \rho$.*

Note 1 *If* $r = \gamma \bmod I$*, then* $I^{-1} \bmod r = \frac{\big((I-\gamma^{-1}) \bmod I\big)r+1}{I} \bmod r$*. Here* $0 < I \leq \rho$ *and* $\gamma \neq 0$*.*

The expression in lemma 2 involves inverses. The above result may be used to simplify the above conditions in certain cases.

For example, let $I = 2$*. Since* r *is a prime,* $r = 1 \bmod 2$*, Or,* $\gamma = 1$*. So* $I^{-1} = \frac{r+1}{2}$*. So lemma 2 states that* (i,j) *is connected to* $(i+I, j+J)$ *provided* $0 \leq -J(\frac{r+1}{2}) \leq k-1$ *for* $J = -2, -1, 0, 1, 2$*.*

If $J = 2$*, then* $0 \leq -(r+1) \leq k-1$*, which is impossible. So* $(i+2, j+2)$ *is not connected to* (i,j)*. If* $J = -2$*, then* $0 \leq (r+1) \leq k-1 \Rightarrow r \leq k-2$*, which is impossible. So* $(i+2, j-2)$ *is not connected to* (i,j)*. If* $J = 0$*, then* $0 \leq 0 \leq k-1$*, which is always true. So* $(i+2, j)$ *is always connected to* (i,j)*. If* $J = 1$*, then* $0 \leq -\frac{r+1}{2} \leq k-1$*, which is impossible. So* $(i+2, j+1)$ *is not connected to* (i,j)*. If* $J = -1$*, then* $0 \leq \frac{r+1}{2} \leq k-1$*, which is true provided* $k \geq \frac{r+3}{2}$*. So* $(i+2, j-1)$ *is connected to* (i,j) *if* $k \geq \frac{r+3}{2}$*.*

Since the sharing or non sharing of a common key is based on the relative distance, we may very well assume the point (i,j) as $(0,0)$ for the purpose of analysis. The important issue is to study the expression $JI^{-1} \bmod r$ (i.e., whether it is $\leq k-1$ or $> k-1$) when $0 < I \leq \rho$ and $0 \leq J \leq \rho$. It should also be noted that all the nodes of the form $(I, 0)$ share a common key with $(0,0)$ and all the nodes of the form $(0, J)$ do not share a common key with $(0,0)$.

Theorem 8 *If* $(i+I, j+J)$ *and* $(i-I, j-J)$ *are connected/disconnected to* (i,j)*, then* $(i-I, j+J)$ *and* $(i+I, j-J)$ *respectively disconnected/connected to* (i,j) *provided*

1. *Either* $x < k < \frac{r-1}{2}$
2. *Or,* $x > k > \frac{r+1}{2}$

Proof :

1. If $x < k < \frac{r-1}{2}$, then $-x > -k > -\frac{r-1}{2} \Rightarrow r - x > r - k > r - \frac{r-1}{2} = \frac{r+1}{2} > k$.
2. Again, if $x > k > \frac{r+1}{2}$, then $-x < -k < -\frac{r+1}{2} \Rightarrow r - x < r - k < r - \frac{r+1}{2} = \frac{r-1}{2} < k$.

■

Let us examine what exactly is happening in each of the four cases described in the beginning of section 7.3. x assumes two values, either JI^{-1} or $-JI^{-1}$ where $0 < I \leq \rho$ and $0 \leq J \leq \rho$. These two values are the additive inverses of each other. If r be a prime, one may be obtained by subtracting the other from r. One can again have the following cases :

1. If both the values are in the "connectivity interval," i.e.,
   $0 \leq x \leq k-1$ and $0 \leq r-x \leq k-1 \Rightarrow r-k+1 \leq x \leq k-1 \Rightarrow \frac{k-1}{r} \geq \frac{1}{2}$
2. If both the values are outside the "connectivity interval," i.e., $k-1 < x \leq r-1$ and $k-1 < r-x \leq r-1 \Rightarrow k-1 < x < r-k+1 \Rightarrow \frac{k-1}{r} < \frac{1}{2}$
3. If one value is inside and the other value is outside the "connectivity interval," i.e., $0 \leq x \leq k-1$ and $k-1 < r-x \leq r-1 \Rightarrow 1 \leq x \leq r-k+1$

   Now again there may be two cases :

   (a) $k-1 < r-k+1 \Rightarrow \frac{k-1}{r} < \frac{1}{2}$
   (b) $k-1 \geq r-k+1 \Rightarrow \frac{k-1}{r} \geq \frac{1}{2}$

   So either $1 \leq x \leq k-1$ and $\frac{k-1}{r} < \frac{1}{2}$ Or $1 \leq x \leq r-k+1$ and $\frac{k-1}{r} \geq \frac{1}{2}$

7.3.1 Connectivity Ratio

Fix a node α located at (i,j) and radius ρ. Consider the set A_ρ of ρ-key-sharing neighbours of α and the set B_ρ of ρ-physical neighbours of α. Recall that $A_\rho = \{(i \pm I, j \pm J) : 0 \leq I, J \leq \rho = tu,\ 0 \leq -JI^{-1} \bmod r \leq k-1,\ I \neq 0\}$ and $B_\rho = \{(i \pm I, j \pm J) : 0 \leq I, J \leq \rho = tu,\ I, J \textit{ both not zero}\}$.

Clearly $|B_\rho| = (2t+1)^2 - 1 = \left(\frac{2\rho+u}{u}\right)^2 - 1 = (2\rho+1)^2 - 1$ (putting $u = 1$). Or, $|B_\rho| = 4\rho(\rho+1)$.

Definition 25 *Let us denote the "connectivity ratio" by $\mathcal{R}_\rho$ and define it as $\mathcal{R}_\rho = \frac{|A_\rho|}{|B_\rho|}$, i.e., the ratio of "number of nodes that are both physical and key-sharing neighbours of A" and the "number of nodes that are physical neighbours of A."*

Proposition 1 *For $1 < k < r$, $\mathcal{R}_1 = 0.5$.*

Proof : If $\rho = 1$, there are eight points surrounding a specific point $(0,0)$. Each of the two points $(1,0)$ and $(-1,0)$ shares a common key with $(0,0)$. Each of the two points $(0,1)$ and $(0,-1)$ does not share a common key with $(0,0)$. Each of the two points $(1,-1)$ and $(-1,1)$ shares a common key with $(0,0)$ as in that case $JI^{-1} \bmod r = 1 \leq k-1$. Each of the two points $(1,1)$ and $(-1,-1)$ does not share a common key with $(0,0)$ as in that case $JI^{-1} = -1 \bmod r = r-1 > k-1$. Thus, $\mathcal{R}_1 = \frac{4}{8} = 0.5$. ■

Proposition 2 *For $1 < k < r$, $\mathcal{R}_{\frac{r-1}{2}} = \frac{k}{r+1}$.*

Proof : For $\rho = \frac{r-1}{2}$, $|A_\rho| = k(r-1)$ and $|B_\rho| = r^2 - 1$. Hence $\mathcal{R}_\rho = \frac{|A_\rho|}{|B_\rho|} = \frac{k(r-1)}{r^2-1} = \frac{k}{r+1}$. ■

Theorem 9 $\frac{1}{2\rho} \leq \mathcal{R}_\rho \leq \frac{k(r-1)}{4\rho(\rho+1)}$ *for* $\rho = 1, 2, \cdots, \frac{r-1}{2}$.

Proof : Consider the set A_ρ. All the 2ρ nodes $(i + I, J)$ for $I = \pm 1, \pm 2, \cdots, \pm \rho$ are connected to (i, j). Also the two nodes $(i+1, j-1)$ and $(i-1, j+1)$ are connected to (i, j) which is evident from lemma 2. The total number of nodes connected to (i, j) is $k(r-1)$. Hence $2\rho + 2 \leq |A_\rho| \leq k(r-1)$. Now $|B_\rho| = 4\rho(\rho+1)$ and $\mathcal{R}_\rho = \frac{|A_\rho|}{|B_\rho|}$. Hence $\frac{2\rho+2}{4\rho(\rho+1)} \leq \mathcal{R}_\rho \leq \frac{k(r-1)}{4\rho(\rho+1)}$. Now $\frac{2\rho+2}{4\rho(\rho+1)} = \frac{1}{2\rho}$ Hence $\frac{1}{2\rho} \leq \mathcal{R}_\rho \leq \frac{k(r-1)}{4\rho(\rho+1)}$ for $\rho = 1, 2, \cdots, \frac{r-1}{2}$. ■

In table 7.1, we show the values of $\mathcal{R}_\rho$ for $k = 7$ and $r = 53$.

In the figures 7.2, 7.3 and 7.4, we have illustrated the three cases:

1. If $\frac{k}{r+1} < \frac{1}{2}$, then $\frac{k}{r+1} \leq \mathcal{R}_\rho \leq \frac{1}{2}$. $(k = 2, r = 13)$
2. If $\frac{k}{r+1} = \frac{1}{2}$, then $\mathcal{R}_\rho = \frac{1}{2}$. $(k = 7, r = 13)$
3. If $\frac{k}{r+1} > \frac{1}{2}$, then $\frac{1}{2} \leq \mathcal{R}_\rho \leq \frac{k}{r+1}$. $(k = 12, r = 13)$

The values of $\mathcal{R}_\rho$ are compared to the bounds obtained from theorem 9.

7.3.2 Coverage

The area covered by a single node is $= (2\rho)^2 = 4\rho^2$. The area of the square grid $= (r-1)^2 u^2$. So minimum number of nodes required to "cover" the entire grid $= \left\lceil \left(\frac{(r-1)u}{2\rho}\right)^2 \right\rceil = \left\lceil \left(\frac{r-1}{2t}\right)^2 \right\rceil$.

7.3.3 Connectivity Does not Imply Coverage and Vice Versa

Consider any single column of the $(r-1)u \times (r-1)u$ grid and the corresponding column of the transversal design. It is obvious that the network comprising of r nodes is connected. However, they may not cover the entire grid unless ρ is not very large compared to $(r-1)u$. On the other hand, one can easily locate $\left\lceil \left(\frac{r-1}{2t}\right)^2 \right\rceil$ nodes covering the entire grid but there may be two nodes present that are not connected by single/multi hop paths.

