Application of Combinatorial Structures to Key Predistribution in Sensor Networks and Traitor Tracing

Thesis submitted to Indian Statistical Institute

Sushmita Ruj
Applied Statistics Unit
Indian Statistical Institute
Kolkata
February, 2009

Application of Combinatorial Structures to Key Predistribution in Sensor Networks and Traitor Tracing

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

by

Sushmita Ruj
Applied Statistics Unit
Indian Statistical Institute
Kolkata, India
email: sush_r@isical.ac.in

under the supervision of

Prof. Bimal Roy
Applied Statistics Unit
Indian Statistical Institute
Kolkata, India
email: bimal@isical.ac.in

Curiouser and curiouser!
-Alice's Adventures in Wonderland

Acknowledgements

It was always my dream to do research and seeing my thesis complete fills me with immense pleasure. Earlier whenever I had imagined to write my thesis, I had thought of thanking everyone who have helped knowingly or unknowingly in my efforts. I am lucky that I had Prof. Bimal Roy (Bimal da) as my thesis advisor. It was he who taught me design theory. For most of the time during my Ph.D. work he was the Dean of Studies at ISI. In spite of his busy schedule he always found time to discuss with me. Bimal da never imposed anything on me and let me work on anything I wanted to (even if it was not a part of my thesis) as long as I was having fun. He was been the greatest source of strength for me and encouraged in all my efforts. He also created the opportunity for my academic visits so that I could interact with many brilliant researchers. I'm not exaggerating when I say that my Ph.D. years wouldn't have been so smooth and fun filled had he not been my advisor. A trillion thanks to you, Bimal da.

I would like to thank Prof. Palash Sarkar who is a wonderful teacher. He taught me many fundamental theory courses during my M.Tech days in ISI. The discussions with Dr. Subhamoy Maitra was specially enlightening. I also thank Prof. Rana Barua for taking a lot of interest in my work and giving me important suggestions. I thank them for creating a very nice and light atmosphere in the group. They were always encouraging and supportive. No doubt I had the best time of my life during my Ph.D.

I am thankful to Prof. S.B. Rao with whom I worked on my M.Tech. dissertation. He taught me graph theory and encouraged me to learn new things. I thank Prof. A.R. Rao and Prof. K. S. Vijayan who were great teachers. Thanks to Prof. Mausumi Basu for her comments, suggestions and discussions. I would like to thank all my teachers for their support and encouragement.

During my Ph.D. I had the privilege to meet and discuss with several great minds. I would like to thank Prof. Jennifer Seberry for discussions and suggestions. I had a great time with her in Wollongong. She was a constant source of inspiration during my entire Ph.D. period. I also had the opportunity to discuss with Prof. Reihaneh Safavi-Naini. Her suggestions greatly helped me to improve my work. I would like to express my gratitude to Prof. Ronald Mullin for discussing a lot on design theory and suggesting useful references. I also had the scope of interacting with Prof. Nicolas Sendrier, whose discussion on coding theory was very insightful. I thank Prof. Amiya Nayak for his encouragement. I also thank all the anonymous reviewers of my papers. Their comments were very valuable in improving my work.

Though I had dreamt of doing research even when I was in school, all hopes seemed to have dampened when I joined engineering. It was then that I met Soumya Das and I became interested in combinatorics. He was a great source of encouragement and inspiration for the years to come. His long mails were always full of brain teasers, mathematical problems, interesting anecdotes and encouraging words. He was always patient with me and answered all my questions,

even if they were very trivial and sometimes stupid. A million thanks to you, Soumya for being there for me during those difficult times.

I had the most wonderful time in CRG-lab in ISI. I got some good friends who always created a light hearted atmosphere and made research enjoyable. I would like to thank Avishek da for the interesting discussions. Thanks to Kishan da, Sumanta da, Deepak da, Prem da and Rishi for their help and encouragement. We had all sorts of crazy discussions. I really had a great time with them specially during tea. I would like to thank Shashwat, Anupama di and Goutam da for proof-reading large portions of my thesis. I would also like to thank Somitra da, Santanu, Srimanta da, Mahavir da, Sumit... for their encouragement and enthusiasm which made research enjoyable and fun filled one.

I also had a great time in the hostel. I found some very good friends, who made my life full. I would like to thank Pallavi who was there from the very first day I came to ISI. I have spent some of the lovely evenings with her. I would like to thank Rajshree di who was there during the best times and the worst and Vasudha for her enthusiasm. I would like to thank Sharmila di, Moumita and Sulagna for being the best neighbors one could ever get. I also thank all those who worked for the hostel library. It has some of the greatest novels, which never made me feel bored or tired. I made some of my best friends in School. Some of them are still my closest friends, always encouraging me. Thanks to them all for rejoicing in my success and being there during hard times.

I have been blessed with wonderful parents who have been there for me through thick and thin. Though my father was a bit opposed to research, he was extremely supportive and encouraging during my Ph.D. days. Perhaps the seeds of research was sown by my mother when she used to tell bed-time stories about Madam Curie when I was just a kid. I thank both of them for their endless love and the confidence they have on me. I also had my wonderful *didibhai* (sister) with me always, pointing out my follies and taking pleasure in my success. Also I had my little niece, Reese with all her lovely antics. It was really fun seeing her grow.

I have drawn inspiration from people about whom I have heard or read about in books, newspapers and magazines. Thanks to them all. A big thanks to all those who kept me cheerful and made this journey most joyful.

Indian Statistical Institute, Kolkata *Sushmita Ruj*
February, 2009

Contents

List of Figures

List of Tables

Chapter 1

Introduction

From time immemorial people have used cryptography for secure communication. Cryptography has decided the fate of many kings and queens as well as the outcome of various wars. It has become all the more important nowadays, with the growing need of electronic communication. We now live in a society where electronic networks pervade all aspects of our professional and private lives. We all use cryptography when we do bank transfers by ATM, or SWIFT, while using the mobile phones and i-phones, SSL protocols or using password protected machines. Though electronic communication is fast and easy, it is vulnerable to security breeches, which are not so widespread in the traditional counterparts. For example we consider writing a letter. A hand-written letter is normally posted in a sealed envelop and delivered to a specified address by the postal service. To read the contents of the letter, the adversary must break open the seal. The recipient can inspect the envelope for damage and can make out if the letter was tampered with by simply inspecting the postmark or the handwriting. However an email is not secure. It passes through a large network of computers. At each router there is a chance of being intercepted. The contents may be read, altered, copied. The recipient has no idea of whether the information has been intercepted. Same situation arises when we transfer money by ATMs, or SWIFT transfers.

Cryptography is essentially a collection of mathematical techniques, algorithms and protocols used to render the core information assurance services that are required in electronic communications. The chief services of any cryptographic protocol are *confidentiality* (hiding the data from the unauthorized users), *data integrity* (protecting data from manipulation), *authentication*(verifying the identity), *non-repudiation*(preventing malicious users to hide their activity).

Most cryptographic mechanisms rely on a basic components called *keys*, which are essentially numbers selected at random from a large space. The security of cryptographic mechanisms rely on the security of the keys contained therein. Since keys form the building block of many cryptographic systems, the secure management of keys form an essential building block of many cryptosystems. The phrase *key management* describes the entire life cycle of keys, including *key generation*,

methods by which it is sent to the relevant users of the system (*key establishment*), the techniques that are used to change and refresh it, also called *key update* and ultimately the means by which it is deleted at the end of its usage called *key destruction*.

According to the nature of keys, cryptographic mechanisms can be divided into two broad categories- *symmetric* mechanisms, in which the same secret key is used for encryption by the sender and decryption by the receiver, and *public-key* mechanisms in which the sender encrypts the information using a public key. However to decrypt the contents, the receiver has to use her own private keys. There are some mechanisms where both symmetric and public-key mechanisms are used together. Generally public key mechanisms involve huge computational costs. Hence in areas of applications where resources are constrained, symmetric key mechanisms are preferred. One such area of application is *sensor networks*. Before discussing about sensor networks we look at the key establishment problem in general and key predistribution in particular.

1.1 Key establishment

The term *key establishment* is used to indicate the framework that primarily covers aspects of key management processes directly related to ensuring that the right keys are established in the right places within the network [101]. It is assumed that there is a central authority, also called the *trusted authority* (TA) which is regarded to be secure and can be relied on by all users in the network. Suppose there are N users in the network. The set of users is denoted by $\mathcal{U} = \{U_1, U_2, \ldots, U_N\}$ and the TA by $\mathcal{T}$. Let $\mathcal{B}$ denote the *communication structure*, that consists of a collection of users who wish to communicate with each other. We note here that many key management schemes, like that of sensor networks assume that cryptographic keys only need to be established between pairs of users. We refer the common key to each member of the set A to be K_A where $A \in \mathcal{B}$ is a group. In case of sensor networks we are mainly interested in finding pairwise keys, that is keys shared by any pair of sensor nodes.

The *key establishment scheme* for any communication structure $\mathcal{B}$ is a set of protocols that allow a set of $A \in \mathcal{B}$ to establish a group key k_A. It consists of the following phases.

1. Initialization: In this step each user U_i is assigned a secret data u_i. This value is known only to $\mathcal{T}$and U_i. It is assumed that there exists some secure channel though which this secret data is assigned. On receiving u_i, user U_i is responsible for protecting the value of u_i. A system-wide public data Pub is also assigned to every user in the network.

2. Key establishment: In this phase a group of users $A \in \mathcal{B}$ establish their common group key k_A. Depending on the way this process is executed with or

without the help of the TA or the other users in the group, key establishment can be of three types which are highlighted in the next section.

3. Update: This is an optional phase which takes places if the secret or the public data is modified. This occurs when the communication structure changes. The most important operation is key refreshment, where existing group keys are simply replaced by new keys.

We just said above that there are three types of key establishment mechanisms. We now discuss them in detail.

1.1.1 Broad classification of key establishment schemes

A major way of classifying different key establishment schemes is the extent to which the different identities in the network interact with each other. For example if the secure channel between the TA and users is costly, then the users may themselves try to execute the key establishment process. Otherwise they may fall back on the TA for key establishment. There are broadly three types of key establishment mechanisms.

1. Group key predistribution schemes: Users have no secure communication channels available to support key establishment and hence do that on their own. Group key predistribution schemes require involvement of an online TA during the update phase.

2. Group key distribution schemes: The TA has some ability to communicate with other users during the key establishment phase.

3. Group key agreement schemes: Users have some ability to communicate with one another during the key establishment phase.

Throughout the thesis we discuss key predistribution and key distribution schemes. We will see later why key predistribution and key distribution is best suited for our applications to sensor network and traitor tracing.

1.1.2 Other ways of classifying key establishing schemes

Given below are other parameters which distinguish different key establishment schemes.

1.1.2.1 Security

The main threat for key establishment is the ability of users to obtain a key they are not entitled to. Another threat is an outsider's access to the system by means of accessing some of the keys. In such cases the unauthorized user may modify, delete or forge some keys and play havoc with the existing network. There are two different aspects of security that needs to be identified.

1. **Types of security** The two most common types of security that are encountered in general are:
    - *Unconditional security*: the security is independent of the resources available to the attacker.
    - *Computational security*: The scheme can be broken if the attacker has sufficient resources.
2. **Resilience**: This specifies the degree to which a system is affected when users collude and share their data and give access to unauthorized users. It also means how much the system is tolerant to outsider attacks, which can compromise the keys possessed by the users. We will refer to the collection $\mathcal{X}$ of users $\mathcal{U}$ who even if they collude and share their data, are unable to obtain any group of keys they are not entitled to know. Based on this there are two different degrees of resilience.
    - *Full collusion security*: $\mathcal{X}$ consists of all subsets of $\mathcal{U}$, implying that no collusion of users should be able to determine a key that they are not authorized to know.
    - *c-security*: $\mathcal{X}$ consists of all subsets of $\mathcal{U}$ of size at most c, meaning that no collusion of up to c users should be able to determine a key that they are not entitled to.

1.1.2.2 Deterministic vs Probabilistic

This says whether the key establishment schemes are:

1. **Deterministic**: We can guarantee that a group $A \in \mathcal{B}$ is able to establish a common key.
2. **Probabilistic**: A group $A \in \mathcal{B}$ is able to establish a common key with certain probability.

In this thesis we will discuss deterministic schemes of key establishment.

1.1.2.3 Communication channels

There are basically two types of channels used in communication.

1. **Secure**: It is assumed that information exchanged through this channel is perfectly secure, ie, it is both confidential and authentic.
2. **Broadcast**: The message sent through this channel is received the same by all the users.

In this thesis we will use broadcast channel.

1.1.2.4 Properties of keys

This concerns the nature and structure of group keys. Group key k_A established may be

1. **Predistributed**: If k_A can be computed from only the values of the secret keys $\{u_i | U_i \in A\}$ and Pub.
2. **Independent**: If the knowledge of other group keys provide no information about the value of k_A.
3. **Combinatorial**: If k_A can be represented as subset of the collective secret user data of users belonging to group A.

1.1.2.5 Other capabilities

Other properties need special mention.

1. **Flexibility**: The extent to which a key establishment scheme is able to efficiently accommodate to perform an update phase.
2. **Computational capability**: The extent to which entities have ability to perform computations.
3. **Decentralization**: Whether the roles conducted by the TA would be distributed or performed by a single TA. This can be because of scalability, security and reliability.
4. **Collaboration**: The degree of collaboration required to take place between users in order to establish a group key.
5. **Robustness**: A stronger security model might be required for applications where either the TA or the users are not trusted to perform their operations honestly.
6. **Temporal restrictions**: Whether key establishment for certain groups are restricted to specific time intervals or limited to a finite number of key establishment events.
7. **Traceability**: Whether it is possible to identify pirate users who collude and give access to unauthorized users.

1.1.3 Evaluation metrics

There are several evaluation metrics that allow us to evaluate a key establishment scheme.

1. **Scalability**: Whether the system can support additional users without altering the keys already held by the existing users.

2. **Secret storage**: The amount of information that needs to be secretly stored. We would want to minimize this quantity. Later in the thesis we will see that it is a very important metric in resource constrained devices.

3. **Public storage**: The public information that needs to be stored. Generally it is not a very serious concern to reduce this storage.

4. **Computational costs**: We would like to minimize the costs needed for key establishment. Efficient computation is particularly important in low-memory device which we will see later in this section.

5. **Communication overheads**: The quantity of data that needs to be exchanged either in secure channels or in the form of broadcasts which we need to minimize.

A good key establishment scheme has high scalability, low storage, low computational and communicational costs.

We now discuss the two problems of sensor networks and traitor tracing. We will see that at the heart of both the problems lies efficient key predistribution techniques.

1.2 Sensor networks

Recently there has been a lot of research in the field of sensor network. This is motivated by the fact that sensor devices have a wide variety of application both in military as well as civilian areas. They are used to collect magnetic, seismic and acoustic information and sense the temperature and pressure of a given region. They are used in wildlife exploration, ocean water monitoring, plantation monitoring, cold chain management, rescue operation, vital sign monitoring and other civilian purposes [25, 130]. They are widely used in military areas for tracking military vehicles, sniper localization, collecting and transmitting secure information. Extensive surveys can be found in [2, 19, 172].

Sensor nodes are very small devices with limited memory, battery power, bandwidth, transmission range and computation power which are scattered in large numbers in the target region and work unattended for large periods of time. For more discussion on the resource constrained nature of sensor networks one may refer to [17, 129]. Since they are deployed in large numbers, their cost have to be minimized. An example of a typical sensor is the MICAz mote, that has an under powered processor with 4KB of RAM, 512 KB of program memory, an Advanced Encryption Standard (AES) cryptographic hardware and run the TinyOS [1] operating system. The transmitter of the UC Berkley Mica platform has bandwidth

of 10Kbps. It may not always be possible to know the exact position of the sensor nodes. However there have been instances of schemes which use deployment knowledge. We will address them later in Section 2.8 and Chapters 5 and 6.

The data in the sensor nodes deployed in military areas and also in health care or commercial purposes needs to be securely transmitted. The interception of such data can cause havoc and must thus be prevented. Security in Wireless Sensor Networks (WSN) has six challenges [19].

1. Wireless nature of communication
2. Resource limitation on sensor nodes
3. Very large and dense WSN
4. Lack of fixed infrastructure
5. Unknown network topology prior to deployment
6. High risk of physical attacks to unattended sensors.

For the above security reasons cryptographic keys must be embedded in the sensor nodes which can carry on communication securely. Hence key management becomes of utmost importance in sensor networks.

Sensor networks must arrange several types of data packets, including packets of routing protocols and packets of key management protocols. The key establishment techniques must incorporate the following properties [172]. Apart from confidentiality, data integrity, authentication, nonrepudiation (as discussed at the beginning of this chapter), the following properties must be incorporated.

1. Availability : Ensuring that the service offered by the whole WSN, by any part of it or by a single sensor node must be available whenever required.
2. Flexibility : Key establishment technique should be useful in multiple applications and allow for adding nodes at any time.
3. Survivability: Ability to provide service in case of power failure or attacks.
4. Adaptive security service: Ability to change security levels as resource availability changes.

Thus the main requirements of a sensor networks apart from those mentioned in Section 1.1.3 are

1. Key connectivity: Probability that two sensor nodes share some common key and thus communicate with each other must be high.
2. Resistance to node fabrication: The schemes must be able to resist node replication to guard against Sybil attacks [46].

3. Revocation: There must be some efficient way to revoke corrupted nodes.

4. Resiliency: Once nodes are captured or compromised, the rest of the network must be least affected.

We have already seen in Section 1.1.1 that there are broadly three ways of key establishment. One method to establish secret keys is by using public-key protocols. Though there has been instances of such schemes [51, 61, 65, 90, 100] using Elliptic curves and RSA, such protocols are quite expensive for sensor networks and are thus not used much in practice.

Another approach involves the Key Distribution Center(KDC) which is a resource rich center and acts as a trusted arbiter for key establishment. Examples of such scheme include TLS [43], SPINS [121] and Kerberos [152]. We will discuss SPINS later in Chapter 2. The Kerberos is an authentication protocol which is based on Needham and Schroeder's protocol [115]. In Kerberos, the truster server shares long lived keys which it shares with every node in the network and transmits session keys to sensor nodes on request. This method is extremely expensive for message reply so is not suitable for sensor networks.

The third method is to preload the keys in sensor nodes prior to deployment. This process is called Key predistribution.

There has been an extensive research on key predistribution schemes. A survey of key predistribution schemes is presented in Chapter 2.

1.2.1 Key predistribution in WSN

WSN which use symmetric key mechanism for key establishment, consists of the following three steps.

1. *Key predistribution*: Preloading keys in sensor nodes prior to deployment. The keys present in a sensor node constitute the key ring of the sensor.

2. *Shared-key discovery*: To find a common shared key between two communicating nodes.

3. *Path-key establishment*: If a common key does not exists, then a path has to be found between the communicating nodes. A path key is then established between the communicating nodes.

1.2.1.1 Basic terminology

1. Key pool: A set of keys from which subsets of keys are selected and placed in the sensor nodes.

2. Key ring (key chain): A group of keys that belong to a sensor node.

3. Node identifier: The unique index that is assigned to a node.

4. Key identifier: The unique index that is assigned to a key.

1.2.1.2 Key predistribution

Key predistribution in WSN can be done in any of the following three ways

1. Probabilistic: Key rings are randomly drawn from a key pool and placed in the sensor nodes. Two nodes communicate with each other with certain probability.

2. Deterministic: Key chains are place in sensor nodes following some definite pattern

3. Hybrid: Makes use of the above two approaches.

A naive approach to predistribute keys is to use a *single master key* in all the nodes. Thus each node can communicate with every other node in the network using this common key. This scheme is most efficient in terms of storage. However each node is a single point of compromise that brings the whole network down. On the other extreme consider a network containing N nodes, each node containing $N-1$ keys. Such nodes are said to share *pairwise keys*. This guarantees that any node is connected to all other nodes in the network. More importantly, the compromise of one or more nodes does not affect the connection between any other uncompromised nodes. However keeping in mind that sensors have limited storage, this choice is not feasible. Thus there is a trade off between the number of keys and the resiliency.

1.2.1.3 Shared-key discovery

There are a few methods of shared-key discovery. In some cases the key identifiers are broadcasted. A node wishing to communicate with another node checks the broadcasted identifiers and its own set of keys and finds out if there is a common key identifier. Then communication takes place using this common key. In another method a challenge response protocol is used. To find one or more common shared keys between two nodes, each node has to broadcast a list $\{\alpha, E_{k_i}(\alpha), i = 1, 2, \cdots, k\}$, where α is a challenge. The decryption of E_{k_i} with proper key by the other node would reveal the challenge α and establish a shared key with the broadcasting node. These approaches have been adopted in schemes like [28, 55]. Another way of finding shared keys was proposed by Pietro, Macini and Mei [126], in which pseudo-random key-index transformations are used to find the common keys. Deterministic methods broadcast only their node identifiers using which the share keys can be found. There is no need to exchange key identifiers. Later we will see throughout the thesis that all our key predistribution schemes using combinatorial structures make use of this method. We will also see that this method is better in many respects compared to the other three methods.

1.2.1.4 Path-key establishment

Where shared key exists between nodes, a secure channel is created and all communications between the nodes are performed using the common key. However there may exist situations where nodes may not share common keys (as in the scheme of [44] which uses *t-designs*) or when common shared keys are exposed because of node compromise. In such cases a path needs to be established between the nodes. Suppose nodes U_i and U_j having no common key need to communicate with each other. U_i establishes communication with some node U_1 through some common key which further establishes communication with U_2 and so on. Let $U_i, U_1, U_2, \cdots, U_l, U_j$ be the path between U_i and U_j. Let U_i share a common key k_1 with U_1. Similarly, let U_1 share a common key k_2 with U_2, and U_{l-1} share a common key k_l with U_l and U_l share a common key k_{l+1} with U_j. U_i generates a random key K, encrypts with k_1 and sends it to U_1. U_1 decrypts K using k_1 and encrypts it using k_2 and sends it to U_2 and the process continues. Ultimately K reaches U_j using k_{l+1}. So U_j can decrypt using k_{l+1} and obtain K. K is the path key and communication between U_i and U_j is done using K. This approach has been taken in [49]. The path is found in a breadth first manner. Sometimes a multipath key establishment scheme is used. In this technique [28] several paths are considered between the communicating nodes and a hash of all the keys is taken as a common key. This improves the resiliency of the network. For more details one may refer to [28].

1.2.2 Network models

There are several ways of classifying networks, based upon the type of operation and application [102, 103]. Broadly they can be classified as

1. **Distributed or Hierarchical**

   (a) *Distributed WSN (DWSN)* have no fixed infrastructure and the network topology is unknown prior to deployment. Sensor nodes are randomly scattered in the target region. Generally all nodes in a DWSN have equal capabilities and are thus homogeneous.

   (b) *Hierarchical WSN (HWSN)* contain different types of nodes based on their capabilities(heterogeneous). There are sensor nodes, cluster heads and base station with increasing order of battery power, memory and communication range. The sensor nodes are small and scattered in large numbers and collect information from the deployment region. This information is then sent to the respective cluster heads which collect information from a group of sensor nodes, process them and send the aggregate data to the base station. The cluster heads have more memory and battery power compared to the sensor nodes. The base station

is very powerful and may be a laptop or other PDA. It is generally secure and processes all the information and manages the overall network system. Data flow in the networks may be pairwise(unicast) among sensor nodes, group-wise(multicast) within cluster of nodes and network-wise(broadcast) from base station to the sensor nodes.

2. **Deployment location Control**

   (a) *Fixed, full control:* In which the precise deployment position is known prior to deployment. This may be applied in a factory warehouse or a plantation region.

   (b) *Fixed, partial control:* Partial information about the deployment of sensors is known. This class includes applications where sensor nodes are scattered in groups from airships.

   (c) *Fixed, no control:* The location of sensors cannot be predicted prior to deployment. This class includes applications where sensors are randomly deployed in the target region.

   (d) *Locally mobile*: In this sensors can move freely within a prescribed region only, but cannot move out of that region.

   (e) *Fully mobile*: The sensors are free to move anywhere within the network environment.

3. **Nature of ideal communication structure:** The ideal structure for communication that is desired for a sensor network [102, 103]. In case of a homogeneous network these are:

   (a) *t-complete*: All subsets of sensors are of size t. The most common example in this class are *pairwise complete*($t = 2$). This class includes networks in which there is no control over the deployment location.

   (b) *Locally t-complete*: All local subsets of size t, generally nodes which are neighbors of other nodes. The most common communication is referred to as *pairwise locally complete*.

   (c) *Regionally t-complete*: All subsets of sensors of size t within a specified region. This type of communication structure might be employed in case of a network with locally mobile sensors.

1.2.3 Attacks on key predistribution schemes

WSN vulnerability to numerous attacks may diminish their value and curtail their use [180]. The following assumptions are made about the capabilities of the attacker [70].

1. The attacker has unlimited energy and computing power.

2. The attacker has an access to all the information stored in a physically captured node.

3. The attacker is privy to all the traffic in the network and can record the same.

4. The attacker can introduce forged messages into the system.

5. The attacker has the ability to physically ascertain the location of a given sensor by listening to the traffic.

6. The attacker has the ability to fabricate similar nodes and deploy them.

One option is to make the senor nodes tamperproof [125]. However owing to the expensive costs, this cannot be afforded for the large number of low powered sensor nodes. There are different types of models for node capture [70].

1. **Random node capture attack**: The nodes are captured randomly from the deployment region.