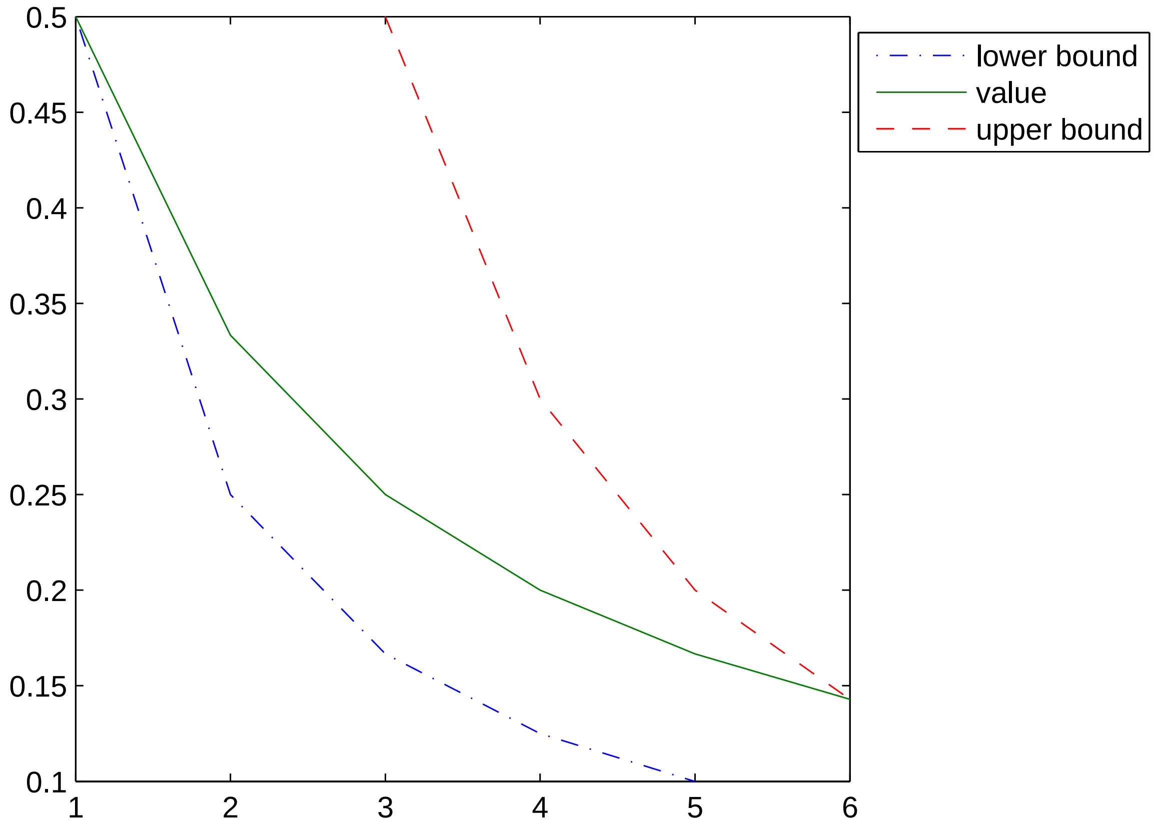

Figure 7.2: If $\frac{k}{r+1} < \frac{1}{2}$, then $\frac{k}{r+1} \leq \mathcal{R}_\rho \leq \frac{1}{2}$ $(k = 2, r = 13)$

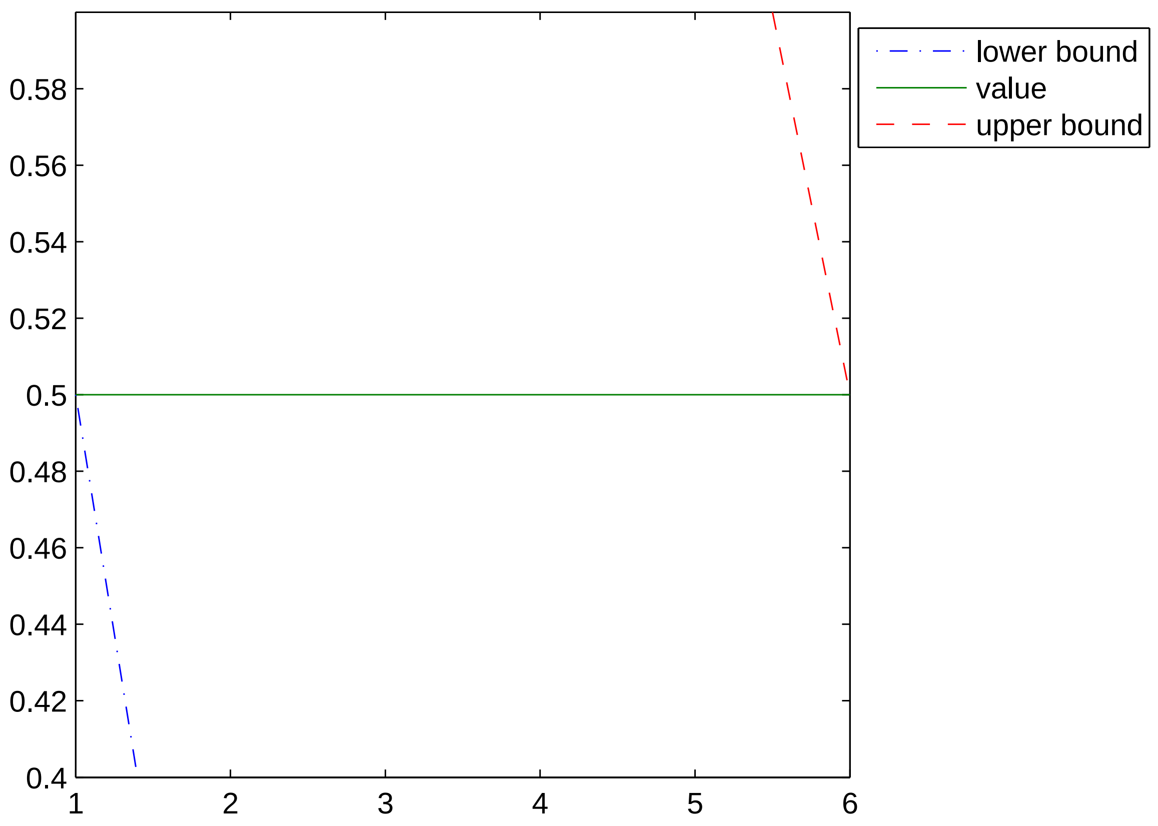

Figure 7.3: If $\frac{k}{r+1} = \frac{1}{2}$, then $\mathcal{R}_\rho = \frac{1}{2}$ $(k = 7, r = 13)$

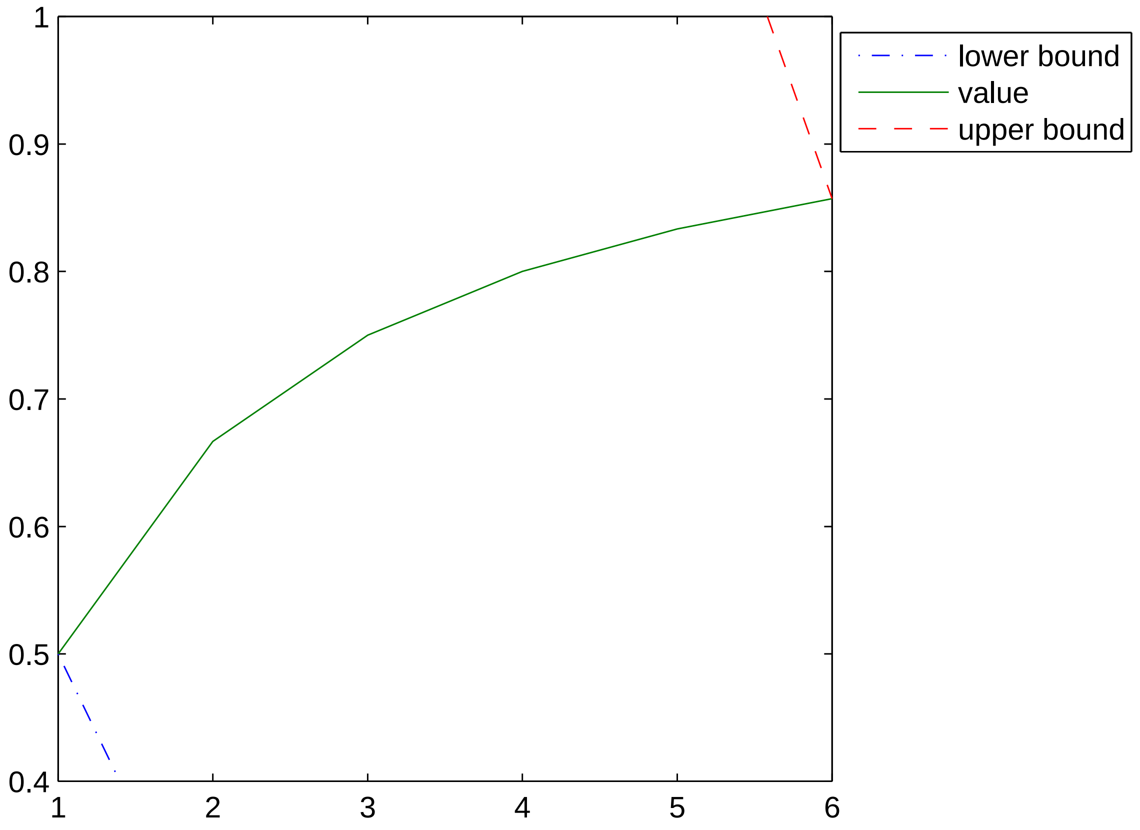

Figure 7.4: If $\frac{k}{r+1} > \frac{1}{2}$, then $\frac{1}{2} \leq \mathcal{R}_\rho \leq \frac{k}{r+1}$ $(k = 12, r = 13)$

7.4 A Design Methodology for a Sensor Network

Suppose we know the area of deployment of the sensor network. For example, if one wants to monitor the temperature of a production unit, it is possible to divide the area in the form of a grid and place the sensor nodes in the intersection points of the grid. The communicating radius and the sensing radius are also known and we choose the lesser one for the purpose of our calculation and denote it by ρ. If we pre-distribute the keys according to a transversal design, we can have several design choices depending on the permissible value of the probability that the network should fail. Here, by "failure," we shall mean that the enter area to be monitored has to be connected and covered by the sensor nodes. A certain fraction of nodes may always fail, still the robustness of the network may be chosen according to user requirement. Consider the event that despite the failure of s nodes, the entire area is covered by the sensor nodes and the nodes form a connected network. Denote the probability of this event by $R(s)$. In fact, this $R(s)$ gives a measure of robustness of the network. Now the number of keys k in each node, the size of the area $(r-1)u \times (r-1)u$, the radius ρ and the measure *fail(s)* are inter-related. Assuming a random mode of failure of the sensor nodes, we have a simple program that gives an estimate of $R(s)$. The experimental values are tabulated. These values are obtained from a the output of a C program and the average value of 100 runs is tabulated as the value of $R(s)$ in the tables 7.2, 7.3.

It may be noted that in the tables 7.2, 7.3, certain combinations of values are not tabulated for $k = 2, 3, 4, 5$, $\rho = 1, 2, 3, 4, 5$, $s = 10, 20, \cdots, 100$ and also $s = 200, 300, \cdots, 1000$. For those values, $R(s) = 0$ whenever $\rho = 1$ and $R(s) = 1$ otherwise.

Example 6 *Suppose the user specifies that the grid size to be covered is* $r = 53$. *The robustness* $R(s)$ *to be maintained is* 90% *at the level* $s = 1000$. *It is immediate from table 7.3 that even with only* $k = 2$ *keys per node, he can achieve this objective.*

7.5 Conclusion and Future Research

In this chapter, we assume that the nodes fail/are put to sleep randomly and we did not specify any particular algorithm that determines which nodes will remain alive and which nodes will be inactive so that the longevity of the network is increased. However, by increasing the number of inactive nodes, one can arrive at a configuration that can maintain a connected network and at the same time can cover all the grid points. It is easy to find two disjoint configurations with only half of the total nodes being live. One can assume that half of the nodes will be alive at any given point of time and the rest will sleep and conserve energy. In our future work, we shall give deterministic algorithms for sleeping of the sensor nodes.