2. **Selective node capture attack**: The random node capture is very weak. Sometimes an adversary may capture nodes selectively so that she can purposely attack certain areas and groups of sensors which are possibly located close to each other. Let s is the number of compromised sensors, J_s is the cardinality of the set of compromised keys when s nodes are compromised, v is the size of the key pool, k is the number of keys preinstalled in each sensor, N is the total number of sensors deployed in the network, and a is a variable. B is used to represent the threshold that an attacker is to inspect and then to decide which sensors to capture next. The mathematical model of selective attack is presented as follows:

$$B = \frac{\binom{v-J_s}{k-a}\binom{J_s}{a}}{\binom{v}{k}}(N-s),$$

where $a = 0, 1, 2, \ldots, k$ and $\frac{\binom{v-J_s}{k-a}\binom{J_s}{a}}{\binom{v}{k}}$ is the probability that there exists uncompromised nodes and each of them has $k-a$ keys not already compromised, $N-s$ is the total number of uncompromised nodes in the system. The heuristic method is described as follows. Initially, when $a = 0$, an attacker can arbitrarily capture a sensor and derive k keys preinstalled in the captured sensor and $J_s = k$. Then, she inspects B: if $B \geq 1$, she continuously captures the nodes with $k-a$ keys that are not already compromised, and for each capture, J_s is increased by $k-a$; if $B < 1$, she increases a by 1 until $B \geq 1$. She then captures the sensors with $k-a$ keys that are not already compromised. The attacker continues this process until the condition $k = a$ is fulfilled or the entire key pool is compromised. The condition $B > 1$ means

there exists uncompromised sensor that has $k - a$ keys that are not already compromised. It has been shown in [70] that that the selective node capture attack can gain more information than random node capture attack with the same number of captured sensors.

Another selective node capture attack is given in [126]. In order to disconnect links, the attacker compromises and collects a subset T of the keys in the pool. The attacker has already compromised a number of sensors, and has collected all their keys in a set J . For every sensor s in the WSN, the *key information gain* $G(s)$ is a random variable equal to the number of keys in the key ring of s which are in T and are not in J . For example, if the attacker's goal is to compromise the channel between sensors U_i and U_j, subset T in the above definition is equal to $P(U_i) \bigcap P(U_j)$, that is it contains all the keys which are in the key ring of both U_i and U_j . Assuming that the attacker has collected a set J of keys, random variable $G(U_a)$ is equal to $|(P(U_a) \bigcap P(U_i) \bigcap P(U_j)) \backslash J|$. At each step of the attack sequence, the next sensor to be tampered with is sensor U_a, where U_a maximizes $E[G(U_a)|I(U_a)]$, the expectation of the key information gain $G(U_a)$ given the information $I(U_a)$ that the attacker knows on sensor U_a key ring. Random attack does not take advantage of any information leaked by compromised sensors. Conversely, the smart attacker greedily uses this information to choose which sensor to corrupt in order to maximize the number of useful keys it is expected to collect. Note that information $I(U_a)$ may also depend on the protocol used by sensor U_a and, in particular, on the shared-key discovery phase.

3. **Node Fabrication Attack:** In the node fabrication attack, the attacker is successful in capturing a small number of nodes and extracting the keys stored therein. The attacker next uses this data, to fabricate nodes that would eventually masquerade as existing uncompromised node and would operate in zones of network where the original node is absent. The uncompromised sensors in the network cannot detect the fabricated nodes as anomalous nodes as long as they can have standard communication with them. This attack is severer as compared to passive listening attacks as the attacker may have enough information to fabricate many sensors with many different identities and possibly outnumber the original set of sensors. These fabricated nodes are apparently good nodes, since they all have valid keys. Thus, the fabricated nodes can quickly outnumber the uncompromised nodes. However, there are a few restrictions for the attacker. A wrong identifier will not guarantee that a fabricated sensor can set up a pairwise key with uncompromised sensors. Thus properly assigning identifiers to nodes can prevent this attack.

We will see later in this thesis that our schemes are secure against selective node capture attack and node fabrication attack.

1.2.4 Definitions

Nodes can communicate with each other provided they are within communication range. We define Radio Frequency (RF) region to be the region around a node where the node can physically communicate with other nodes provided they share a common key.

Definition 1.2.1. ***RF region*** *is the circular region around a node within which all the nodes can communicate with each other, provided they share a common key. The radius of the circle is called the RF radius.*

Throughout this thesis we stress on the *connectivity ratio* or simply called *connectivity.*

Definition 1.2.2. *The* ***connectivity ratio*** *or* ***connectivity****, denoted by p_c of a network is the probability that a link exists between two nodes. This is defined as*

$$p_c = \frac{\textit{Number of links present between nodes}}{N(N-1)/2},$$

where, N is the number of nodes in the network.

We use two measures of resiliency throughout the thesis. We can think of the sensor network as a a graph with nodes as the vertices and edges existing between two nodes, if the nodes share common keys. We define two measures of resiliency, one related to vertex connectivity, denoted by $V(s)$ and other related to edge connectivity, denoted by $E(s)$, where s is the number of nodes compromised.

Definition 1.2.3. *$V(s)$ is defined as the fraction of nodes disconnected when s nodes are compromised. This is given by,*

$$V(s) = \frac{\textit{Number of nodes disconnected when } s \textit{ nodes are compromised}}{N}$$

where, N is the number of nodes in the network.

We say that a node is disconnected when all the keys in the nodes are present in the compromised nodes.

Definition 1.2.4. *$E(s)$ is defined as the fraction of edges disconnected when s nodes are compromised. This is given by,*

$$E(s) = \frac{\textit{Number of links (edges) present after } s \textit{ nodes are compromised}}{\textit{Number of links (edges) present before } s \textit{ nodes are compromised}}$$

1.3 Key distribution for Traitor Tracing

For multicast security where information needs to be sent to many users securely, the data is so encrypted that only authorized users can access it, while unauthorized users are unable to access it. One common example is the pay-TV broadcast. Here the users are given decryption keys and the content is encrypted and broadcasted so that only authorized users with valid keys can decrypt the content and view it. The group of users may be long lived but is dynamic, that is, any user can join or leave at any point of time. However sometimes dishonest users retransmit their contents once they decrypt it or disclose their personal keys to unauthorized users. Such dishonest users are called *traitors*. Traitor tracing essentially aims at tracing the source of piracy once it has taken place.

A problem related to traitor tracing is to device a good Broadcast Encryption Scheme (BES) [57]. In a broadcast encryption scheme, a trusted authority wants to broadcast a message to a group of users. The message may be encrypted suitably so that only the intended set of users can obtain it. The main concerns of a good BES is that the communication overhead and the space required for key storage should be minimized.

Let us consider an example in which message $\mathcal{M}$ is to be broadcasted. Suppose the message is encrypted with a key K and the encrypted message $y = e_K(\mathcal{M})$ is broadcasted. Let $Z \subseteq \mathcal{U}$ be a set of users who are authorized to receive the contents. The users are given decryption keys to decrypt the contents. Z is called the privileged set of users or authorized users. Any set $R \nsubseteq Z$ cannot decrypt the contents. R is called the unauthorized set. One option is to assign a distinct key to each user and encrypt the message $\mathcal{M}$ with each of these keys. This is infeasible as it incurs a huge communication overhead. Another simple solution is to provide every possible subset of users with key, i.e., give every user the keys corresponding to the subsets it belongs to. This requires every user to store a huge number of keys. So keys must be intelligently distributed in such a way that the communication overheads and the storage requirement for secret information is optimum.

For broadcasting secret information, every valid user U_i is given a set of keys which is called its *personal set* and is denoted by $P(U_i)$. Two blocks of data is then broadcasted by the Trusted Authority (TA). One block is called the *cipher block*. It contains the message encrypted by a session key k_s. The other broadcasted block is called the *enabling block* from which the users can generate their session keys using the personal set of keys. The users can then decrypt the contents of the cipher block using this session key. The operation of a decoding box is shown in Figure 1.1.

Sometimes users come together and contribute keys to form a pirate decoder. The pirate decoder is then given to unauthorized users who can then generate session keys and decrypt the contents of the cipher block. The problem is to trace at least one traitor once the pirate decoder is confiscated. There are several

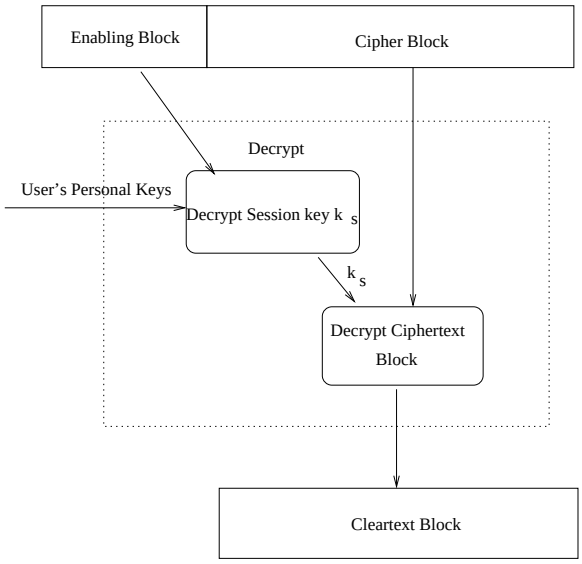


Figure 1.1: Operation of a decoding box

techniques for key distribution and suitable tracing methods. In the next chapter we describe the key distribution techniques and the tracing schemes used.

1.3.1 Properties and contents

A traitor tracing scheme consists of the following elements.

1. A **user initialization scheme** used by the data supplier to add new users and distribute the set of personal keys to them.

2. An **encryption/decryption scheme**: Encryption scheme E_α is used by the data supplier to encrypt the session key before broadcasting, and a decryption scheme D_β is used by each authorized user (i.e. its decoder) to decrypt the session key, where $\beta = P(U_i)$, where U_i is the i-th user. Obviously, for any session key k_s, $k_s = D_\beta(E_\alpha(k_s))$. The session key is used to encrypt data content using a off-the-shelf symmetric encryption scheme such as DES.

3. A **traitor tracing algorithm** which upon confiscation or probing determines the identity of at least one traitor.

Traitor tracing systems can have a number of properties informally defined below [13]:

1. **Open vs. secret schemes**: The decryption scheme is known to the public and only the keys are secret in a open scheme. However both the decryption scheme as well as the keys are secret in the secret schemes.

2. **Public vs. secret broadcast key**: Some traitor tracing systems require that the broadcasters key BK remain secret; Other systems [11, 83, 114, 146, 165] support a public BK so that anyone can create broadcast data. Combinatorial systems such as [5, 6, 33, 34, 58, 60, 111, 141, 148, 151, 158, 159] are typically designed for the secret BK settings, but can be made public-key by replacing the underlying ciphers by public key systems.

3. **Public vs. secret tracing key**: Many traitor tracing systems assume that the tracer is a trusted party and require that the tracers key TK be kept secret. Some exceptions are [81, 122, 124, 170]. In these settings TK is part of the public key.

4. **Collusion resistance**: A traitor tracing system is said to be c-collusion resistant if tracing will work as long as the pirate has fewer than c user keys at her disposal. If $c = N$ the system is said to be fully collusion resistant.

5. **Black box tracing**: The tracing algorithm must do its job by treating the pirate decoder as a black-box oracle. Clearly, the pirate is free to build the decoder however she wants. The pirate could obfuscate the code, use tamper resistant hardware, and try to randomize the secret keys at her disposal. Hence, it is safest to treat the pirate decoder as a black box so that the tracing algorithm cannot look into the inner-workings of the decoder. More precisely, the tracer gives the decoder encrypted content and the decoder responds with valid or invalid. In real world settings, valid means the decoder plays the given encrypted content and invalid means it does not. The tracer learns nothing else.

6. **Stateful vs. stateless decoders**: A stateless decoder is one that does not keep state between decryptions. For instance, software decoders, such as DeCSS, cannot keep any state. Pirate decoders embedded in tamper resistant hardware, such as a pirate cable box, can keep state between successive decryptions. When the decoder detects that it is being traced it could shutdown and refuse to decrypt further inputs. Kiayias and Yung [80] show how to convert any tracing system for stateless decoders into a tracing system for stateful decoders by embedding robust watermarks in the content.

In this thesis we discuss several predistribution schemes for sensor networks and traitor tracing. We discuss the advantages and disadvantages of each scheme and compare them with other related works in these areas. Though the two problems of sensor networks and traitor tracing have different applications, the key distribution problems lie at the core of these two areas. We will actually see in this thesis, that several key predistribution techniques that work for sensor networks also work for traitor tracing.

1.4 Combinatorial Designs

The main emphasis of this thesis is to demonstrate the efficient usage of combinatorial designs and codes for key predistribution. The bulk of this thesis has been dedicated to the study of different combinatorial designs and their correspondence to the two areas of application - sensor networks and traitor tracing. In this thesis we study several designs like Balanced Incomplete Block Designs (BIBD), Partially Balanced Incomplete Block Designs (PBIBD), Transversal designs (TD), Group-divisible Designs, Orthogonal Arrays (OA). We also propose an innovative methods to use codes for key predistribution in sensor networks. In this regard we study and apply the Reed-Solomon codes to key predistribution in sensor networks. The correspondence of codes and designs has been know since a long time. This motivated us to use codes in key predistribution.

Combinatorial designs and codes have very interesting patterns. By studying these patterns we can device efficient key establishment schemes which were not possible in many randomized key predistribution schemes. We also come up with a new type of designs construction called *expanded design* which we study in Chapter 8. On one hand we are able to design better key predistribution schemes, on the other the designs we use, have their own mathematical interest.

1.5 Definitions and notations

Definition 1.5.1. *A **set system** or **design** [86] is a pair $(X, \mathcal{A})$, where $\mathcal{A}$ is a set of subsets of X, called **blocks**. The elements of X are called **varieties** or **elements**. A Balanced Incomplete Block Design $BIBD(v, b, r, k, \lambda)$, is a **design** which satisfy the following conditions:*

1. *$|X| = v$, $|\mathcal{A}| = b$,*
2. *Each subset in $\mathcal{A}$ contains exactly k elements,*
3. *Each variety in X occurs in r blocks,*
4. *Each pair of varieties in X is contained in exactly λ blocks in $\mathcal{A}$.*

A $BIBD(v, b, r, k, \lambda)$ design can be represented by a *incidence matrix* $M = [m_{ij}]$ of dimension $v \times b$ with entries 0 and 1. $m_{ij} = 1$, if the ith variety is present in the jth block and 0 otherwise.

Definition 1.5.2. *Suppose that $(X, \mathcal{A})$ is a set system, where*

$$X = \{x_i : 1 \leq i \leq v\}$$

and

$$\mathcal{A} = \{A_j : 1 \leq j \leq b\}.$$

*The **dual set system** of (X, A) is any set isomorphic to the set system (X', A') where*

$$X' = \{x'_j : 1 \leq j \leq b\},$$

$$\mathcal{A}' = \{A'_i : 1 \leq i \leq v\},$$

and where

$$x'_j \in A'_i \iff x_i \in A_j.$$

It follows that if we take the dual of a $BIBD(v, b, r, k, \lambda)$, we arrive at a design containing b varieties, v blocks each block containing exactly r varieties and each variety occurring in exactly k blocks. We also note that any two blocks contain λ elements in common.

Definition 1.5.3. *When $v = b$, the $BIBD$ is called a **symmetric BIBD** (SBIBD) and denoted by $SB[v, k; \lambda]$.*

Definition 1.5.4. *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 1.5.5. *The **intersection number** between any two blocks is the number of elements common to the blocks.*

Definition 1.5.6. *Let the intersection numbers between any the blocks in a $BIBD$ be μ_1, μ_2, $\cdots$, μ_x. Let $M = \{\mu_i : i = 1, 2, \cdots, x\}$. Let $\mu = max\{\mu_1, \mu_2, \cdots, \mu_x\}$. μ is called the **linkage** of the design.*

We note that for a $SBIBD$, $|M| = 1$ and $\mu = \lambda$.

Definition 1.5.7. *A balanced incomplete block design is said to be **resolvable** if the set of b blocks can be partitioned into t classes such that each variety appears in exactly one class. The classes are called **resolution classes** of the design.*

A detailed discussion of resolvable designs is presented in [162, Chapter 8]

Definition 1.5.8. *An **association scheme with m associate classes** [162, Section 11] on the set X is a family of m 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, $1 \leq i \leq m$,*

3. *for each i, $1 \leq i \leq m$, 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, $n_i (1 \leq i \leq m)$ and $p^i_{jl} (1 \leq i, j, l \leq m)$ are called the* parameters of the association scheme.

p^i_{jl} are represented in the form of a matrix with $P_i = [p^{(i)}_{jl}]$, for $1 \leq i, j, l \leq m$.

Definition 1.5.9. *A **partially balanced incomplete block design with m associate classes**, denoted by $PBIBD(m)$ [162] is a design on a v-set X, with b blocks each of size k and with each element of X being repeated r times, such that if there is an association scheme with m classes defined on X where, two elements x and y are i-th $(1 \leq i \leq m)$ associates, then they occur together in λ_i blocks. We denote such a design by $PB[k, \lambda_1, \lambda_2, \cdots, \lambda_m; v]$.*

The association scheme for a two-associate-class PBIBD [162, Section 11.3] are most interesting.

Definition 1.5.10. *Let X be a set of varieties such that*

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

*The G_is are called **groups** 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.*

Definition 1.5.11. *A **transversal design** [162, Section 6.3] $TD(k, \lambda; r)$, with k groups of size r and index λ, is a triple $(X, G, \mathcal{A})$ where*

1. *X is a set of kr elements (varieties),*
2. *$G = \{G_1, G_2, \cdots, G_k\}$ is a family of k sets (each of size r) which form a partition of X,*
3. *A is a family of k-sets (or blocks) of varieties such that each k-set in $\mathcal{A}$ intersects each group G_i in precisely one variety, and any pair of varieties which belong to different groups occur together in precisely λ blocks in $\mathcal{A}$.*

We denote a transversal design with $\lambda = 1$ as $TD(k, r)$. It can be shown that if there exists a $TD(k, r)$, then there exists a (v, b, r, k) design with $v = kr$, $b = r^2$.

Let us now explain $(X, \mathcal{A})$ in a transversal design $TD(k, r)$.

1. $X = \{(x, y) : 0 \leq x < k, 0 \leq y < r\}$,
2. For all i, $G_i = \{(i, y) : 0 \leq y < r\}$,
3. $\mathcal{A} = \{A_{i,j} : 0 \leq i < r\ \&\ 0 \leq j < r\}$.

We consider the $TD(k, r)$ in which we define a block $A_{i,j}$ by

$$A_{i,j} = \{(x, xi + j \bmod r) : 0 \leq x < k\} \tag{1.5.1}$$

This construction of TD has been used in many schemes.

Let v, k, λ and t be positive integers such that $v > k \geq t$.

Definition 1.5.12. *A t-(v, k, λ) is a design $(X, \mathcal{A})$ such that the following properties are satisfied:*

1. *$|X| = v$,*
2. *Each block in $\mathcal{A}$ contains exactly k elements and*
3. *Each set of t distinct points is contained in exactly λ blocks.*

*Such a design is called a t-**design**.*

For a $BIBD(v, b, v, k, \lambda)$, $t = 2$. A detailed study of *t-design* appears in [154, Chapter 9]. According to the construction given in [154, Chapter 9] the following result can be stated.

Theorem 1.5.13. *For all prime powers q, there exists a $3-(q^2+1, q+1, 1)$ design.*

Let t, v, k, and λ be positive integers such that $k \geq t \geq 2$.

Definition 1.5.14. *A t- (v, k, λ) **orthogonal array** (denoted by t-$(v, k, \lambda)-OA$) is a pair (X, D) such that the following properties are satisfied.*

1. *X is a set of v elements called points.*
2. *D is a λv^t by k array whose entries are chosen from the set X.*
3. *Within any t columns of D, every t-tuple of points is contained in exactly λ rows.*

Let $V_{n+1}(q)$ denote the $(n + 1)$-dimensional vector space over $GF[q]$ and let $PG(n, q)$ be the set of all subspaces of $V_{n+1}(q)$. Then $PG(n, q)$ is called the *projective geometry of dimension n* over $GF[q]$ [162]. The points of $PG(n, q)$ are the subspaces of pdim 0, the lines are the subspaces of pdim 1, the planes are subspaces of pdim 2 and the hyper planes are subspaces of pdim $(n - 1)$. Two spaces of $PG(n, q)$ are said to be incident if they have some common points.

Definition 1.5.15. *A **Finite Projective Plane** consists of a finite set of points P and a set of subset of P called* lines. *For an prime power q, a* Finite Projective Plane *consists of $q^2 + q + 1$ points, $q^2 + q + 1$ lines where each line contains $q + 1$ points and each point occurs in $q + 1$ lines.*

Definition 1.5.16. *A **Finite Generalized Quadrangle GQ(s,t)** is an incidence structure $S = (P, B, I)$ where P and B are disjoint and nonempty sets of points and lines respectively, and I is a symmetric point-line incidence relation satisfying the following conditions:*

1. *Each point is incident with $t + 1$ lines ($t \geq 1$) and two distinct points are incident with at most one line,*
2. *Each line is incident with $s + 1$ points ($s \geq 1$) and two distinct lines are incident with at most one point,*
3. *If x is a point and L is a line not incident with x, then there is a unique pair $(y, M) \in P \times B$ for which $xIMIyIL$.*

In a $GQ(s, t)$, there are $v = (s + 1)(st + 1)$ points and $b = (t + 1)(st + 1)$ lines where each line contains $s + 1$ points and each point appears on $t + 1$ lines. A detailed discussion on generalized quadrangles can be found in [119].

Definition 1.5.17. *A* code *[154, Chapter 10] is a pair $(Q, \mathcal{C})$ such that the following properties are satisfied.*

1. *Q is a set of elements called* symbols.
2. *$\mathcal{C}$ is a set of n-tuples of symbols called* codewords *(i.e. $\mathcal{C} \subseteq Q^n$), where $n \geq 1$ is an integer.*

Let $(Q, \mathcal{C})$ be a code, where $\mathcal{C} \subseteq Q^n$. Let $x, y \in Q^n$, we define the *Hamming distance* between x and y to be

$$d(x, y) = |\{i : x_i \neq y_i\}|,$$

where $x = (x_1, x_2, \cdots, x_n)$ and $y = (y_1, y_2, \cdots, y_n)$. The distance of the code $(Q, \mathcal{C})$, denoted by $d(\mathcal{C})$, is the smallest positive integer d such that $d(x, y) \geq d$ for all $x, y \in \mathcal{C}$, $x \neq y$.

$(Q, \mathcal{C})$ is an (n, M, d, q)-code if the following properties are satisfied:

1. $|Q| = q$,
2. $\mathcal{C} \subseteq Q^n$,
3. $|\mathcal{C}| = M$,
4. $d(\mathcal{C}) \geq d$.

A code $(Q, \mathcal{C})$ is a *linear code* of dimension k, if $Q = F_q$ for some prime power q and $\mathcal{C}$ is a k - dimensional subspace of the vector space $(F_q)^n$.

We deviate from the usual symbols used to represent codes to maintain uniformity of our definition. We represent the length of the code by k, the number of codewords by N, the dimension by dim and keep the distance as d. Thus in our notation, a (k, N, d, q) defines a code.

Symbol	Meaning
N	Number of nodes or users in the network
X	Key pool
$v = \|X\|$	Size of the key pool
U_i	Sensor node i or user i
$P(U_i)$	Keys possessed by the user U_i
$k = \|P(U_i)\|$	Number of keys possessed by a sensor node or a user
p_c	Connectivity ratio of a sensor network
ρ	RF radius
ρ_l	Lee distance
s	Number of compromised sensor nodes
J_s	Number of keys compromised when s node are compromised
B_s	Number of links broken when s node are compromised
$V(s)$	Fraction of nodes disconnected when s nodes are compromised
$E(s)$	Fraction of links broken when s nodes are compromised
c	Number of colluding users, nodes
$\mathcal{B}$	Communication structure
Z	Permissible set of users
R	Forbidden set of users
$\mathcal{M}$	Message
k_s	Session key
F_q	Field containing q elements

Table 1.1: Notations

1.6 Notations

The Table 1.1 gives a list of notations used throughout this thesis.

1.7 Thesis plan

This thesis is broadly based on the papers [131–138]. A large portion of this thesis is dedicated to Sensor Networks. In Chapter 8, we discuss key distribution techniques for traitor tracing. In Chapter 2, we provide the basic background for sensor network and traitor tracing. We discuss several predistribution schemes in sensor networks and point out their merits and demerits. We also discuss several traitor tracing schemes, involving both public key as well as combinatorial techniques.

In Chapter 3, we discuss a key predistribution scheme using Partially Balanced Incomplete Block Design (PBIBD). It is based on the three papers [132, 134, 136]. We analyze the resiliency of the network and give upper bounds for $E(s)$. The

scheme has $O(\sqrt{N})$ keys per sensor, where N is the size of the network. The key establishment algorithms are simple and run in $O(1)$ time. The communication overhead is $O(\log N)$. By carefully manipulating the design we can scale the network to twice its original size.

In Chapter 4, we present a key predistribution scheme using codes. In general, the predistribution scheme can make use of any codes. In particular we use Reed-Solomon codes for predistribution. Our scheme however has the greatest advantage that the system is scalable. The scheme requires $O(\sqrt{N})$ keys. Key establishment can be done in constant time and communication overhead is $O(\log N)$. This chapter is based on the paper [137].

In Chapter 5, we present a grid based deployment scheme. According to this scheme, the sensors are placed at the grid intersection point. We study the resiliency and connectivity of the network with changing RF radius. We consider the concept of Lee distance while estimating the connectivity and resiliency of the network. This chapter is based on the papers [131, 133].

Chapter 6 is based on the paper [138]. This is also a deployment knowledge based scheme, in which the entire deployment region is broken down into smaller regions. The nodes within each region can communicate with each other. There are specialized nodes called agents, which have higher battery power and storage. Nodes across regions can communicate via agents. The keys in the small sensor nodes are predistributed using projective planes, whereas the keys in the agents are predistributed using a modified form of transversal design. The small sensor nodes have $O(\sqrt{N})$ keys, where N is the number of sensors in each region. The communication overhead is $O(\log N)$ for the small sensor nodes and key establishment algorithm takes constant time to execute. The most important feature of this scheme is that there are only three agents are enough to ensure that there are direct link between regions. We study the inter region and intra regions resiliency. The connectivity within each region and across regions is always one. We compare the scheme with existing schemes.

Chapter 7 discusses key establishment algorithms for existing schemes like Çamtepe and Yener [18, 20] and Dong, Pei and Wang [44]. The shared key discovery algorithms were not present in the papers. When devising key establishment algorithms we also modified the Çamtepe and Yener scheme, which makes key predistribution efficient. We present the correctness of each of these algorithms and show that the algorithms run in $O(1)$ time. Chapter 7 is the outcome of the paper [136].