ρ	No. of Physical Neighbours	No. of Physical and key-sharing Neighbours	$\mathcal{R}_\rho$
1	8	4	0.50000
2	24	10	0.41667
3	48	16	0.33333
4	80	24	0.30000
5	120	30	0.25000
6	168	40	0.23810
7	224	44	0.19643
8	288	54	0.18750
9	360	64	0.17778
10	440	76	0.17273
11	528	86	0.16288
12	624	102	0.16346
13	728	116	0.15934
14	840	128	0.15238
15	960	140	0.14583
16	1088	156	0.14338
17	1224	174	0.14216
18	1368	194	0.14181
19	1520	212	0.13947
20	1680	232	0.13810
21	1848	252	0.13636
22	2024	272	0.13439
23	2208	296	0.13406
24	2400	318	0.13250
25	2600	340	0.13077
26	2808	364	0.12963

Table 7.1: A few values of $\mathcal{R}_\rho$ for $k = 7$ and $r = 53$

k	ρ	s	$R(s)$		k	ρ	s	$R(s)$		k	ρ	s	$R(s)$		k	ρ	s	$R(s)$
2	2	10	0.88		3	2	10	0.90		4	2	10	0.93		5	2	10	0.92
2	2	20	0.86		3	2	20	0.90		4	2	20	0.92		5	2	20	0.92
2	2	30	0.86		3	2	30	0.90		4	2	30	0.92		5	2	30	0.92
2	2	40	0.86		3	2	40	0.89		4	2	40	0.92		5	2	40	0.92
2	2	50	0.86		3	2	50	0.89		4	2	50	0.92		5	2	50	0.92
2	2	60	0.86		3	2	60	0.88		4	2	60	0.92		5	2	60	0.92
2	2	70	0.86		3	2	70	0.88		4	2	70	0.91		5	2	70	0.92
2	2	80	0.86		3	2	80	0.88		4	2	80	0.91		5	2	80	0.92
2	2	90	0.85		3	2	90	0.87		4	2	90	0.91		5	2	90	0.92
2	2	100	0.85		3	2	100	0.87		4	2	100	0.91		5	2	100	0.92

Table 7.2: Values of k, ρ, s and $R(s)$ for small values of s (for $r = 53$)

k	ρ	s	$R(s)$		k	ρ	s	$R(s)$		k	ρ	s	$R(s)$		k	ρ	s	$R(s)$
2	2	200	0.68		3	2	200	0.87		4	2	200	0.86		5	2	100	0.92
2	2	300	0.63		3	2	300	0.85		4	2	300	0.83		5	2	200	0.92
2	2	400	0.48		3	2	400	0.77		4	2	400	0.81		5	2	300	0.90
2	2	500	0.41		3	2	500	0.72		4	2	500	0.71		5	2	400	0.84
2	2	600	0.26		3	2	600	0.63		4	2	600	0.66		5	2	500	0.80
2	2	700	0.16		3	2	700	0.55		4	2	700	0.56		5	2	600	0.69
2	2	800	0.06		3	2	800	0.48		4	2	800	0.42		5	2	700	0.57
2	2	900	0.03		3	2	900	0.34		4	2	900	0.32		5	2	800	0.51
2	2	1000	0.01		3	2	1000	0.19		4	2	1000	0.23		5	2	900	0.39
2	3	200	0.98		3	3	100	0.99		4	3	300	0.96		5	2	1000	0.27
2	3	300	0.96		3	3	200	0.97		4	3	400	0.95		5	3	200	0.96
2	3	400	0.95		3	3	300	0.97		4	3	500	0.95		5	3	300	0.96
2	3	500	0.93		3	3	400	0.97		4	3	600	0.95		5	3	400	0.95
2	3	600	0.86		3	3	500	0.94		4	3	700	0.94		5	3	500	0.95
2	3	700	0.83		3	3	600	0.91		4	3	800	0.92		5	3	600	0.95
2	3	800	0.77		3	3	700	0.90		4	3	900	0.90		5	3	700	0.94
2	3	900	0.71		3	3	800	0.84		4	3	1000	0.87		5	3	800	0.93
2	3	1000	0.56		3	3	900	0.81		4	4	200	0.99		5	3	900	0.91
2	4	200	0.99		3	3	1000	0.77		4	4	300	0.99		5	3	1000	0.89
2	4	300	0.97		3	4	600	0.99		4	4	400	0.99		5	4	100	0.99
2	4	400	0.97		3	4	700	0.98		4	4	500	0.99		5	4	200	0.99
2	4	500	0.97		3	4	800	0.98		4	4	600	0.99		5	4	300	0.99
2	4	600	0.96		3	4	900	0.96		4	4	700	0.99		5	4	400	0.99
2	4	700	0.96		3	4	1000	0.96		4	4	800	0.98		5	4	500	0.99
2	4	800	0.94		3	5	200	0.99		4	4	900	0.97		5	4	600	0.99
2	4	900	0.93		3	5	300	0.99		4	4	1000	0.98		5	4	700	0.97
2	4	1000	0.89		3	5	400	0.99		4	5	200	0.99		5	4	800	0.95

Table 7.3: Values of k, ρ, s and $R(s)$ for large values of s (for $r = 53$)

Chapter 8

The Deterministic Extension of the Sensor Network

In chapter 4, we have shown the construction of designs probabilistically. In this chapter, we shall show that it is possible to construct deterministic designs which may also be used. We revert to our usage in chapter 3 of regarding the (simple) sensor network as a (0,1) matrix, A, the incidence matrix of the sensor network. Clearly any (0,1) matrix could be used but we assigned properties to A so that we could ensure certain properties and obtain bounds on other properties. In previous work, it has been assumed that the number of common keys between any two blocks has been guaranteed to assume one of a set with few values. Historically, the values would be "smallish" due to the storage limits of the sensor device and the average number of common keys tends to decrease. It is also not easy to find an accurate estimate of *fail(s)* . However, there are methods which could be used to guarantee upper and lower limits on the numbers of common keys and bounds on *fail(s)*, the negative resilience measure.

In this chapter, we consider the application of some other combinatorial designs to sensor network key distribution.

It may be noted that throughout this chapter, the rows of the incidence matrix (of a combinatorial design) will correspond to the sensor devices and the columns will correspond to the keys.

Linkage: *The minimum number of common* 1*s between any two rows of the incidence matrix of a combinatorial design is denoted by* Λ *and defined as the "linkage" of the incidence matrix or the corresponding combinatorial design.*

1. *BIBDs:* If we consider the matrix B, the incidence matrix of a BIBD (v, b, r, k, λ), then it is easy to see that any two rows will have λ elements in common. In other words,

the "dot product," or "linkage" of the incidence matrix will be $\Lambda = \lambda$. We shall prefer to use this notion and think of the rows of the incidence matrix, B, as our "sensor nodes" and the columns as the "keys."

If B constitutes the whole of the incidence matrix of the sensor network, then using the desirability criteria listed in Section 3.1 we can choose any BIBD with suitably large v, suitably small b. Since $bk = vr$, we have the number of keys r given by $r = bk/v$. Because we have the "linkage property" (provided $\lambda > 0$) every pair of nodes will have a pair of keys in common and can communicate directly.

For the notations, please refer to chapter 3.

2. *PBIBDs:* If we extend the incidence matrix B of the previous subsection to a PBIBD

$$(v, b, r, k, \lambda_1, \lambda_2, \cdots, \lambda_\ell)$$

then any two blocks will have $\Lambda = min\{\lambda_1, \lambda_2, \cdots, \lambda_\ell\}$ keys in common and so provided $\Lambda \neq 0$ again every pair of nodes will have a pair of keys in common and can communicate directly.

For the notations, please refer to chapter 3.

3. *Pairwise Balanced Designs PBD($\mathcal{K}$, λ):* If the incidence matrix of a PBD($\mathcal{K}$, λ) is used as the incidence matrix of a sensor network, the number of keys in each sensor node is less than the size of the maximum element in $\mathcal{K}$. Hence there is no need to restrict r, k or λ. For the notations, please refer to chapter 3.

8.1 Some Thoughts on Constructions

Since our aim is to have large number of sensor nodes sharing common keys between any two nodes, we illustrate how we may begin with a pair of small known BIBDs and generate large incidence matrices of sensor networks from them.

Consider two BIBDs: $(v_1, b_1, r_1, k_1, \lambda_1)$ and $(v_2, b_2, r_2, k_2, \lambda_2)$ such that $v_2 = r_1$. Let X and Y denote their respective incidence matrices.

Theorem 10 *Replace the ith 1 of each row of X by the ith row of Y for $i = 1, 2, \cdots, r_1$. The zeros in X are replaced by the appropriate $1 \times b_2$ matrix of zeros. The resultant matrix Z has $\lambda_1\lambda_2$ common 1s between any two rows.*

Proof : Consider the p-th row and the q-th row of the Z, $1 \leq p, q \leq r_1$. The p-th row of Z is formed by replacing the i-th 1 of the p-th row of X by the i-th row of Y, $i = 1, \cdots, r_1$.

Similarly the q-th row of Z is formed by replacing the i-th 1 of the q-th row of X by the i-th row of Y, $i = 1, \cdots, r_1$. In X, there were λ_1 1s common between the p-th row and the q-th row. Similarly, in Y, there were λ_2 1s common between the p-th row and the q-th row. Now for each of the λ_1 common 1s between the p-th row and the q-th row, there will be λ_2 additional common 1s contributed by Y. Hence the number of common 1s between the p-th row and the q-th row is $\lambda_1\lambda_2$. ■

This incidence is of the type described early in this chapter and so it can be used as the "linked" incidence matrix of any sensor network where each node can reach any other node in at most one step provided $\lambda_1\lambda_2 > 0$.

Example 7 *Consider two specific SBIBDs. For example, consider a Hadamard Design* $(4t-1, 4t-1, 2t-1, 2t-1, t-1)$ *and a Projective Plane*$(n^2+n+1, n^2+n+1, n+1, n+1, 1)$. *These can be combined as in Theorem 10 to produce the incidence matrix of a sensor network which will have* $v = b = (n^2+n+1) = (2t-1)$, $k = (n+1)(2t-1)$ *and "linkage"* $\Lambda = t-1$. *The construction of the theorem requires that* $v_2 = r_1$, *which forces us to have* $v = b = (n^2+n+1) = (2t-1)$. *Note that all the rows of the resulting incidence matrix may or may not be used.*

Specifically choose t *and* n *such that* $2t-1 = n^2+n+1 \Longrightarrow t = \frac{n^2+n+2}{2}$.

As a concrete example, let $n = 3$, *then* $t = 7$. *So we have a Hadamard Design* $(27, 27, 13, 13, 6)$ *and a Projective Plane* $(13, 13, 4, 4, 1)$.

Consider the incidence matrix of the Hadamard Design. Denote it by A *where* A *is of order* 27×27. *Also consider the incidence matrix of the Projective Plane. Denote it by* B *where* B *is of order* 13×13.

Now replace the i*th* 1 *of each row by the* i*th row of* B *for* $i = 1, 2, \cdots, 13$. *More generally, one can replace it by* s *copies of the* i*th row of* B *and the* 0*s of* A *are replaced by* 1×13 *matrix of* 0*s.*

The resulting matrix will be of order $27 \times 27 \cdot 13s$. *However, the number of common* 1*s between any two rows will be* $6s$.