In Chapter 8 we present several combinatorial key distribution schemes for traitor tracing. Our schemes can trace all traitors who collude. We also device a new kind of design which is very similar to the Kronecker product. This results in a special type of design which we call *Expanded design*. We study the properties of this design and present constructions of PBIBD. We use BIBDs, PBIBDs, Transversal designs, generalized quadrangles, t-designs for efficient key distributions schemes. This Chapter is based on [135].

We conclude in Chapter 9, with some open problems and mention the scope of future work.

Chapter 2

Background

This chapter can be broadly divided into two parts. In the first part we present a literature survey of key predistribution schemes in context of Sensor Networks. We mention several schemes and point out their advantages and disadvantages.

In the second part of the chapter we discuss several traitor tracing schemes. These schemes have again been divided into public key schemes and combinatorial schemes. We discuss several types of traitor tracing strategy and their applicability to various scenarios.

2.1 Key predistribution in WSN

key predistribution in WSN can be done using any of the following three techniques.

1. Probabilistic: Key chains are randomly drawn from a key pool and placed in the sensor nodes. Two nodes communicate with each other with certain probability.

2. Deterministic: Key chains are placed in sensor nodes following some definite pattern

3. Hybrid: Makes use of the above two approaches.

We have already seen in 1.2.1.2 that one naive approach to predistribution is to use a *single master key* across all the nodes. So each node can communicate with every other node in the network using this common key. This scheme is most efficient in terms of storage. However if one node is compromised, the whole network crumbles down. On the other extreme for a network containing N nodes, each node is assigned $N-1$ keys, nodes are said to share *pairwise keys*. This guarantees that any node is connected to all other nodes in the network. More importantly, the compromise of one or more nodes does not affect the connection between any other uncompromised nodes. However keeping in mind that sensors

have limited storage, this choice is not feasible. Thus there is a trade off between the number of keys and the resiliency.

We now discuss two approaches which form the basis of several key predistribution schemes. These schemes though not introduced for WSN, has been modified and suitably applied by several researchers for key predistribution in sensor networks.

2.1.1 Blom's Scheme

Blom [9] proposed a key predistribution scheme that allows any two nodes of a group to find a pairwise key. The security parameter of the scheme is c. As long as no more than c nodes are compromised, the network is perfectly secure (this is called the c-secure property). During the pre-deployment phase, the the distribution server first constructs a $(c+1) \times N$ matrix G over a finite field $GF(q)$, where N is the size of the network. G is considered as public information; any node can know the contents of G, and even adversaries are allowed to know G. Then the base station creates a random $(c+1) \times (c+1)$ symmetric matrix D over $GF(q)$, and computes an $N \times (c+1)$ matrix $A = (D.G)^T$, where $(D.G)^T$ is the transpose of $D.G$. Matrix D needs to be kept secret, and should not be disclosed to adversaries or any node (although, as will be discussed later, one row of $(D.G)^T$ will be disclosed to each node). Since D is symmetric, it is easy to see that: $A.G = (D.G)^T.G = G^T.D^T.G = G^T.D.G = (A.G)^T$.

Thus $A.G$ is a symmetric matrix. Let $K = A.G$, we know that $K_{ij} = K_{ji}$, where K_{ij} is the element in K located in the ith row and jth column. K_{ij} (or K_{ji}) is the pairwise key between node U_i and node U_j. To carry out the above computation, nodes U_i and U_j should be able to compute K_{ij} and K_{ji}, respectively. This can be easily achieved using the following key pre-distribution scheme, for $w = 1, \cdots, N$:

1. store the wth row of matrix A at node U_w, and
2. store the wth column of matrix G at node U_w.

Therefore, when nodes U_i and U_j need to find the pairwise key between them, they first exchange their columns of G, and then they can compute K_{ij} and K_{ji}, respectively, using their private rows of A. Because G is public information, its columns can be transmitted in plaintext. It has been proved that the above scheme is c-secure if any $c+1$ columns of G are linearly independent. This property guarantees that no member other than U_i and U_j can compute K_{ij} or K_{ji} if no more than c members are compromised.

A construction for matrix G [48]: We note that any $c+1$ columns of G must be linearly independent in order to achieve the c-secure property. Since each pairwise key is represented by an element in the finite field $GF(q)$, if the length of pairwise keys is 64 bits, then q can be chosen as the smallest prime number that is

larger than 2^{64}. Let a be a primitive element of $GF(q)$ and $N < q$. That is, each nonzero element in $GF(q)$ can be represented by some power of a, namely a^i for some $0 < i \leq q-1$. A feasible G can be designed as follows [98]:

$$G = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ a & a^2 & a^3 & \cdots & a^N \\ a^2 & (a^2)^2 & (a^3)^2 & \cdots & (a^N)^2 \\ a^3 & (a^2)^3 & (a^3)^3 & \cdots & (a^N)^3 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a^c & (a^2)^c & (a^3)^c & \cdots & (a^N)^c \end{bmatrix}$$

It is well-known that $a^i \neq a^j$ if $i \neq j$ (this is a property of primitive elements). Since G is a Vandermonde matrix, it can be shown that any $c+1$ columns of G are linearly independent when $a, a^2, a^3, \ldots, a^N$ are all distinct. In practice, G can be generated by the primitive element a of $GF(q)$. Therefore, the w-th column of G is stored at node U_w, it is only required to store the seed a^w, and any node can regenerate the column given the seed.

2.1.2 Blundo et al's Scheme

This scheme was proposed by Blundo, Santis, Herzberg, Kutten, Vaccaro, Yung [10] and was not originally used for sensor networks. It uses a symmetric bivariate polynomial over some finite field $GF(q)$, i.e. a polynomial $P(x,y) \in GF(q)[x,y]$ with the property that $P(i,j) = P(j,i)$ for all $i, j \in GF(q)$. A node with ID i stores a share in P , consisting of the univariate polynomial $f_i(y) = P(i,y)$. In order to communicate with node U_j, it computes the common key $K_{ij} = f_i(j) = f_j(i)$; this process enables any two nodes to share a common key. If P has degree t, then each share consists of a degree t univariate polynomial; each node must then store the $t+1$ coefficients of this polynomial. These are elements of $GF(q)$, as are the pairwise keys that are established; thus, storing a degree t share requires as much space as storing $t+1$ keys. If an adversary captures s nodes, where $s \leq t$, then it does not learn any information about keys established between uncompromised nodes; however, if it captures $t+1$ or more nodes then it can interpolate to compute the polynomial P and hence learn all the keys.

In the next few sections we discuss several key predistribution schemes for various kinds of networks.

2.2 The Basic Scheme of Eschenauer and Gligor

The first random key predistribution scheme for WSN was proposed by Eschenauer and Gligor in ACMCCS'02 [55]. This scheme is known as the *Basic Scheme*. Many predistribution techniques use this as the underlying scheme. The basic scheme consists of three steps: key predistribution, shared-key discovery and path-key

establishment. We have discussed them in Section 1.2.1.2. When sensor nodes are compromised, the compromised keys must be revoked. For this a controller node broadcasts a revocation message containing the list of identifiers of keys which have been compromised. To sign the list of key identifiers, the controller generates a signature key K_e and unicasts it to each node by encrypting it with a key K^{ci}. $K^{ci} = E_{K_x}(ci)$, where $K_x = K_1 \oplus K_2 \cdots, \oplus K_k$, where K_i are the keys in the node's key ring, ci denotes the identity of the controller node and E_{K_x} denotes the encryption with key K_x. After obtaining the signature key, each node verifies the signature of the signed list of key identifiers, locates those identifiers and removes them from the key ring. After the keys are removed, the shared key discovery and path key establishment steps are carried out again. A similar process takes place during re-keying. Re-keying takes place when the lifetime of the keys expire.

Suppose the probability of the common key existing between two node is p. The degree of a node is simply the average number of edges connecting that node with other nodes in neighborhood and is given by $d = p * (N - 1)$, where N is the number of nodes in the network.

A random graph $G(N, p)$ is a graph consisting of N nodes and p representing the probability of establishing a link between two nodes. Erdös and Rényi [54] showed that there exists a probability state p, which moves from state zero to state one for large random graphs. The function that defines p is called the threshold function of a property. If we are given a desired probability (P_c) for graph connectivity, then P_c is given as $P_c = lim_{N\to\infty} Pr[G(N, p)\text{is connected}] = e^{e^{-c}}$, $p = \frac{ln(N)}{N} + \frac{c}{N}$, where c is a real constant. Probability that two nodes share a common key is given by $p_c = 1 - Pr[\text{two nodes do not share any key}] = 1 - \frac{((|X|-k)!)^2}{(|X|-2k)!|X|!}$. For example if $|X| = 10,000$ then $k = 75$ when $p = 0.5$.

The main advantages of this scheme are that the scheme is flexible, efficient and easy to implement. However, the main disadvantages are that it cannot be used in regions which require high security and node to node authentication.

2.3 Q-composite Scheme

A variation of the basic scheme was introduced by Chan, Perrig and Song [28]. This scheme is called the Q-composite scheme. Suppose X is the key pool and the number of keys in each sensor is k. According to this scheme two nodes can communicate with each other provided they share $q' \geq q$ keys between them. The key predistribution step is the same as the basic scheme, in which k keys are selected from a key pool P and placed in sensor nodes. Two nodes wishing to communicate, broadcast either their key identifiers or use a challenge-response protocol to find out the common shared keys. If $q' \geq q$ keys are shared between two nodes, then a link key K is generated as the hash of all shared keys as $K = hash(k_1||k_2||\cdots||k_{q'})$. The keys are hashed in some canonical order, for example based on the order in which they occur in the original key pool X. No key establishment occurs between

nodes which share less than q keys. The probability of establishing a link between two keys is given by

$$p_c = 1 - \Big(p(0) + p(1) + \cdots + p(q-1)\Big), \text{ where}$$
$$p(i) = \frac{\binom{|X|}{i}\binom{|X|-i}{2(k-i)}\binom{2(k-i)}{k-i}}{\binom{|X|}{k}^2}$$

When establishing the key pool the largest $|X|$ is chosen, such that $p_c \geq p$.

The Q-composite scheme has very high resiliency when the number of nodes compromised is small. However as the number of nodes compromised increases, more and more keys are compromised and nodes share fewer than q keys and hence no link key exists. Thus resiliency decreases drastically as the number of compromised node increases. Probability that a secure link is broken when s nodes are compromised is given by

$$\sum_{i=q}^{k} \left(1 - (1 - \frac{k}{|X|})^s\right)^i \frac{p(i)}{p_c}$$

Hwang and Kim [72] revisit the random graph theory and use giant component theory by Erdös and Rényi to show that even if the node degree is small, most of the nodes in the network can be connected. Further, they use this fact to analyze the basic scheme [55], Du et al [48], and Chan et als [28] key pre-distribution schemes and evaluate the relation between connectivity, memory size, and security.

2.4 Random pairwise schemes

In a pairwise scheme each pair of nodes share a secret key with each other. For a network consisting of N nodes, each node has $N-1$ keys which it shares with the $N-1$ nodes. As already mentioned in Section 2.1, this is a huge burden on the sensor, since sensors have very limited memory. We now discuss some pairwise schemes.

2.4.1 Chan-Perrig-Song scheme

The *Random pairwise keys scheme*, proposed by Chan, Perrig and Song [28] is a modification to the above scheme, in which not all the $N-1$ keys are to be stored in the nodes. Let p_c be the smallest probability that two nodes are connected. Then $k = Np_c$ keys are stored in each node instead of $N-1$ keys. Looking the other way round, the maximum size of the network that can be supported is $N = k/p_c$. The pairwise keys scheme gives node-to-node authentication.

Key predistribution is then done as in the basic scheme. For finding the shared key each node broadcasts its identifier. All nodes within communication range which share keys with the broadcasting node carry on a cryptographic handshake

thereby forming a secure communication link. The process of broadcasting can also be extended to other nodes with the help of intermediate nodes which share common keys with the broadcasting nodes. Revocation of node is done via voting. If node U_i finds that a certain node U_j is compromised, then it casts a public vote against it. If the number of such votes increase beyond a threshold t, then node U_i disconnects its communication link with U_j. This process continues throughout the network unless the node U_j is completely deleted. The voting method must have some properties like broadcasting public votes without replay value, disallowing a voting member from forging another vote, providing a means for each voting member to verify the validity of the votes that are being broadcasted. In this regard it is important to set the proper threshold value t. If t is high, there may not be enough neighboring nodes to revoke a node that has been compromised. If t is low, a group of compromised nodes may cause the revocation of many legitimate nodes. The other problem while broadcasting a public vote may lead to denial of service attack. To solve this problem, only voting members should be required to rebroadcast votes between each other, while the remaining nodes are forced to ignore the communication.

The advantage of this scheme is that it provides node revocation, the disadvantage is that the network is not scalable.

A modification of this scheme is the closest pairwise key scheme of Liu and Ning discussed in 2.8.1

2.4.2 Liu-Ning-Li polynomial-pool-based key predistribution

The pairwise key scheme of Liu and Ning [91] (later extended by Liu, Ning and Li [96]) using polynomial-pool-based key predistribution is based on the scheme given by Blundo et al [10]. The keys in the sensor nodes are generated from the the subset of polynomials in the pool. Two sensor nodes share some secret key if the keys are generated from the same polynomial. The polynomial scheme has been used in two different ways : a grid-based predistribution scheme and a hypercube-based key predistribution scheme.

According to the polynomial based scheme, a key pool is generated which consists of a set of $\mathcal{F}$ bivariate t-degree polynomials $f(x,y) = \sum_{i,j=0}^{t} a_{ij}x^i y^j$ over a finite field F_q, where q is a prime number such that the function has the property that $f(x,y) = f(y,x)$. A node U_i is given a set of $\mathcal{F}_i \subseteq \mathcal{F}$ of s' polynomials. The polynomial share given to node U_i is $f(i,y)$, where $f \in \mathcal{F}_i$. If node U_i wants to find the common key with node U_j, then U_i finds if they have shares of the same polynomial $f(x,i)$. Then node U_i evaluates the function $f(x,i)$ at the point j as $f(i,j)$. Similarly U_j computes the function at the point i. The $f(i,j) = f(j,i)$ is the common key between the nodes U_i and U_j.

According to this approach, each key occupies $(t+1)\log q$ storage space. The probability that two node share a common key is

$$p_c = 1 - \prod_{i=0}^{s'-1} \frac{|\mathcal{F}| - s' - 1}{|\mathcal{F}| - i}$$

The scheme is t collusion resistant meaning that a minimum of t nodes have to be compromised to compromise the entire network. The probability that a key is compromised is given by

$$p_{comp} = p_c \times P_{cd} + (1 - p_c)[1 - (1 - p'_c)(1 - P_{cd})^2] \text{ where}$$
$$P_{cd} = 1 - \sum_{i=0}^{t} \frac{s!}{(s-i)!i!} \left(\frac{s'}{|\mathcal{F}|}\right)^i \left(1 - \frac{s'}{|\mathcal{F}|}\right)^{s-i},$$

where s is the number of compromised nodes, p_c is the fraction of direct links, $1 - p_c$ is the fraction of indirect links and $p'_c = s/N$.

The hypercube based key predistribution is a modification of the polynomial based scheme. Given a total of N sensor nodes in the network, the hypercube-based scheme constructs an n-dimensional hypercube with m^{n-1} bivariate polynomials arranged for each dimension j,
$\{f_{\langle i_1,\ldots,i_{n-1}\rangle}(x,y)\}_{0\le i_1,\ldots,i_{n-1}<m}$, where $m = \lceil \sqrt[n]{N} \rceil$. The setup server then assigns each node in the network to a unique coordinate in this n-dimensional space. For the sensor node at coordinate $(j_1,\ldots,j_n)$, the setup server predistributes the polynomial shares of $\{f^1_{\langle j_2,\ldots,j_n\rangle}(x,y),\ldots,f^n_{\langle j_1,\ldots,j_{n-1}\rangle}(x,y)\}$ to this node. As a result, sensor nodes can perform share discovery and path discovery using this predistributed information. The authors call this scheme a grid-based design when $n = 2$.

2.4.3 Probabilistic scheme of Zhu et al

In another scheme [182], Zhu, Xu, Setia and Jajodia proposed a scheme for establishing pairwise keys using probabilistic key sharing [16] and threshold secret sharing [145]. This scheme enables any two nodes to establish a pairwise key on the fly, without the use of an online key predistribution center. Nodes find their pairwise keys only by knowing the key ids and no key identifier list is to be broadcasted. Hence the communication overhead is minimized.

Consider two nodes U_i and U_j that want to communicate. These nodes U_i and U_j may already have shared one or more keys from the pool of keys after the key predistribution phase. However, these keys are not known exclusively to U_i and U_j because every key in the key pool may be allocated to multiple nodes; hence, they cannot be used for encrypting any message that is private to U_i and U_j. The the goal of the algorithm is to establish a key, S, that is known exclusively to U_i and U_j. The basic idea underlying the establishment of such a key S is as follows: The sender node splits S into multiple shares using an appropriate secret sharing scheme. The sender then transmits to the recipient node all these shares, using a

different logical path for each share. The recipient node then reconstructs S after it receives all (or a certain number of) the shares. This is similar to the multipath key reinforcement of Chan, Perrig and Song [28].

Initially k keys are predistributed in each sensor from a key pool consisting of v keys. During pairwise key establishment, any node U_i randomly generates the secret key S and derives shares sk_1, sk_2, ..., sk_n from S. The random strings sk_1, sk_2, ..., sk_{n-1} are such that $|sk_1| = |sk_2| = \cdots = |sk_{n-1}| = |S|$ and $sk_n = S \oplus sk_1 \oplus sk_2 \oplus \cdots \oplus sk_{n-1}$, where $\oplus$ is the XOR operation. Node U_i transmits all the shares using a separate secure logical path to node U_j. Node U_j then reconstructs S from the n received shares. Node U_j sends back node U_i a HELLO message, authenticated with S as the MAC key. Node U_i verifies the HELLO message. Key establishment is done if the HELLO message is correct; otherwise node U_i aborts the process or tries again with a different set of logical paths after a certain time period.

There are three classes of logical paths used for pairwise key establishment.

Class C_1: The Class C_1 includes all the direct paths. Consider two nodes U_i and U_j. Let $z_1 = |P(U_i) \bigcap P(U_j)|$. sk_1 be the share generated by node U_i for class C_1. To deliver sk_1, node U_i computes the XORed key $k_{enc} = XOR\ \delta_l$, $\forall \delta_l \in P(U_i) \bigcap P(U_j)$, then encrypts (with appropriate authentication) sk_1 with k_{enc}.

Class C_2: The second class, denoted by C_2, includes indirect logical paths that use an intermediate node on the physical path between the two nodes as a proxy. Let node U_x be a proxy node for node U_i and node U_j, as long as $P(U_i) \bigcap P(U_x) = \emptyset$ and $P(U_x) \bigcap P(U_j) = \emptyset$. Suppose there are s_1 and s_2 keys in $P(U_i) \bigcap P(U_x)$ and $P(U_x) \bigcap P(U_j)$ respectively and $z_s = min(s_1, s_2)$, then the number of keys in $P(U_i) \bigcap P(U_x)$ and $P(U_j) \bigcap P(U_x)$ used to encrypt a share is z_s. More specifically, (i) node U_i generates a new secret s_{k_x}, (ii) it then (randomly) selects z_s keys in $P(U_i) \bigcap P(U_j)$ to compute the XORed keys k_{ix}^1, (iii) it encrypts (with appropriate authentication) s_{k_x} with k_{ix}^1, (iv) it sends the encrypted share to node U_x. Node U_x decrypts s_{k_x}, re-encrypts it with the XORed key k_{xj} computed using any z_s keys in $P(U_x) \bigcap P(U_j)$ and sends the result to node U_j. Since node U_x is on the physical path from U_i to U_j, no extra message overhead is incurred in the use of such proxies.

Class C_3 Class C_3 includes the indirect logical paths that use nodes that do not belong to class C_2, i.e., nodes that are not on the path from U_i to U_j. Both the neighbors of the source node and the neighbors of the destination node are potential proxy nodes for logical paths in class C_3. The source node can discover the neighbors of the destination node via an explicit message exchange with the destination. Alternatively, it may be possible to extend the underlying routing protocol to provide this information to the source at the time of route formation.

The advantages of this approach is that the communication overhead is very low, since only the identifiers of the nodes need to be broadcasted to find the common keys. The disadvantage is that if some node in a path is compromised,

creation of shares and pairwise key establishment process has to be started afresh.

2.5 Grid-based predistribution schemes

2.5.1 PIKE scheme of Chan and Perrig

In many schemes a path between two nodes is established where the trusted intermediary is the base station. This leads to the problem that the nodes close to the base station forward most of the information and continually loses battery power. As a result these nodes die out before the other nodes in the network. To overcome this problem Chan and Perrig proposed the PIKE scheme [27]. PIKE stands for Peer Intermediaries for Key Establishment. According to this scheme keys are established between nodes such that other sensor nodes act as trusted intermediaries. PIKE achieves an overhead of $O(\sqrt{N})$. PIKE establishes keys between any two nodes regardless of the network topology or node density.

Suppose the maximum number of nodes in the network is N. In PIKE the node identifiers are arranged in a square grid structure having a dimension $\sqrt{N} \times \sqrt{N}$. Any node is represented by (x, y) the coordinates of its position in the grid. The deployment order of the nodes is $(0,0), (0,1), (0,2), \ldots, (0, \sqrt{N}-1)$, $(1,0), (1,1), (1,2), \ldots, (1, \sqrt{N}-1)$, and so on. Each node (x, y) is then loaded with a secret pairwise keys with the nodes which lie in the xth row and yth column. Each key is unique and shared only between two nodes, hence they are called *pairwise* keys. Each node thus stores $2(\sqrt{N}-1)$ keys. Suppose two nodes U_A and U_B having identifiers (x_A, y_A) and (x_B, y_B) respectively want to communicate. If U_A and U_B are either in the same row or same column, then they share a pairwise key and can thus communicate directly. If they lie in different row and column, then they can establish a communication between each other through two nodes U_1 and U_2, such that U_1 is in the same row as U_A and same column as U_B and U_2 is in the same column as U_A and same row as U_B. Thus for every pair of nodes, there are two intermediaries. The intermediary is so chosen such that $m = argmin_{i \in \{1,2\}} d(A,i) + d(i,B)$, where $d(F,G)$ is the physical distance between the two nodes U_F and U_G. Suppose two nodes U_A and U_B communicate with each other via an intermediary U_C, then the following sequences take place.

1. $U_A \rightarrow U_C : E_{K_{AB}}\{U_A, U_B, K_{AB}\}, MAC_{K_{AC}}(E_{K_{AC}}\{U_A, U_B, K_{AB}\})$
2. $U_C \rightarrow U_B : E_{K_{BC}}\{U_A, U_B, K_{AB}\}, MAC_{K_{BC}}(E_{K_{BC}}\{U_A, U_B, K_{AB}\})$
3. $U_B \rightarrow U_A : E_{K_{AB}}\{U_A, U_B, N_B\}, MAC_{K_{AB}}(E_{K_{AB}}\{U_A, U_B, N_B\})$

$MAC_K(\mathcal{M})$ is the Message Authentication Code associated with message $\mathcal{M}$, K_{ij} is the pairwise key between U_i and U_j. N_B is the nonce generated by node U_B.

Chan and Perrig further reduce the storage space to contain $\sqrt{N}$ keys each. They also extend the results to a three dimensional grid structure of node IDs. In the paper the communication structure is implemented as GPSR [79] using a modified GHT [128]. The scheme has the advantage that it has high resiliency and the deterministic nature of key predistribution guarantees that two nodes will be able to establish a common key. The disadvantages of PIKE are that the communication overhead is high. A large fraction of keys do not share common keys and path must be established through intermediary nodes which may be time consuming.

2.5.2 Kalindi et al's Scheme

Another grid-based key predistribution scheme was proposed by Kalindi, Kannan, Iyengar and Durresi [75] which is a modification of PIKE. In PIKE the number of keys assigned to each sensor is $2(\sqrt{N}-1)$. First a $\sqrt{N}\times\sqrt{N}$ grid G is constructed. This grid is now divided into $l \times l$ cells each containing $m \times m$ keys, where $m = \lfloor\sqrt{N}/l\rfloor$. Let K_{ij} denote a unique key placed at position (i, j) on the grid. Let U_{ij} denote a node in position (i, j) on the grid, C_{xy} represent a cell in the grid, SG_{xy} represent the grid formed by the cell C_{xy} and its 8 adjacent cells and V_{ij} represent the key vector for a node U_{ij} in SG_{xy}. Then the key vector for a node U_{ij} is

$$V_{ij} = \bigcup k_{i,c'} + \bigcup k_{r',j},$$

where,
$(y-1) \mod k \times m < c' < (y+1) \mod l \times m$ and
$(x-1) \mod k \times m < r' < (x+1) \mod l \times m$.

Thus the number of keys assigned to each sensor is $6\sqrt{N}/l - 1$. This is much less than $2(\sqrt{N}-1)$. The number of keys shared by two nodes $U_{i_1j_1}$ and $U_{i_2j_2}$ is

$$\begin{cases} 3\sqrt{N}/l \text{ if } U_{i_1j_1}, U_{i_2j_2} \text{ are in the same cell and } i_1 = i_2 \text{ and } j_1 = j_2 \\ 2\sqrt{N}/l \text{ if } U_{i_1j_1}, U_{i_2j_2} \text{ are in cells which have common sides and } i_1 = i_2 \text{ and } j_1 = j_2 \\ 2 \quad \text{if } U_{i_1j_1}, U_{i_2j_2} \text{ are adjacent and } i_1 \neq i_2 \text{ and } j_1 \neq j_2 \\ 0 \quad \text{otherwise} \end{cases}$$

The choice of l is of great importance. If l is high, the size of key ring is small. This results in lower connectivity and increased security. The opposite happens when l is low. During shared-key discovery phase, the nodes exchange their node-ids to determine the number of keys they share. This can be done as the node identifiers can be used to determine the cell of the node that identifies the key vector of the node. Only neighboring nodes that share exactly two keys are allowed to securely communicate with each other by establishing a common shared key to form a direct link. Nodes belonging to the same cell and in the same row or column share more number of keys. However these two nodes are not allowed to use the common keys because capturing a single node in that row or column reveals those keys.