One problem with the previous result is that although the number of common keys between any two nodes increases, the total number of nodes does not increase. The next result is suitable for "blowing up" the design because we can start with v_1 much less compared to b_1 and v_2 much less compared to b_2.

Consider two BIBDs: $X = (v_1, b_1, r_1, k_1, \lambda_1')$ and $Y = (v_2, b_2, r_2, k_2, \lambda_2')$. Denote their incidence matrices by A and B respectively. Consider the Kronecker product $C = A \otimes B$. We have the following well known result [99]. We give a proof for the sake of completeness. More importantly, the technique of the proof will allow us to obtain good bounds on the resilience measure *fail(s)*.

Theorem 11 *Let C be the Kronecker product of the incidence matrices of two BIBDs $X = (v_1, b_1, r_1, k_1, \lambda'_1)$ and $Y = (v_2, b_2, r_2, k_2, \lambda'_2)$. Then C is the incidence matrix of a PBIBD with (at most) three associate classes and parameters $v = v_1 \cdot v_2$, $b = b_1 \cdot b_2$, $r = r_1 \cdot r_2$, $k = k_1 \cdot k_2$, $\lambda_1 = \lambda'_1 \cdot \lambda'_2$, $\lambda_2 = \lambda'_1 \cdot r_2$, $\lambda_3 = r_1 \cdot \lambda'_2$.*

Proof : We define the associates classes as follows: two rows of C, the $(x_1v_2 + y_1)$th row and $(x_2v_2 + y_2)$th row are

1. 1st associates if $x_1 \neq x_2$ and $y_1 \neq y_2$.
2. 2nd associates if $x_1 \neq x_2$ and $y_1 = y_2$.
3. 3rd associates if $x_1 = x_2$ and $y_1 \neq y_2$.

where $0 \leq x_1, x_2 \leq v_1$ and $0 \leq y_1, y_2 \leq v_2$.

It may be verified that
$p^1_{11} = (v_1 - 2)(v_2 - 2)$
$p^1_{12} = p^1_{21} = (v_1 - 2)$
$p^1_{13} = p^1_{31} = (v_2 - 2)$
$p^1_{22} = 0$
$p^1_{23} = p^1_{32} = 1$
$p^1_{33} = 0$
$p^2_{11} = (v_1 - 2)(v_2 - 1)$
$p^2_{12} = p^2_{21} = 0$
$p^2_{13} = p^2_{31} = (v_2 - 1)$
$p^2_{22} = (v_1 - 2)$
$p^2_{23} = p^2_{32} = 0$
$p^2_{33} = 0$
$p^3_{11} = (v_1 - 1)(v_2 - 2)$
$p^3_{12} = p^3_{21} = (v_1 - 1)$
$p^3_{13} = p^3_{31} = 0$
$p^3_{22} = 0$
$p^3_{23} = p^3_{32} = 0$
$p^3_{33} = (v_2 - 2)$

The above follows from (1), (2) and (3), using simple counting arguments.

$$\text{So } P_1 = \begin{pmatrix} (v_1-2)(v_2-2) & (v_1-2) & (v_2-2) \\ (v_1-2) & 0 & 1 \\ (v_2-2) & 1 & 0 \end{pmatrix}$$

$$P_2 = \begin{pmatrix} (v_1-2)(v_2-1) & 0 & (v_2-1) \\ 0 & (v_1-2) & 0 \\ (v_2-1) & 0 & 0 \end{pmatrix}$$

$$P_3 = \begin{pmatrix} (v_1-1)(v_2-2) & (v_1-1) & 0 \\ (v_1-1) & 0 & 0 \\ (v_2-1) & 0 & (v_2-2) \end{pmatrix}$$

Also $n_1 = (v_1-1)(v_2-1)$, $n_2 = v_1 - 1$ and $n_3 = v_2 - 1$.

For our analysis of *fail(s)* we now explicitly obtain the values of λ_1, λ_2 and λ_3. ■

Lemma 3 *Let C be the Kronecker product of the incidence matrices of two BIBDs $X = (v_1, b_1, r_1, k_1, \lambda_1')$ and $Y = (v_2, b_2, r_2, k_2, \lambda_2')$. If a sensor network is constructed using C as the underlying incidence structure, then there will be at least τ_{min} common keys between any two nodes of the sensor network where*

$$\tau_{min} = min\{\lambda_1 = \lambda_1' \cdot \lambda_2',\ \lambda_2 = \lambda_1' \cdot r_2,\ \lambda_3 = r_1 \cdot \lambda_2'.\}$$

Proof :

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,b_1} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,b_1} \\ \cdots & \cdots & \cdots & \cdots \\ a_{v_1,1} & a_{v_2,2} & \cdots & a_{v_1,b_1} \end{pmatrix}$$

$$B = \begin{pmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,b_2} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,b_2} \\ \cdots & \cdots & \cdots & \cdots \\ b_{v_2,1} & b_{v_2,2} & \cdots & b_{v_2,b_2} \end{pmatrix}$$

$$C = A \otimes B = \begin{pmatrix} a_{1,1}B & a_{1,2}B & \cdots & a_{1,b_1}B \\ a_{2,1}B & a_{2,2}B & \cdots & a_{2,b_1}B \\ \cdots & \cdots & \cdots & \cdots \\ a_{v_1,1}B & a_{v_2,2}B & \cdots & a_{v_1,b_1}B \end{pmatrix}$$

It may be noted that the $xv_2 + y$th row of C has the following structure:
$a_{x+1,1}(b_{y,1} \quad b_{y,2} \quad \cdots \quad b_{y,b_2}) \quad a_{x+1,2}(b_{y,1} \quad b_{y,2} \quad \cdots \quad b_{y,b_2}) \quad \cdots \quad a_{x+1,b_1}(b_{y,1} \quad b_{y,2} \quad \cdots \quad b_{y,b_2})$
where $0 \le x \le v_1$ and $0 \le y \le v_2$.

Consider the dot product τ of the $x_1v_2+y_1$th and $x_2v_2+y_2$th rows of C where $0 \leq x_i \leq v_1$ and $0 \leq y_i \leq v_2$, $i = 1, 2$. This can take one of three values λ_1, λ_2, or λ_3, depending on the rows chosen.

1. If $x_1 \neq x_2$, then $a_{x_1+1,i} = a_{x_2+1,i}$ for λ'_1 values of i. For each of these λ'_1 values of i, the dot product of $(b_{y_1,1} \quad b_{y_1,2} \quad \cdots \quad b_{y_1,b_2})$ and $(b_{y_2,1} \quad b_{y_2,2} \quad \cdots \quad b_{y_2,b_2})$ will be λ'_2 if $y_1 \neq y_2$ and r_2 if $y_1 = y_2$.

   So the dot product, λ_1 is $\lambda'_1\lambda'_2$ if $x_1 \neq x_2$ and $y_1 \neq y_2$.

   The dot product λ_2 is $\lambda'_1 r_2$ if $x_1 \neq x_2$ and $y_1 = y_2$. Note that $x_1 \neq x_2$ and $y_1 \neq y_2$ is possible in $\frac{v_1(v_2-1)v_2(v_1-1)}{2} = 2\binom{v_1}{2}\binom{v_2}{2}$ ways. Also $x_1 \neq x_2$ and $y_1 = y_2$ is possible in $\frac{v_1(v_1-1)v_2}{2} = \binom{v_1}{2}v_2$ ways.

2. If $x_1 = x_2$, then $a_{x_1+1,i} = a_{x_2+1,i}$ for r_1 values of i. For each of these r_1 values of i, the dot product of $(b_{y_1,1} \quad b_{y_1,2} \quad \cdots \quad b_{y_1,b_2})$ and $(b_{y_2,1} \quad b_{y_2,2} \quad \cdots \quad b_{y_2,b_2})$ will be λ'_2 if $y_1 \neq y_2$ and r_2 if $y_1 = y_2$.

   So the dot product λ_3 is $r_1\lambda'_2$ if $x_1 = x_2$ and $y_1 \neq y_2$ and r_1r_2 if $x_1 = x_2$ and $y_1 = y_2$. Again $x_1 = x_2$ and $y_1 \neq y_2$ may happen in $\frac{v_1v_2(v_2-1)}{2} = v_1\binom{v_2}{2}$ ways.

We therefore see that if C were used to design the sensor network we would have the result of the enunciation. ■

Corollary 1 *Average value of* $\tau = \frac{\lambda'_1\lambda'_2\left(2\binom{v_1}{2}\binom{v_2}{2}\right)+\lambda'_1 r_2\left(\binom{v_1}{2}v_2\right)+r_1\lambda'_2\left(v_1\binom{v_2}{2}\right)}{2\binom{v_1}{2}\binom{v_2}{2}+\binom{v_1}{2}v_2+v_1\binom{v_2}{2}}$
$= \lambda'_1\lambda'_2\frac{(v_1-1)(v_2-1)}{v_1v_2-1} + \lambda'_1 r_2\frac{v_1-1}{v_1v_2-1} + r_1\lambda'_2\frac{v_2-1}{v_1v_2-1} \approx \lambda'_1\lambda'_2\left(1-\frac{1}{v_1}-\frac{1}{v_2}\right) + \lambda'_1 r_2\left(\frac{1}{v_2}\right) + r_1\lambda'_2\left(\frac{1}{v_1}\right)$.
In other words, the average value will be very close to $\lambda'_1\lambda'_2$.

Consider any two rows of the incidence matrix C who share the minimum number of keys, τ_{min}. Denote these keys by $\alpha_1, \cdots, \alpha_{\tau_{min}}$. Recall that the resilience measure *fail(s)* means the probability of the key(s) shared between two given nodes is/are revealed consequent upon the failure of s random nodes. In this context, *fail(s)* $= Prob(\bigcap_{i=1}^{\tau_{min}} E_i)$ where E_i denotes the event that the α_i-th key is revealed, $i = 1, \cdots, \tau_{min}$.

Theorem 12 fail(s)$\leq [Prob(\bigcap_{i=1}^{\tau_{min}} E_i)]_{\tau_{min}=1} = 1 - \left(\frac{\binom{v_1v_2-k_1k_2}{s}}{\binom{v_1v_2-2}{s}}\right) \leq 1 - \left(\frac{v_1v_2-k_1k_2-s+1}{v_1v_2-s-1}\right)^s$.

Note: The above theorem gives a rather loose bound for *fail(s)* . Since the number of keys is large, it will be more reasonable to estimate *fail(s)* by considering the E_is to be independent, $i = 1, \cdots, \tau_{min}$.