On completion of the key discovery phase all the neighboring nodes may not have established common shared keys. In order that a node establishes keys with non-key-neighbors, it must go through the path-key establishment phase. In this phase, a node searches among its key-neighbors recursively to find a key-path to the non-key-neighbor.

The number of nodes to be captured to compromise a single link is given by

$$N_c = \frac{\sum x N_{capt}(x)}{\sum N_{capt}(x)},\ 2 \leq x \leq N$$

where

$$N_{capt}(x) = \binom{N}{y} - 2\binom{N-6m+1}{x} + \binom{N-12m+4}{x}.$$

The probability that a link is compromised when x nodes are captured is given by

$$P(x) = 1 - \frac{2\binom{N-6m+1}{x} - 2\binom{N-12m+4}{x}}{\binom{N}{x}}.$$

2.5.3 Sadi-Kim-Park Scheme

Sadi, Kim and Park [139] proposed another grid-based random scheme based on bivariate polynomials which they called GBR (Grid-Based Random) key predistribution scheme. In GBR, a $m \times m$ grid is considered where $m = \sqrt{N}$, and N is the number of sensors. The setup server generates $2m\omega$ bivariate polynomials $F = \{f_i^c(x,y), f_i^r(x,y)\}_{i=0,1,2,\ldots,m\omega-1}$ of degree t over a finite field F_q. From these polynomials, the server makes m group of polynomials $f_{\omega_i}^c(x,y)_{i=0,1,2,\ldots,m-1}$ which are assigned to each column c and $f_{\omega_i}^r(x,y)_{i=0,1,2,\ldots,m-1}$ polynomials which are assigned to each row r.

For each node (i,j) a set of 2τ polynomials are randomly selected from the ith row and jth column and assigned to the sensor.

To find the shared key two nodes U_{r_i,c_i} and U_{r_j,c_j} first broadcast their ids. If $r_i = r_j$ or $c_i = c_j$, then the nodes use a challenge response protocol to find out if they share a common polynomial. If they do not share, they execute a path discovery process. During the path discovery process two nodes U_{r_i,c_i} and U_{r_j,c_j} find two other intermediate nodes U_{r_i,c_k}, U_{r_j,c_k} such that a path is established. There are up to $2m-2$ pairs of such nodes in the grid.

The percentage of nodes which can establish a pairwise key directly is $2/(m+1)$.

The probability that a polynomial is compromised is

$$P_c = 1 - \sum_{t=0}^{l} \frac{s!}{(s-l)!l!}(\frac{\tau}{m\omega})^l(1-\frac{\tau}{m\omega})^{s-l},$$

where s is the number of compromised nodes and l is the number of times the polynomial is chosen among the s compromised nodes.

2.5.4 Mohaisen-Maeng-Nyang scheme

Mohaisen, Maeng and Nyang [110] introduced a grid-based 3-dimensional grid-based scheme in which each sensor node correspond to a grid-intersection point. A $m \times m \times m$ grid is considered where $m = \lceil \sqrt[3]{N} \rceil$. $3\sqrt[3]{N}$ symmetric polynomials of degree t with coefficients in F_q are assigned corresponding in such a way that all the plat with the same axis value owns the same corresponding polynomial. Each node is assigned three polynomials depending on the coordinate of its location on the grid. Each sensor U_i having coordinates $< c_x, c_y, c_z >$ is assigned a polynomial $< f_{c_x}, f_{c_y}, f_{c_z} >$. For each sensor node U_i with polynomials $< f_{c_x}, f_{c_y}, f_{c_z} >$ is assigned shares $g_{c_x} = f_{c_x}(i, y)$, $g_{c_y} = f_{c_y}(i, y)$, $g_{c_z} = f_{c_z}(i, y)$. These shares are loaded into node U_i's memory.

For direct key establishment two nodes $i =< c_x||c_y||c_z >$ and $j =< c_x||c_y||c_z >$checks if $i.c_x = j.c_x$ or $i.c_y = j.c_y$ or $i.c_z = j.c_z$ which, means if they share any common polynomial. If it is so, then from the polynomial the common key can be calculated. If a common polynomial cannot be found, then a path is to be established containing one or more intermediate polynomials.

The connectivity of this scheme is $\frac{3}{m+1}$. The memory requirement is $M = 3((N_c+1)\lceil \log(N^{1/3}) \rceil + (t+1)\log(q))$. The communication overhead is $1.5\lceil \log N \rceil$. The fraction of nodes compromised when a single plat is compromised is

$$p_c = 1 - \sum_{i=0}^{t} \left(\frac{(N^{2/3})!}{i!(N^{2/3} - i)!} \right) (F_c)^i (1 - F_c)^{N^{2/3} - i},$$

where F_c is the fraction of compromised nodes and i is the number of compromised shares of a given polynomial. Though the communication overhead of this scheme is low, the resiliency is very poor.

2.6 Group-based key predistribution

In many applications for example when nodes are scattered from an airship, nodes may be deployed in groups. Sometimes we may know the approximate location of the sensor groups, but sometimes we may not know. However the knowledge that some nodes are close together than others help in devising key predistribution schemes which provide better connectivity and resiliency. Group-based key predistribution can be subdivided into two classes, one that uses deployment knowledge and one that do not use deployment knowledge. In this section we discuss group-based key predistribution schemes which do not use deployment. We discuss group-based schemes that use deployment knowledge in Section 2.8.

2.6.1 Liu-Ning-Du Scheme

Group-based key predistribution scheme without using deployment knowledge was proposed by Liu, Ning and Du [94, 95]. The authors propose framework of group-

based deployment scheme and present two key predistribution schemes which are very suitable in this framework. The first scheme is a *hash key-based scheme* and the other a *polynomial-based scheme*.

Liu, Ning and Du [95] observed that sensor nodes in the same group are usually close to each other after deployment, since they are thrown in one cluster. Each group of sensors that are deployed together form a *deployment group*. The sensor nodes to be deployed are divided into n groups each consisting of m sensor nodes. The sensor nodes that are deployed in the same deployment group G_i are deployed from the same place and at the same time with deployment index i. During deployment of any group G_i, the resident point in any sensor node in the group follows a probability distribution function(pdf) $f_i(x, y)$, which is called the *deployment distribution* of the group G_i. The deployment distribution for any node of the deployment group G_i follows a two-dimensional Gaussian distribution centered at the deployment point (x_i, y_i). The pdf for any group G_i is

$$f_i(x, y) = \frac{1}{2\pi\sigma^2} e^{-[(x-x_i)^2+(y-y_i)^2]/2\sigma^2} = f(x - x_i, y - y_i),$$

where σ is the standard deviation and $f_i(x, y) = \frac{1}{2\pi\sigma^2} e^{-[x^2+y^2]/2\sigma^2}$.

There is a predistribution scheme to establish pairwise keys within each group (called *in-group predistribution*) and a key predistribution method is used to establish pairwise keys in different deployment groups (called the *cross-group predistribution*). The in-group instance D_i can be any existing key predistribution technique. The m cross groups $\{G'_i\}_{i=1,2,\cdots,m}$ are constructed such that

1. each cross-group includes exactly one sensor node from the deployment group, ie, $|G'_i \bigcap G_j| = 1$, $i \neq j$

2. there are no common sensor nodes between any two different cross groups, ie, $G'_i \bigcap G'_j = \emptyset$.

A key predistribution instance is generated from cross group. The cross-group instance D'_i can be the instance of any key predistribution technique. Each deployment group G_i contains the sensor nodes with IDs $\{(i-1)m + j\}_{j=1,2,\ldots,n}$ and each cross group G'_i contains the sensor nodes with IDs $\{i + (j-1)m\}_{j=1,2,\ldots,n}$. This makes it easy to identify the in-group and cross group to which a node belongs given only its group ID. There are key distribution instances D_i and D'_i to distribute keys in in-group nodes and cross-group nodes.

For key establishment, given two nodes U_i and U_j we find if $\lfloor \frac{i}{m} \rfloor = \lfloor \frac{j}{m} \rfloor$. If this condition arises then the nodes U_i and U_j are in the same deployment group. If $i \mod m = j \mod m$, then the nodes U_i and U_j belong to the same cross group. If two sensor nodes are in the same deployment group G_i, they can follow the path key establishment in D_i. If two sensor nodes are in two different groups G_i and G_j then a bridge $< a, b >$is constructed such that $U_a \in G_i$ and $U_b \in G_j$ and U_a and U_b belong to the same cross group G'_k.

The path key establishment takes place as follows [95].

1. The source node U_i first tries the bridge involving itself to establish an indirect key with the destination node U_j. Assume this bridge is $< i, j' >$. Node U_i first sends a request to $U_{j'}$ if it can establish a direct key with U_j. If node $U_{j'}$ can also establish a (direct or indirect) key with the destination node U_j, node $U_{j'}$ forwards this request to the destination node v to establish an indirect key.

2. If the first step fails, node U_i will try the bridge involving the destination node U_j. Assume the bridge is $< i', j >$. In this case, node U_i sends a request to node $U_{i'}$ if it can establish a (direct or indirect) key with $U_{i'}$. If node $U_{i'}$ can establish a direct key with node U_j, it forwards the request to the destination node U_j to establish an indirect key. Note that if nodes U_i and U_j are in the same cross group, this step can be skipped.

3. When both of the above steps fail, node U_i has to try other bridges. It randomly chooses a bridge $< i', j' >$ other than the above two, assuming $U_{i'}$ is in the same deployment group with U_i, and $U_{j'}$ is in the same deployment group with U_j. Node U_i then sends a request to $U_{i'}$ if it can establish a (direct or indirect) key with $U_{i'}$. Once $U_{i'}$ receives this request, it forwards the request to $U_{j'}$ in the bridge if they share a direct key. If $U_{j'}$ can establish a (direct or indirect) key with the destination node U_j, it forwards the request to node U_j to establish an indirect key.

In the above approach, the path key establishment in a cross-group instance has never been used. The reason is that the sensor nodes in a cross group usually spread over the entire deployment field, which makes it expensive to perform path key establishment in a cross group. Thus it is interesting to find ideas that can discover a valid bridge by only contacting a few sensor nodes in the network.

Liu, Ning and Du [95] present two instantiations of predistribution, one using hash functions and the other using polynomials.

In the hash-based scheme before deployment, every sensor node U_i is predistributed with a master key K_i , known only by the trusted central server (e.g., base station) and the node U_i. Let G be either a deployment group or a cross group. Assume that the node IDs in G have already been sorted in an ascending order. For any sensor node $U_i \in G$, let Pos(i) be the position of this node in the ordered group G. For any two nodes U_i and U_j (Pos(i) $<$ Pos(j)) in this group, the value (Pos(i)+ Pos(j)) is checked. If it is an odd value (ie, (Pos(i)+Pos(j)) $\mod 2 = 1$), node U_j is predistributed $H(K_i||j)$; otherwise, node U_i is predistributed $H(K_j||i)$, where H is a one-way hash function. Due to the group construction method, the positions of a sensor node in its deployment group and cross group, respectively can be calculated. Specifically, given a sensor node ID i, the in group can be calculated as $\lceil i/m \rceil$ and the cross group can be calculated as $((i-1) \mod m)+1$. Therefore, in case of the deployment group, Pos(i) $= ((i-1) \mod m) + 1$; in the case of the cross group, Pos(i) $= \lceil i/m \rceil$.

For nodes U_i and U_j the shared key algorithm proceeds as follows.

- If Pos(i) <Pos(j) and (Pos(i)+Pos(j)) mod 2 = 1, node U_i can compute the shared key $H(K_i||j)$, while node U_j has already been predistributed with this key.
- If Pos(i) < Pos(j) and (Pos(i)+ Pos(j)) mod 2 = 0, node U_i has the predistributed key $H(K_j||i)$, while node U_j can compute this key easily.
- If Pos(i) > Pos(j) and (Pos(i)+ Pos(j)) mod 2 = 1, node U_i has the predistributed key $H(K_j||i)$, while node U_j can compute this key easily.
- If Pos(i) > Pos(j) and (Pos(i)+ Pos(j)) mod 2 = 0, node U_i can compute the shared key $H(K_i||j)$, while node U_j has already been predistributed with this key.

Since two nodes share a pairwise key, the scheme is perfectly secure. The number of keys in node U_i for a deployment group G is estimated as $n/2$. The total number of keys in node U_i for its deployment group and cross group is approximately as $(m+n)/2$. The second predistribution method proposed uses polynomials as done in Blundo et al's [10] scheme. For any group G (either a deployment group or a cross group), a unique symmetric t-degree bivariate polynomial $f_g(x,y)$ is generated with the property of $f_g(x,y) = f_g(y,x)$. Every sensor node $U_i \in G$ gets predistributed a polynomial share $f_g(i,x)$ by evaluating the bivariate polynomial $f_g(x,y)$ at $x = i$. It is assumed that the polynomial $f_g(x,y)$ is only known by a trusted server. To establish a pairwise key with node U_j, node U_i only needs to compute $f_g(i,j)$, which equals $f_g(j,i)$, the value that can be computed by node U_j as well. There is no additional communication involved.