Example 8 *Consider two specific BIBDs,* $X = (v_1 = 28, b_1 = 36, r_1 = 9, k_1 = 7, \lambda'_1 = 2)$ *and* $Y = (v_2 = 91, b_2 = 91, r_2 = 10, k_2 = 10, \lambda'_2 = 1)$*.*

If we take the Kronecker Product of the incidence matrices of X *and* Y*, the resulting three associate class PBIBD will have the following parameters:* $v = 28 \cdot 91 = 2548$*,* $b = 36 \cdot 91 = 3276$*,* $r = 9 \cdot 10 = 90$*,* $k = 7 \cdot 10 = 70$*,* $\lambda_1 = 2 \cdot 1 = 2$*,* $\lambda_2 = 2 \cdot 10 = 20$*,* $\lambda_3 = 9 \cdot 1 = 9$*.*

$$\tau = \begin{cases} \lambda_1 = 2 \text{ (} \textit{happens } 2\binom{28}{2}\binom{91}{2} = 3095820 \textit{ times)} \\ \lambda_2 = 20 \text{ (} \textit{happens } \binom{28}{2}91 = 34398 \textit{ times)} \\ \lambda_3 = 9 \text{ (} \textit{happens } 28\binom{91}{2} = 114660 \textit{ times)} \end{cases}$$

Average value of $\tau = 2.43816254$*, which is very close to* $\lambda'_1\lambda'_2 = 2 \cdot 1 = 2$*.*

$$fail(10) \le 1 - \left(\frac{\binom{2478}{10}}{\binom{2546}{10}}\right) = 0.2375 \le 1 - \left(\frac{2469}{2537}\right)^{10} = 0.2379.$$

However, it appears more reasonable to estimate $fail(10)$ *by assuming that the revealing of the keys are independent of each other (as described in the note above).*
So $fail(10) \approx \frac{k_1k_2}{v_1v_2} \cdot \frac{k_1k_2}{v_1v_2} = \frac{70}{2548} \cdot \frac{70}{2548} = 0.00075$*.*

Remark 3 *Let us compare the examples given in section 4.2.6 and 8. The number of keys in each node is between* 126 *and* 128 *in example of section 4.2.6, but it is only* 90 *in example 8. Though the average number of common keys is higher (more than* 5*) in example of section 4.2.6, but there are cases where two nodes may not share a common key. Whereas, in example 8, the average number of common keys is fewer, but it is guaranteed to have at least* 2 *common keys between any two blocks. If we compare the* fail(10) *values, we find that in the example of section 4.2.6, it is around* 2%*. In example 8, it is less than* 23%*, but in reality, it appears to be much less since the bound is not tight. In fact, the value should be close to* 0.075%*.*

We have compiled a long table of PBIBDs. The original table is available at [104]. In fact, it is evident that given a table of n BIBDs, one can generate a new table of $\frac{n(n+1)}{2}$ PBIBDs. A few rows from the original table is given in table 8.1.

8.2 Key Agreement

For the key agreement purpose, let each node be given an identification number, which in binary represents the row corresponding to that particular node of the incidence matrix. Each key is assigned an index and that index is stored in each node along with the keys. Let two nodes have the *ids* ν_0 and ν_1 and they want to agree upon a common key. At first, they exchange their respective *ids*. Next they calculate the logical AND of the binary

v	b	r	k	λ_1	λ_2	λ_3
2541	2772	60	55	1	5	12
2541	9240	120	33	1	10	12
2541	5544	120	55	2	10	24
2541	3960	120	77	3	10	36
2541	5544	144	66	3	12	36
2541	8316	180	55	3	36	15
2541	4620	180	99	6	72	15
2548	5733	90	40	1	9	10
2548	12285	135	28	1	9	15
2548	3276	90	70	2	9	20
2548	7020	135	49	2	9	30
2548	3822	150	100	5	50	15
2548	8190	225	70	5	75	15
2553	2553	121	121	5	55	11
2555	2555	126	54	5	45	14
2556	5964	210	90	6	42	30
2556	2556	225	225	18	90	45
2560	9360	117	32	1	9	13
2560	3744	117	80	3	27	13
2560	2880	117	104	4	36	13

Table 8.1: A few values from the table of three associate class PBIBDs

representation of the *id* received with the binary representation of its own *id*. So both the nodes now have the knowledge of the indices of the common keys between them from the presence of 1s in the expression. Next, one of them randomly chooses the index of one of the common keys and sends it across to the other node. The other node accepts it.

Algorithm 3 *1. Node ν_i sends its id to node ν_{1-i} where $i = 0, 1$.*

2. Node ν_i computes the logical AND of the binary representation of ν_i and ν_{1-i}. Denote it by x_i, $i = 0, 1$.

3. Node ν_i learns about the common keys from the presence of 1s in x_i, $i = 0, 1$. Let the positions of those 1s be at $t_1, t_2, \cdots, t_\alpha$.

4. Node ν_0 chooses a j randomly from $[1, \alpha]$ and sends it across to ν_1.

5. Now node ν_1 also gets informed that the agreed upon key is the key with index t_j.

The above protocol exchanges bit-strings of size $\lfloor \log_2(v) \rfloor + 1$ where v = no of sensor nodes and stores the indices for each key which is also of the size $\lfloor \log_2(v) \rfloor + 1$, thus effectively increasing the key size by a few bits or equivalently, at the expense of storing a few extra keys in each sensor node.

For example, consider a network with $v = 3276$. The bit-string exchanged will be of length $\lfloor \log_2(3276) \rfloor + 1 = 12$ only, which is less than 2 bytes. For 90 keys per node, it will take an additional $90 \times 12 = 1080$ bits or 135 bytes. If we assume 16 byte keys, the memory requirement for the keys will be $90 \times 16 = 1440$ bytes and the total memory requirement will be $1440 + 135 = 1575$ bytes, which is even less than 2Kbytes.

8.2.1 A Special Case for SBIBDs

There is an elegant algorithm for key agreement if the underlying BIBDs are symmetric and generated from difference sets. Assume that the SBIBD generated from a difference set has the blocks from the elements in $\mathbb{Z}_v$. Let there be v blocks. The i-th block contains the element i for $i \in \mathbb{Z}_v$. So the blocks may be indexed by the elements of $\mathbb{Z}_v$. In other words, we may consider v blocks indexed by $\{0, 1, \cdots, v-1\}$. Let the 0th block be $\{0, \alpha_1, \alpha_2, \cdots, \alpha_{k-1}\}$. So the ith block will be $\{i, \alpha_1 + i, \alpha_2 + i, \cdots, \alpha_{k-1} + i\}$ where $i \in \mathbb{Z}_v$ and the additions are modulo v. Also consider an empty list L of size v having elements $L_0, L_1, \cdots, L_{v-1}$, where each element L_i is a vector of size λ whose elements are in $\mathbb{Z}_v$. Assume that two blocks intersect at λ points.

Method 1 *1. For $i \in \mathbb{Z}_v$, consider the intersection of the i-th block with the 0th block and put the λ elements in L_i. Denote them by $L_{i_1}, L_{i_2}, \cdots, L_{i_\lambda}$.*

2. Let two nodes with indices p and $p + i$ want to agree upon a common key.

3. They calculate the difference in their indices. Here it is i.

4. Look up the i-th element L_i from the list.

5. The common key is $p + L_{i_j}$ where $1 \leq j \leq \lambda$.

Proof of correctness of method 1

We are required to prove that if the 0th block and the ith block intersect at α_n (say), then the pth block and the $(p + i)$th block will intersect at $p + \alpha_n$.
Proof : Consider the 0th block $\{0, \alpha_1, \alpha_2, \cdots, \alpha_{k-1}\}$
and the ith block $\{i, \alpha_1 + i, \alpha_2 + i, \cdots, \alpha_{k-1} + i\}$ where $i \in \mathbb{Z}_v$ and the additions are modulo v. Let $\alpha_n = \alpha_m + i$ where $\alpha_m, \alpha_n, i \in \mathbb{Z}_v$ and $0 \leq n, m \leq k - 1$. So we put α_n in the list

against i. In other words, α_n is one of the λ elements that are put in L_i. Next consider the p-th block $\{p,\ \alpha_1 + p,\ \alpha_2 + p,\ \cdots,\ \alpha_{k-1} + p\}$ where $p \in \mathbb{Z}_v$ and the $(p+i)$th block $\{(p+i),\ \alpha_1 + (p+i),\ \alpha_2 + (p+i),\ \cdots,\ \alpha_{k-1} + (p+i)\}$ where $(p+i) \in \mathbb{Z}_v$. Now let $\alpha_x + p = \alpha_y + p + i \Longrightarrow \alpha_x = \alpha_y + i$. This is obviously satisfied if $x = n$ and $y = m$. The common key is $\alpha_x + p = \alpha_y + p + i = p + \alpha_n = p + \alpha_m + i$. ■

Example 9 *Let us cite an example of the projective plane* $(13, 13, 4, 4, 1)$. *The blocks of the projective plane are as follows:*

0	1	3	9
1	2	4	10
2	3	5	11
3	4	6	12
4	5	7	0
5	6	8	1
6	7	9	2
7	8	10	3
8	9	11	4
9	10	12	5
10	11	0	6
11	12	1	7
12	0	2	8

We can easily index the blocks by the first elements of each block. It is easy to see that the i-th block is $\{i \bmod 13, (i+1) \bmod 13, (i+3) \bmod 13, (i+9) \bmod 13\}$. *Next we construct the table 8.2:*

Suppose we need to calculate the common key between the blocks (nodes) 7 *and* 11. *Node* 7 *at once calculates the difference* $11 - 7 = 4$ *and looks up table 8.2 against the difference value* 4. *The corresponding index of the key is* $i \bmod 13$. *Putting* $i = 7$, *it finds the index to be* 7. *On the other hand, node* 11 *calculates the difference* $7 - 11 = -4 = 9$, *since everything is calculated modulo* 13. *Node* 11 *looks up table 8.2 against the difference value* 9 *and finds that the corresponding index of the key is* $(i+9) \bmod 13$. *Putting* $i = 11$, *it finds the index to be* 7. *Hence the index of the mutually agreed upon key is* 7.

Example 10 *Let us cite an example of the biplane* $(11, 11, 5, 5, 2)$. *The blocks of the biplane are as follows:*

$Difference$	$Indices\ of\ the\ Keys$
1	$(i+1) \bmod 13$
2	$(i+3) \bmod 13$
3	$(i+3) \bmod 13$
4	$i \bmod 13$
5	$(i+1) \bmod 13$
6	$(i+9) \bmod 13$
7	$(i+3) \bmod 13$
8	$(i+9) \bmod 13$
9	$(i+9) \bmod 13$
10	$i \bmod 13$
11	$(i+1) \bmod 13$
12	$i \bmod 13$

Table 8.2: Pre-computed table of $Difference$ and $Value\ of\ the\ Key$.

0	2	3	4	8
1	3	4	5	9
2	4	5	6	10
3	5	6	7	0
4	6	7	8	1
5	7	8	9	2
6	8	9	10	3
7	9	10	0	4
8	10	0	1	5
9	0	1	2	6
10	1	2	3	7

We can easily index the blocks by the first elements of each block. It is easy to see that the i-th block is $\{i \bmod 11, (i+2) \bmod 11, (i+3) \bmod 11, (i+4) \bmod 11, (i+8) \bmod 11\}$. *Next we construct the table 8.3:*

Suppose we need to calculate the common key between the blocks 4 *and* 10. *Node* 4 *calculates the difference* $10 - 4 = 6$ *and looks up table 8.3 against the difference value* 6. *The corresponding indices of the keys are* $(i+3) \bmod 11$, $(i+8) \bmod 11$. *Putting* $i = 4$, *it finds the indices to be* 7 *and* 1. *Node* 10 *calculates the difference* $4 - 10 = -6 = 5$ *and looks up table 8.3 against the difference value* 5. *The corresponding indices of the keys are* $(i+2) \bmod 11$, $(i+8) \bmod 11$. *Putting* $i = 10$, *it finds the indices to be* 1 *and* 7. *Hence both the nodes agree upon the keys with indices* 1 *and* 7. *Now they have an option of selecting either of the two keys. Note that all the computations are done modulo* 11.

$Difference$	$Indices\ of\ the\ Keys$
1	$(i+3)$ mod 11, $(i+4)$ mod 11
2	$(i+2)$ mod 11, $(i+4)$ mod 11
3	$(i+3)$ mod 11, i mod 11
4	$(i+4)$ mod 11, $(i+8)$ mod 11
5	$(i+2)$ mod 11, $(i+8)$ mod 11
6	$(i+3)$ mod 11, $(i+8)$ mod 11
7	i mod 11, $(i+4)$ mod 11
8	i mod 11, $(i+8)$ mod 11
9	i mod 11, $(i+2)$ mod 11
10	$(i+2)$ mod 11, $(i+3)$ mod 11

Table 8.3: Pre-computed table of $Difference$ and $Value\ of\ the\ Key$.

Example 11 *We shall cite a final example of the projective plane* $(21, 21, 5, 5, 1)$. *The blocks of the projective plane are as follows:*

0	1	6	8	18
1	2	7	9	19
2	3	8	10	20
3	4	9	11	0
4	5	10	12	1
5	6	11	13	2
6	7	12	14	3
7	8	13	15	4
8	9	14	16	5
9	10	15	17	6
10	11	16	18	7
11	12	17	19	8
12	13	18	20	9
13	14	19	0	10
14	15	20	1	11
15	16	0	2	12
16	17	1	3	13
17	18	2	4	14
18	19	3	5	15
19	20	4	6	16
20	0	5	7	17

Difference	*Indices of the Keys*
1	$(i+1) \bmod 21$
2	$(i+8) \bmod 21$
3	$i \bmod 21$
4	$(i+1) \bmod 21$
5	$(i+6) \bmod 21$
6	$(i+6) \bmod 21$
7	$(i+8) \bmod 21$
8	$(i+8) \bmod 21$
9	$(i+6) \bmod 21$
10	$(i+18) \bmod 21$
11	$(i+8) \bmod 21$
12	$(i+18) \bmod 21$
12	$(i+18) \bmod 21$
13	$i \bmod 21$
14	$(i+1) \bmod 21$
15	$i) \bmod 21$
16	$(i+1) \bmod 21$
17	$(i+18) \bmod 21$
18	$(i+18) \bmod 21$
19	$(i+6) \bmod 21$
20	$i \bmod 21$

Table 8.4: Pre-computed Table of *Difference* and *Value of the Key.*

Index the blocks by the first elements of each block. It is easy to see that the i-th block is $\{i \bmod 21, (i+1) \bmod 21, (i+6) \bmod 21, (i+8) \bmod 21, (i+18) \bmod 21\}$. *Next we construct the table 8.4:*

Suppose we need to calculate the common key between the blocks 15 *and* 18. *For node* 18, *we at once calculate the difference* $15 - 18 = -3 = 18$ *(all the calculations are done modulo* 21*) and lookup table 8.4 against the difference value* 18. *The corresponding index of the key is* $(i + 18) \bmod 21$. *Putting* $i = 18$, *the index is readily found to be* 15. *For node* 15, *we at once calculate the difference* $18 - 15 = 3$ *and lookup table 8.4 against the difference value* 3. *The corresponding index of the key is* $i \bmod 21$. *Putting* $i = 15$, *the index is readily found to be* 15. *So the both the nodes agree upon the key with index* 15.

8.2.2 Another special case for designs generated from SBIBDs

If we generate PBIBDs by the Kronecker Product of the incidence matrix of two SBIBDs resulting from difference sets, then also we can calculate the indices of the keys common between any two nodes of the resulting sensor network. Consider two SBIBDs having parameters $(v_1, b_1 = v_1, r_1, k_1 = r_1, \lambda_1)$ and $(v_2, b_2 = v_2, r_2, k_2 = r_2, \lambda_2)$. Denote their incidence matrices by A and B respectively. Let $C = A \otimes B$. We can easily prepare one list each for both the SBIBDs, as shown in method 1. Let us denote those two lists by L_1 and L_2. Now the sensor network resulting from C will have $v_1 v_2$ blocks (nodes) and these nodes may be indexed by $xv_2 + y$ where $0 \leq x \leq v_1$ and $0 \leq y \leq v_2$. Consider two such nodes $x_1 v_2 + y_1$ and $x_2 v_2 + y_2$.

Method 2 *1. Find the common keys between x_1th row and x_2th row from L_1 using method 1. Denote them by $\gamma_1, \ \gamma_2, \ \cdots, \ \gamma_{\lambda_1}$*

2. Find the common keys between y_1 and y_2 from L_2 using method 1. Denote them by $\delta_1, \ \delta_2, \ \cdots, \ \delta_{\lambda_2}$

3. The indices of the common keys will be

$$\begin{array}{c} v_2\gamma_1 + \delta_1 - 1, \ v_2\gamma_1 + \delta_2 - 1, \ \cdots, \ v_2\gamma_1 + \delta_{\lambda_2} - 1, \\ v_2\gamma_2 + \delta_1 - 1, \ v_2\gamma_2 + \delta_2 - 1, \ \cdots, \ v_2\gamma_2 + \delta_{\lambda_2} - 1, \\ \cdots \ \cdots \ \cdots \ \cdots \ \cdots \ \cdots \ \cdots \ \cdots \ \cdots \ \cdots \\ v_2\gamma_{\lambda_1} + \delta_1 - 1, \ v_2\gamma_{\lambda_1} + \delta_2 - 1, \ \cdots, \ v_2\gamma_{\lambda_1} + \delta_{\lambda_2} - 1 \end{array}$$

Proof of Correctness of Method 2

Proof : There will be common 1s between the two rows of A at columns $\gamma_1, \ \gamma_2, \ \cdots, \ \gamma_{\lambda_1}$ and similarly there will be common 1s between the two rows of B at columns $\delta_1, \ \delta_2, \ \cdots, \ \delta_{\lambda_2}$. Next consider the dot product of the two rows $x_1 v_2 + y_1$ and $x_2 v_2 + y_2$ of $C = A \otimes B$. Each row may be thought of as v_1 blocks of length v_2 each. The ith block contains the columns $v_2 i$ to $v_2(i+1) - 1$, both inclusive, where $i = 0, 1, \cdots, v_1 - 1$. We are interested in the presence of common 1s between the aforesaid rows of C. Now the x_1th row and the x_2th row of A contain common 1s at $\gamma_1, \ \gamma_2, \ \cdots, \ \gamma_{\lambda_1}$ column positions and the y_1th row and the y_2th row of B contain common 1s at $\delta_1, \ \delta_2, \ \cdots, \ \delta_{\lambda_2}$ column positions. So there will be common 1s between the aforesaid rows of C at $\gamma_1, \ \gamma_2, \ \cdots, \ \gamma_{\lambda_1}$th blocks since all other blocks will contain only 0s. Again, in each block, there will be common 1s at $\delta_1, \ \delta_2, \ \cdots, \ \delta_{\lambda_2}$ positions. Hence the absolute indices of the columns containing the common 1s are

$$v_2\gamma_1 + \delta_1 - 1, \ v_2\gamma_1 + \delta_2 - 1, \ \cdots, \ v_2\gamma_1 + \delta_{\lambda_2} - 1,$$

$$v_2\gamma_2 + \delta_1 - 1,\ v_2\gamma_2 + \delta_2 - 1,\ \cdots,\ v_2\gamma_2 + \delta_{\lambda_2} - 1,$$
$$\cdots\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots$$
$$v_2\gamma_{\lambda_1} + \delta_1 - 1,\ v_2\gamma_{\lambda_1} + \delta_2 - 1,\ \cdots,\ v_2\gamma_{\lambda_1} + \delta_{\lambda_2} - 1$$

■

Example 12 *Let A be the incidence matrix of the projective plane $(v_1 = 13, b_1 = 13, r_1 = 4, k_1 = 4, \lambda'_1 = 1)$ and B be the incidence matrix of the biplane $(v_2 = 11, b_2 = 11, r_2 = 5, k_2 = 5, \lambda'_2 = 2)$. We take the Kronecker product of the incidence matrices of these two designs to generate the incidence matrix of a 3 associate class PBIBD with parameters $v = 143, b = 143, r = 20, k = 20, \lambda_1 = 2, \lambda_2 = 5, \lambda_3 = 8$. Suppose we want to find the common keys between the nodes 50 and 100. We first note that $x_1 = \lfloor\frac{50}{11}\rfloor = 4$, $y_1 = 50 \bmod 11 = 6$, $x_2 = \lfloor\frac{100}{11}\rfloor = 9$ and $y_2 = 100 \bmod 11 = 1$. So there are $\lambda_1 = 2$ common keys between these two nodes. From example 9, the common keys between 4 and 9 is $\gamma_1 = 5$ and from example 10, the common keys between 6 and 1 are $\delta_1 = 3$ and $\delta_2 = 9$. So the indices of the common keys are $v_2\gamma_1 + \delta_1 - 1 = 11 \cdot 5 + 3 - 1 = 57$ and $v_2\gamma_1 + \delta_2 - 1 = 11 \cdot 5 + 9 - 1 = 63$. The incidence matrices of example 9 and example 10 are shown below.*

$$A = \begin{pmatrix}
1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1
\end{pmatrix}$$

$$B = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

The dot product of the 50*th and the* 100*th rows of* C *comprises of two* 1*s at the* 57*th and* 63*rd column position and the rest* 141 *column positions are all* 0*s.*

8.3 Conclusion and Future Research

In this chapter, we have noted that many combinatorial designs can be used to give key distributions for sensor networks. We have studied the use of BIBDs to give a rather loose upper bound for *fail(s)* and also an approximate estimate. It will be a worthwhile exercise to calculate the value of this probability with more accuracy. We have explored in detail the use of SBIBDs for key distributions for sensor keys. We note that relatively speaking, SBIBDs are rare compared with other combinatorial designs.

We have given a general protocol for key agreement. Several special cases are also discussed for easier methods of key agreement when the underlying pre-distribution schemes result from difference sets (which are rarer than SBIBDs).

We are working on some other techniques of key pre-distribution using error correcting codes (n, m, d) where n corresponds to the total number of keys, m corresponds to the total number of sensor nodes and "the number of keys in each node" corresponds to the weighs of the codewords. For more details on these techniques, refer to [3, 35, 71, 84].

Chapter 9

Conclusion

We have plans for future research work as follows.

In chapter 4, we have presented a randomized block merging strategy. The resulting designs are readily available according to user requirements, but they are not directly available from combinatorial designs. The target is to get more than one common keys between any two nodes. A heuristic improvement of the basic randomized block merging strategy is presented for merging blocks in a (v, b, r, k) configuration (resulting from the $TD(k, r)$). It is done in such a manner that the blocks constituting a node will not share any common key among themselves. This provides better parameters than our basic design. It will be interesting to regularize the key pre-distribution after random merging. In the strategy presented in this chapter, the number of common keys between any two nodes follow binomial distribution. Thus, there is a probability (though very low) that there may be no common key between two nodes (for the time being, to get around this difficulty, two nodes can always communicate via an intermediate node with almost certainty). It looks promising to apply more sophisticated heuristic re-arrangement of blocks among the nodes available after the merging so that the number of common keys between any two nodes becomes more or less constant and at least one.

In chapter 5, we have studied the subset of nodes that are securely connected to each other (clique). We have studied the cliques corresponding to the (v, b, r, k) configuration (resulting from the $TD(k, r)$) where each block corresponds to a node. We also consider the scenario where more than one blocks are merged to form a node. We show that the clique size increases in such a scenario. An interesting future work in this area is to implement a merging strategy such that one can get cliques of maximum size after the merging.

In chapter 6, we have further investigated networks where users have differing resources and capacity requirements. One application involves a large network with large, mostly self-contained sub-networks, while another application involves networks with different robustness at different levels. For example, at the second level of hierarchy (i.e., the level containing the special nodes), one may need to have different number of common keys shared between two given nodes. It will be an interesting combinatorial problem to find out a design having such a property. One may even look for better alternatives compared to the use of copies of projective planes at this level.

In chapter 7, we have considered the case of random and independent failure of nodes. We have not specified any particular algorithm that determines which nodes will remain alive and which nodes will be put to sleep so that the longevity of the network increases. However, by increasing the number of compromised nodes, one can arrive at a configuration that can maintain a connected network and at the same time can cover all the grid points. It is easy to find two disjoint configurations with only half of the total nodes being live. One can assume that half of the nodes will be alive at any given point of time and the rest will sleep and conserve energy. In our future work, we shall give deterministic algorithms for sleeping of the sensor nodes.

In chapter 8, we have given an upper bound for *fail(s)* and also an approximate estimate. It will be a worthwhile exercise to calculate the value of this probability with more accuracy. We have given a general protocol for key agreement. Several special cases are also discussed for easier methods of key agreement when the underlying pre-distribution schemes result from difference sets. An interesting exercise will be to explore other combinatorial designs (linked designs in particular), which may also be used in designing sensor networks.

Bibliography

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci. *A survey on sensor networks,* IEEE Communications Magazine, vol. 40, no. 8, pp. 102–116, August, 2002.

[2] K. T. Arasu, C. Ding, T. Helleseth, P. V. Kumar and H. Martinsen. *Almost difference sets and their sequences with optimal autocorrelation,* IEEE Transactions on Information Theory, vol. 47, no. 7, pp. 2834-2943, 2001.

[3] E. F. Assmus and J. D. Key. *Designs and their codes,* Cambridge University Press, Cambridge University Press Tracts in Mathematics, no. 103, 1992.

[4] R. Blom. *An optimal class of symmetric key generation systems,* Advances in Cryptology: Proceedings of Eurocrypt 1984, LNCS 209, pp. 335–338, 1985.

[5] C. Blundo, A. Santis, A. Herzberg, S. Kutten, U. Vacarro and M. Yung. *Perfectly-secure key distribution for dynamic conferences,* Advances in Cryptology: Proceedings of Crypto 1992, LNCS 740, pp. 471–486, 1993.

[6] M. Bohge and W. Trappe. *An authentication framework for hierarchical ad hoc sensor networks,* Proceedings of the 2003 ACM workshop on Wireless security, pp. 79–87, 2003.

[7] B. Bollobas. *Random graphs,* Academic Press, pp. 336–346, 1985.

[8] D. Boneh and M. Franklin. *Identity-based encryption from the Weil Pairing,* Advances in Cryptology: Proceedings of Crypto 2001, LNCS 2139, pp. 213–229, 2001.

[9] R. C. Bose and K. R. Nair. *Partially balanced incomplete block designs,* Sankhya, vol. 4, pp. 337–372, 1939.

[10] R. C. Bose, S. S. Shrikhande and K. N. Bhattacharya. *On the Construction of Group Divisible Incomplete Block Designs,* The Annals of Mathematical Statistics, vol. 24, no. 2, pp. 167–195, June, 1953.

[11] M. Burmester and Y. Desmedt. *A secure and efficient conference key distribution system,* Advances in Cryptology: Proceedings of Eurocrypt 1994, LNCS 950, pp. 275–286, 1995.

[12] S. Camtepe and B. Yener. *Combinatorial design of key distribution mechanisms for wireless sensor networks,* Proceedings of Esorics 2004, LNCS 3193, pp. 293–308, 2004.

[13] S. Capkun and J. P. Hubaux. *Secure positioning in wireless networks,* IEEE journal on selected areas in communications, vol. 24, no. 2, pp. 221–232, February, 2006.

[14] S. Capkun and J. P. Hubaux. *Secure positioning of wireless devices with application to sensor networks,* Proceedings of IEEE INFOCOM, pp. 1–13, 2005.

[15] D. Carman, B. Matt and G. Cirincione. *Energy-efficient and low-latency key management for sensor networks,* Proceedings of the 23rd Army Science Conference, Paper no. OO-03, pp. 1–8, December, 2002.

[16] D. Chakrabarti, S. Maitra and B. K. Roy. *Clique size in sensor networks with key pre-distribution based on transversal design,* International Journal of Distributed Sensor Networks, vol. 1, Issues 3-4, pp. 345–354, 2005.

[17] D. Chakrabarti, S. Maitra and B. K. Roy. *A key pre-distribution scheme for wireless sensor networks: merging blocks in combinatorial design,* Proceedings of 8th Information Security Conference, ISC 2005, LNCS 3650, pp. 89–103, 2005.

[18] D. Chakrabarti, S. Maitra and B. K. Roy. *A hybrid design of key pre-distribution scheme for wireless sensor networks,* Proceedings of 1st International Conference on Information Systems Security, ICISS 2005, LNCS 3803, pp. 228–238, 2005.

[19] D. Chakrabarti, S. Maitra and B. K. Roy. *Clique size in sensor networks with key pre-distribution based on transversal design,* Proceedings of 7th International Workshop on Distributed Computing, IWDC 2005, LNCS 3741, pp. 329–337, 2005.

[20] D. Chakrabarti and J. Seberry. *Combinatorial structures for design of wireless sensor networks,* Proceedings of 4th International Conference on Applied Cryptography and Network Security, ACNS 2006, LNCS 3989, pp. 365–374, 2006.

[21] D. Chakrabarti, S. Maitra and B. K. Roy. *A key pre-distribution scheme for wireless sensor networks: merging blocks in combinatorial design,* International Journal of Information Security, vol. 5, Issue 2, pp. 105–114, April, 2006.

[22] H. Chan, A. Perrig and D. Song. *Random key pre-distribution schemes for sensor networks,* Proceedings of IEEE Symposium on Security and Privacy, pp. 197–213, May, 2003.

[23] M. Chen, W. Cui, V. Wen and A. Woo. *Security and deployment issues in a sensor network,* Ninja Project: A Scalable Internet Services Architecture, Berkeley, pp. 1–11, 2000.

[24] C. J. Colbourn and J. H. Dinitz. *The CRC handbook of combinatorial designs,* CRC Press, Boca Raton, 1996.

[25] B. Deb, S. Bhatnagar and B. Nath. *RelnForM: Reliable information forwarding using multiple paths in sensor networks,* Proceedings of 28th Annual IEEE International Conference on Local Computer Networks, LCN 2003, pp. 406–415, October, 2003.

[26] J. Deng, R. Han and S. Mishra. *Enhancing base station security in wireless sensor networks,* Technical Report CU-CS-951-03, Department of Computer Science, University of Colorado, April, 2003.

[27] J. Deng, R. Han and S. Mishra. *A performance evaluation of intrusion-tolerant routing in wireless sensor networks,* Proceedings of IEEE 2nd International Workshop on Information Processing in Sensor Networks, IPSN 2003, pp. 349–364, 2003.

[28] C. Ding and J. Yuan. *A family of skew Hadamard difference sets,* Journal of Combinatorial Theory, Series A, vol. 113, pp. 1526-1535, 2006.

[29] J. R. Douceur. *The Sybil attack,* Proceedings of the IPTPS02 Workshop, Cambridge, MA (USA), LNCS 2429, pp. 251–260, March, 2002.

[30] W. Du, J. Ding, Y. Han, S. Chen and P. Varshney. *A key management scheme for wireless sensor networks using deployment knowledge,* Proceedings of IEEE Infocom04, pp. 586–597, 2004.

[31] W. Du, J. Ding, Y. Han and P. Varshney. *A pair-wise key pre-distribution scheme for wireless sensor networks,* Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS '03, pp. 42–51, 2003.

[32] B. Dutertre, S. Cheung and J. Levy. *Lightweight key management in wireless sensor networks by leveraging initial trust,* Technical Report SRI-SDL-04-02, System Design Laboratory, April, 2002.

[33] J. Elson, L. Girod and D. Estrin. *Fine-grained network time syncronization using reference broadcasts,* Proceedings of the 5th symposium on Operating systems design and implementation, vol. 36, Issue SI (Winter 2002), pp. 147–163, 2002.

[34] L. Eschenauer and V. D. Gligor. *A key management scheme for distributed sensor networks,* Proceedings of the 9th ACM conference on computer and communications security, pp. 41–47, November, 2002.

[35] S. Foldes and N. M. Singhi. *On instantaneous codes,* Rutcor Research Report 44-2004, November, 2004.

[36] D. Ganesan, R. Govindan, S. Shenker and D. Estrin. *Highly resilient, energy-efficient multipath routing in wireless sensor networks,* Mobile Computing and Communications Review, vol. 5, no. 5, pp. 11–25, October, 2001.

[37] M. Grossgluauser and D. Tse. *Mobility increases the capacity of ad-hoc wireless networks,* Proceedings of IEEE INFOCOM 2001, Anchorage, Alaska, pp. 1360–1369, April, 2001.

[38] P. Gupta and P. R. Kumar. *Critical power for asymptotic connectivity in wireless networks,* Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W. H. Fleming, pp. 547–566, 1998.

[39] P. Gupta and P. R. Kumar. *The capacity of wireless networks,* IEEE Transactions on Information Theory, vol. 46, no. 2, pp. 388–404, March, 2000.

[40] N. Gura, A. Patel, A. Wander, H. Eberle and S. C. Shantz. *Comparing elliptic curve cryptography and RSA on 8-bit CPUs,* Proceedings of the 2004 Workshop on Cryptographic Hardware and Embedded Systems, CHES 2004, LNCS 3156, pp. 119–132, 2004.

[41] P. Hall. *Introduction to the theory of coverage processes,* John Wiley and Sons, 1988.

[42] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, Luo L. Liqian, R. Stoleru, T. Yan and L. Gu. *Energy-efficient surveillance system using wireless sensor networks,* Proceedings of the 2nd international conference on Mobile systems, applications and services, pp. 270–283, 2004.

[43] S. Hedetniemi, S. Hedetniemi and A. Liestman. *A survey of gossipping and broadcasting in communication networks,* Networks, vol. 18, pp. 319–349, 1988.

[44] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan. *Energy-efficient communication protocol for wireless microsensor networks,* Proceedings of the 33rd Hawaii International Conference on System Sciences, vol. 8, pp. 1–10, January, 2000.

[45] W. R. Heinzelman, J. Kulik and H. Balakrishnan. *Adaptive protocols for information dissemination in wireless sensor networks,* Proceedings of the ACM MobiCom '99, Seattle, WA, pp. 174–185, 1999.

[46] Y. C. Hu, A. Perrig and D. B. Johnson. *Ariadne: A secure on-demand routing protocol for ad hoc networks,* Proceedings of the MobiCom 2002, pp. 12–23, September, 2002.

[47] Y. C. Hu, A. Perrig and D. B. Johnson. *Packet leashes: a defense against wormhole attacks in wireless networks,* Proceedings of IEEE INFOCOM 2003, pp. 1976–1986, April, 2003.

[48] C. Huang and Y. Tseng. *The coverage problem in a wireless sensor network,* Proceedings of WSNA, San Diego, CA, pp. 115–121, 2003.

[49] D. Huang, M. Mehta, D. Medhi and L. Harn. *Location aware key management scheme for wireless sensor networks,* Proceedings of the 2nd ACM workshop on Security of Ad Hoc and Sensor Networks, SASN 2004, pp. 29–42, 2004.

[50] J. Hui, Z. Ren and B. H. Krogh. *Sentry-based power management in wireless sensor networks,* Proceedings of IPSN, Palo Alto, CA, pp. 458–472, 2003.

[51] J. Hwang and Y. Kim. *Revisiting random key pre-distribution for sensor networks,* Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks, SASN 2004, pp. 43–52, 2004.

[52] C. Intanagonwiwat, R. Govindan and D. Estrin. *Directed diffusion: A scalable and robust communication paradigm for sensor networks,* Proceedings of the ACM MobiCom'00, Boston, MA, pp. 56–67, 2000.

[53] J. M. Kahn, R. H. Katz and K. S. J. Pister. *Next century challenges: mobile networking for smart dust,* Proceedings of the ACM MobiCom '99, Washington, DC, pp. 271–278, 1999.

[54] C. Karlof, N. Sastry and D. Wagner. *Tinysec: Link layer security for tiny devices,* Proceedings of the 2nd international conference on Embedded networked sensor systems, SensSys 2004, pp. 162–175, November, 2004.

[55] C. Karlof and D. Wagner. *Secure routing in wireless sensor networks: attacks and countermeasures,* Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols, vol. 1, Issues 2–3, pp. 293–315, September, 2003.

[56] B. Karp and H. T. Kung. *GPSR: Greedy perimeter stateless routing for wireless networks,* Proceedings of 6th annual international conference on Mobile computing and networking, MobiCom 2000, pp. 243–254, 2000.

[57] A. M. Kermarrec, L. Massoulie and A. Ganesh. *Reliable probabilistic communication in large-scale information dissemination systems,* Microsoft Research Tech Report MSR-TR-2000-105, October, 2000.

[58] A. M. Kermarrec, L. Massoulie and A. Ganesh. *Scamp: Peer-to-peer lightweight membership service for large-scale group communication,* Proceedings of the Third International Workshop on Networked Group Communications, NGC 2001, London, UK, pp. 44–55, November, 2001.

[59] S. Kumar, T. H. Lai and J. Balogh. *On k-Coverage in a mostly sleeping sensor network,* Proceedings of the 10th annual international conference on Mobile computing and networking MobiCom'04, Philadelphia, Pennsylvania, USA, pp. 144–158, 2004.

[60] B. Lai, S. Kim and I. Verbauwhede. *Scalable session key construction protocol for wireless sensor networks,* IEEE Workshop on Large Scale Real Time and Embedded Systems, LARTES, pp. 1–6, 2002.

[61] C. W. H. Lam. *The search for a finite projective plane of order 10,* American Mathematical Monthly, vol. 98, pp. 305–318, 1991.

[62] L. Lamport, R. Shostak and M. Pease. *The byzantine generals problem,* ACM Transactions on Programming Languages and Systems, vol. 4, no. 3, pp. 382–401, July, 1982.

[63] Y. Law, R. Corin, S. Etalle and P. Hartel. *A formally verified decentralised key management for wireless sensor networks,* 4th IFIP TC6/WG6.8 International Conference on Personal Wireless Communications, PWC 2003, LNCS, vol. 2775, pp. 27-39, 2003.

[64] J. Lee and D. Stinson. *Deterministic key pre-distribution schemes for distributed sensor networks,* Proceedings of SAC 2004, LNCS 3357, pp. 294–307, 2004.

[65] J. Lee and D. Stinson. *A combinatorial approach to key pre-distribution for distributed sensor networks,* IEEE Wireless Computing and Networking Conference, WCNC 2005, New Orleans, LA, USA, vol.2, pp. 1200–1205, 2005.

[66] L. Li and J. Y. Halpern. *Minimum energy mobile wireless networks revisited,* Proceedings of IEEE International Conference on Communications, ICC 2001, Helsinki, Finland, vol.1, pp. 278–283, June, 2001.

[67] D. Liu and P. Ning. *Establishing pairwise keys in distributed sensor networks,* Proceedings of the 10th ACM Conference on Computer and Communication Security, pp. 52–61, 2003.

[68] D. Liu and P. Ning. *Location-based pairwise key establishment for static sensor networks,* 1st ACM Workshop on Security of Ad Hoc and Sensor Networks, SASN 2003, pp. 72-82, 2003.

[69] D. Liu and P. Ning. *Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks,* 10th Annual Network and Distributed System Security Symposium, pp. 263-276, 2003.

[70] D. Liu and P. Ning. *Multi-level μTESLA: A broadcast authentication system for distributed sensor networks,* ACM Transactions on Embedded Computing Systems, TECS, vol. 3, no. 4, pp. 800–836, 2004.

[71] F. J. MacWilliams and N. J. A. Sloane. *The theory of error-correcting codes, Part I, II,* North Holland Publishing Company, 1977.

[72] R. Meester and R. Roy. *Continuum percolation,* Cambridge University Press, 1996.

[73] R. Merkle. *Secure communication over insecure channels,* Communications of the ACM, vol. 21, no. 4, pp. 294–299, 1978.

[74] J. Newsome, E. Shi, D. Song and A. Perrig. *The Sybil attack in sensor networks: analysis and defenses,* Proceedings of the IEEE International Conference on Information Processing in Sensor Networks, pp. 259–268, April, 2004.

[75] R. Nishi and K. Sakurai. *Group key distribution scheme for reducing required rekey message size,* Proceedings of the 11th International Conference on Parallel and Distributed Systems, vol. 2, Issue 20-22, pp. 280–284, July, 2005.

[76] A. Perrig, R. Szewczyk, V. Wen, D. Culler and J. Tygar. *SPINS: Security protocols for sensor networks,* Wireless Networks Journal, vol. 8, no. 5, pp. 521–534, 2002.

[77] A. Perrig, R. Canetti, J. Tygar and D. X. Song. *Efficient authentication and signing of multicast streams over lossy channels,* IEEE Symposium on Security and Privacy, pp. 56–73, 2000.

[78] R. L. Pickholtz, D. L. Schilling and L. B. Milstein. *Theory of spread spectrum communications: a tutorial,* IEEE Transactions on Communications, vol. COM-30, no. 5, 1982, pp. 855–884, 1982.

[79] R. Pietro, L. Mancini and A. Mei. *Random key assignment secure wireless sensor networks,* Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks, pp. 62–71, 2003.

[80] G. J. Pottie and W. J. Kaiser. *Wireless integrated network sensors,* Communications of the ACM, vol. 43, no. 5, pp. 551–558, 2000.

[81] B. Przydatek, D. Song and A. Perrig. *SIA: Security information aggregation in sensor networks,* Proceedings of the 1st ACM International Conference on Embedded Networked Sensor Systems, pp. 255–265, 2003.

[82] V. Rodoplu and T. H. Meng. *Minimum energy mobile wireless networks,* IEEE JSAC, vol. 17, no. 8, pp. 1333–1344, 1999.

[83] N. Sastry, U. Shankar and D. Wagner. *Secure verification of location claims,* Proceedings of the ACM Workshop on Wireless Security, pp. 1–10, September, 2003.

[84] N. Sendrier. *Finding the permutation between equivalent linear codes: The support splitting algorithm,* IEEE Transactions on Information Theory, vol. 46, no. 4, pp. 1193–1203, 2000.

[85] Y. Shaked and A. Wool. *Cracking the Bluetooth PIN,* Proceedings of the 3rd USENIX/ACM Conference on Mobile Systems, Applications and Services, MobiSys, Seattle, WA, pp. 39–50, 2005.

[86] A. Shamir. *Identity-based cryptosystems and signature schemes,* Advances in Cryptology: Proceedings of Crypto 1984, LNCS 196, pp. 47–53, 1984.

[87] C. Shen, C. Srisathapornphat and C. Jaikaeo. *Sensor information networking architecture and applications,* IEEE Personal Communications, pp. 52–59, 2001.

[88] E. Shi and A. Perrig. *Designing secure sensor networks,* IEEE Wireless Communications, vol. 11, Issue 6, pp. 38–43, December, 2004.

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

[90] S. Sinha and A. Chandrakasan. *Dynamic power management in wireless sensor networks,* IEEE Design & Test of Computers, vol. 18, no. 2, pp. 62–74, March/April, 2001.

[91] S. Slijepcevic, M. Potkonjak, V. Tsiatsis, S. Zimbeck and M. Srivastava. *On communication security in wireless ad-hoc sensor network,* Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2002, pp. 139–144, 2002.

[92] K. Sohrabi, J. Gao, V. Ailawadhi and G. J. Pottie. *Protocols for self-organization of a wireless sensor network,* IEEE Personal Communications, vol. 7, no. 5, pp. 16–27, 2000.

[93] J. Staddon, D. Balfanz and G. Durfee. *Efficient tracing of failed nodes in sensor networks,* 1st ACM International Workshop on Wireless Sensor Networks and Applications, WSNA, pp. 122–130, 2002.

[94] M. Steiner, G. Tsudik and M. Waidner. *Key agreement in dynamic peer groups,* IEEE Transactions on Parallel and Distributed Systems, vol. 11, no. 8, pp. 769–780, 2000.

[95] D. Stinson. *Cryptography: Theory and practice (Second Edition),* Chapman & Hall, CRC Press, 2002.

[96] D. Stinson. *Combinatorial designs: Constructions and analysis,* Springer, New York, 2003.

[97] A. P. Street and D. J. Street. *Combinatorics of experimental design,* Clarendon Press, Oxford, 1987.

[98] J. Undercoffer, S. Avancha, A. Joshi and J. Pinkston. *Security for sensor networks,* CADIP Research Symposium, pp. 1–11, 2002.

[99] M. N. Vartak. *On an application of Kronecker product of matrices to statistical designs,* The Annals of Mathematical Statistics, vol. 26, no. 3, pp. 420–438, 1955.

[100] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless and C. Gill. *Integrated coverage and connectivity configuration in wireless sensor networks,* Proceedings of ACM Sensys, Los Angeles, CA, pp. 28–39, 2003.

[101] A. Woo and D. Culler. *A transmission control scheme for media access in sensor networks,* Proceedings of ACM MobiCom '01, Rome, Italy, pp. 221–235, 2001.

[102] A. D. Wood and J. A. Stankovic. *Denial of service in sensor networks,* IEEE Computer, vol. 35, no. 10, pp. 54–62, October, 2002.

[103] S. Zhu, S. Setia and S. Jajodia. *Leap: Efficient security mechanisms for large-scale distributed sensor networks,* 10th ACM Conference on Computer and Communications Security, CCS 2003, pp. 62–72, 2003.

[104] URL: http://www.geocities.com/dibyenduc123/sortedpbibd.txt