For any node in group G_i , the probability of having direct keys with its neighbor nodes can be estimated as

$$p_d = \int\int_S f(x-x_i, y-x_i)p_i(x,y)dxdy,$$

where,

$$p_i(x,y) = \frac{(n_{i,i}(x',y')+n'_i(x',y'))}{n_A},$$
$$n'_i(x',y') = \frac{n_A - n_{i,i}(x',y')}{n},$$
$$n_i(x',y') = \frac{\sum_{j=1,j\neq i}^{n} n_{i,j}(x',y')}{n},$$
$$n_{i,j}(x',y') = n\int\int_A f(x-x_j, y-y_j)dxdy \text{ and}$$

A is a circle centered at (x',y') with radius R, where R is the signal range of a sensor node and n_A is the average number of sensor nodes in the communication range of a sensor node.

It has been shown in [95] that both of the instantiations result in the same probability of establishing a direct key between two neighboring nodes in the network since the probability of having a direct key between two sensor nodes in the

same group (either deployment group or cross group) for both instantiations is always 1.

The indirect key establishments depends on the establishment of bridges between groups. The probability of finding one bridge can be estimated by

$$p_i = 1 - (1 - \frac{n_d}{n})^2 \left(\frac{(n-2-n_d)!(n-2-n_d)!}{(n-2)!(n-2-2n_d)!} \right),$$

where,

$$n_d = \int \int_S f(x - x_i, y - y_i) n_d(x, y) dx dy, \text{ and}$$
$$n_d(x', y') = n \int \int_B f(x - x_i, y - y_i) dx dy,$$

B denote the area that is no more than $d \times R$ meters away from (x', y') and S denotes the entire deployment field.

The probability that any direct key between two non-compromised sensor nodes in G_i being compromised is

$$p_{cd} = 1 - \sum_{j=0}^{t} \frac{s!}{(s-j)!j!} \frac{(n-1)^{s-j}}{n^s},$$

where, s is the number of compromised sensors.

Overall, the probability of an indirect key between noncompromised sensor nodes being compromised can be given by

$$p_{ci}(s) = \frac{2p_1 + (m-2)p_2}{m},$$

where,

$$p_1 = 1 - (1 - p_{cd})^2 (1 - \frac{s}{nm-2}) \text{ and } p_2 = 1 - (1 - p_{cd})^3 (1 - \frac{s}{nm-2})^2.$$

The advantages of this scheme are that it does not do not use deployment knowledge and give resiliency and connectivity similar to the deployment knowledge based schemes. The polynomial based schemes can be made scalable. The framework can be used to improve any existing predistribution schemes.

The disadvantages of this scheme is that the probability of secure communication between cross-group neighbors is very less. The scheme is not suitable for networks which have small group size.

2.6.2 Martin-Paterson-Stinson's improvement of Liu et al's scheme

To alleviate the problems of Liu et al's [94,95] scheme, Martin, Paterson and Stinson [104] proposed a group based design using resolvable transversal designs. For definition of transversal design and resolvable design one may refer to Section 1.5. In their scheme a $TD(m, n)$ is considered. In order to increase the cross-group

connectivity, at the cost of additional storage; each node is then contained in m cross groups, rather than just one. The structure of the resolvable transversal design implies that each node is contained in precisely m cross groups, each cross group contains at most one node from each group, and two cross groups have at most one node in common.

Let λ be a divisor of n. The n parallel classes of the $TD(m, n)$ are partitioned into λ disjoint sets $S_1, S_2, \cdots, S_\lambda$, of n/λ parallel classes and set the blocks in the parallel classes of each set S_i is associated with the nodes of a node group. When node groups are formed by merging parallel classes in this fashion, each cross group contains more than one node (in fact, precisely n/λ nodes) from each node group. This is necessary to enable connectivity to be maintained without adversely affecting the storage when the number of groups is small. This is because the size of the groups is inversely proportional to the number of groups and, if we were to require each cross group to intersect each node group in precisely one point, then the number of cross groups required to ensure each node can establish keys with a given proportion q of nodes in other groups would grow with the size of the group.

The predistribution scheme can be described as below [104]. Given a prime power n and non-negative integers $1 \leq m \leq n$, $\lambda | n$ and $0 \leq t < n^2/\lambda - 1$, a $TD(m, n)$ is used to construct a KPS for a network of n^2 nodes deployed in λ groups $G_1, G_2, \cdots, G_\lambda$ of size n^2/λ as follows:

1. Partition the parallel classes P_1, P_2, ..., P_n of the $TD(m, n)$ into λ sets of n/λ parallel classes, denoted $S_1, S_2, \cdots, S_\lambda$. Each S_i contains n^2/λ blocks.

2. Associate each node with a block of the design, such that the nodes of group G_i correspond to the blocks in the parallel classes contained in set S_i .

3. Associate a degree t Blom's scheme with each point of the design, and assign shares in a given scheme to each node whose corresponding block contains the relevant point. (Note that resiliency is maximized by taking $t = n^2/\lambda - 2$, which is equivalent in terms of storage and resiliency to assigning a distinct key to each pair of nodes involved in the scheme. As such it is never necessary to consider $t \geq n^2/\lambda - 1$.)

4. Associate a degree t Blom's scheme with each parallel class P_i , for $1 \leq i \leq n$, and assign shares in a given scheme to each node whose corresponding block is contained within that parallel class.

From the above design the following observations can be made.

1. Each node stores the equivalent of $(m + 1)(t + 1)$ keys.

2. For each instance of Bloms scheme, there are n nodes possessing shares in that scheme.

3. Probability q that two nodes from different groups share a key is k/n

4. Probability p that two nodes from the same groups share a key is $\frac{n-1}{n^2/\lambda-1} + \frac{n^2/\lambda-n}{n^2/\lambda-1}mn$

5. Probability that a uncompromised link is broken when s nodes are compromised is $fail(s) = \begin{cases} 0 & \text{if } s \leq t, \\ 1 - \sum_{i=0}^{t} \frac{\binom{n-2}{i}\binom{n^2-n}{s-i}}{\binom{n^2-2}{s}} & \text{if } s > t \end{cases}$

On one hand this scheme alleviates the problem faced with Liu, Ning Du's scheme [95]. However the resolvable designs can be applied where such designs exists. No particular algorithm is given for the construction of such designs for a given set of parameters.

2.7 Key predistribution using combinatorial structures

Mitchell and Piper [109] were the first to apply combinatorial designs in key distribution. Combinatorial designs were used for the first time in key predistribution by Çamtepe and Yener [18], which was modified by them [20].

Recall the definition of a set system $(X, \mathcal{A})$ and $BIBD(v, b, r, k, \lambda)$ from Section 1.5. We can map this to a sensor network, X is the pool of key identifiers. So there are v keys in the key pool. The network can support a maximum of b nodes and each node contains k keys. Each key occurs in r nodes and any pair of keys occur in λ nodes. Different authors have used different combinatorial structures in devising key predistribution schemes. The pioneering work of Çamtepe and Yener [18, 20] used projective planes and generalized quadrangles, Lee and Stinson [86, 89] used transversal designs, Chakrabarti, Maitra and Roy [23, 24] used merging blocks constructed from transversal designs. $3-designs$ [44] and orthogonal arrays [45] was proposed by Dong, Pei and Wang, *Costas arrays* and distinct difference configuration were proposed by Blackburn, Etzion, Martin and Paterson [7, 8], Expander graphs by was proposed Çamtepe, Yener and Yung [21] and product construction was presented by Wei and Wu [171]. It is seen that key predistribution using combinatorial designs has several advantages. By properly choosing the design resiliency can be improved. Again it can be seen that some designs result in sensor networks where every node is connected to each other. The most important point in favor of using combinatorial designs is that because of the pattern inherent in the designs efficient shared key discovery algorithms can be devised.

We now discuss each of the schemes in details, presenting the merits and demerits of each.

2.7.1 Çamtepe and Yener's scheme

As already mentioned, Çamtepe and Yener [18, 20] were the first to use combinatorial designs for key predistribution in WSN. The authors used projective planes and generalized quadrangles for predistribution. We note that a finite projective plane $PG(2, q)$ (q is a prime power) is the same as the symmetric BIBD, $BIBD(q^2 + q + 1, q^2 + q + 1, q + 1, q + 1, 1)$. Let the nodes (or blocks) be indexed by (a, b, c) where $a, b, c \in GF(q)$. The nodes are given by the identifiers $(1, b, c)$, $(0, 1, c)$ and $(0, 0, 1)$, where $b, c \in GF(q)$. So there are a total of $q^2 + q + 1$ nodes. Similarly the keys are indexed by (x, y, z) where $x, y, z, \in GF(q)$. The identifiers of the keys are given by $(x, y, 1)$, $(x, 1, 0)$ and $(1, 0, 0)$, where $x, y \in GF(q)$. So there are a total of $q^2 + q + 1$ keys (or elements). A key (x, y, z) is assigned to node (a, b, c) if $ax + by + cz = 0$.

The other predistribution scheme involves generalized quadrangles. Three known designs for generalized quadrangles have been used : $GQ(q, q)$, $GQ(q, q^2)$ and $GQ(q^2, q^3)$. The construction of $GQ(q, q)$, $GQ(q, q^2)$ and $GQ(q^2, q^3)$ have been done from $PG(4, q)$, $PG(5, q)$ and $H(4, q^2)$ respectively. Details can be found in [20, Section B] We discuss the shared key discovery for the scheme obtained by $GQ(q, q)$. Although the first scheme using symmetric design results in full connectivity of the network, the resiliency is poorer than the second scheme using generalized quadrangles.

2.7.2 Lee and Stinson's schemes

Lee and Stinson [86] formalized the definition of key predistribution schemes using set systems. They introduced the concept of common intersection designs [88]. They discussed the use of block graphs for sensors, since by this design every pair of nodes is connected by a maximum of two hop path. They also used transversal designs for key predistribution. They used the following construction of a transversal design $TD(k, r)$ [86].

1. $X = \{(x, y) : 0 \leq x < k, 0 \leq y < r\}$,
2. For all i, $G_i = \{(i, y) : 0 \leq y < r\}$,
3. $\mathcal{A} = \{A_{i,j} : 0 \leq i < r \ \& \ 0 \leq j < r\}$.

We define a block $A_{i,j}$ by

$$A_{i,j} = \{(x, xi + j \bmod r) : 0 \leq x < k\} \tag{2.7.1}$$

Each of the r^2 nodes (r is a prime power) is assigned a set of k keys, in such a way that the (i, j)th node receives the keys $\{(x, xi + j \bmod r) : 0 \leq x < k\}$. The design is such that two nodes share 0 or 1 common keys. The expected number of common keys between two nodes is given by $p_1 = \frac{k(r-1)}{b-1} = \frac{k}{r+1}$. When two

nodes want to communicate they broadcast their identifiers (i_1, j_1) and (i_2, j_2). The nodes can communicate if they share some common key. This happens when $xi_1 + j_1 = xi_2 + j_2 (\mod r)$, such that $x = (j_2 - j_1)(i_1 - i_2)^{-1} (\mod r) < k$. This shared key can thus be found only by computing an inverse which can be done in $O(1)$ time. The communication overhead is $O(\log r) = O(\log \sqrt{N})$, where N is the size of the network. They also gave a estimate of the resiliency of the network in terms of the proportion of links disconnected. Suppose s nodes are compromised, then the probability that a link is broken is given by $fail(s) = 1 - (1 - \frac{r-2}{b-2})^s$. Considering the example given in [86] we consider a configuration $v = 1470$, $b = 2401$, $r = 49$, $k = 30$, $p_1 = 0.6$, $fail(10) = 0.1795$.

A generalization of the above scheme was made by Lee and Stinson [89]. A $TD(t, k, p)$ (where p is prime and $t \leq k \leq p$) is constructed in the following way.

Let $X_i \subseteq X$, $|X_i| = k$,

$$X = X_1 \times \mathbb{F}_p.$$

For $x \in X_1$ define

$$H_x = \{x\} \times \mathbb{F}_p$$

and define

$$\mathcal{H} = \{H_x : x \in X_1\}.$$

For every ordered t-tuple $\mathbf{c} = (c_0, c_1, \ldots, c_{t-1}) \in (\mathbb{F}_p)^t$, define a block

$$A_c = \{(x, \sum_{i=0}^{t-1} c_i x^i) : x \in X_1\} \text{ and}$$
$$\mathcal{A} = \{A_c : \mathbf{c} \in (\mathbb{F}_p)^t\}.$$

A_c are the blocks of the design. We can see that for $t = 2$ the construction is same as given in [86]. In [84] a quadratic scheme with $TD(3, k, p)$ is considered.

Let p be a prime, and let F_p be a finite field of order p. We define a set system $(X, \mathcal{A})$ such that $X = D \times F_p$, $D = \{0, \cdots, k-1\}$, and

$$\mathcal{A} = \{A_{a,b,c} : a, b, c \in F_p\},$$
$$\text{where } A_{a,b,c} = \{(x, ax^2 + bx + c) : x \in D\}.$$

Local Connectivity through direct connection between nodes is given by $Pr_1 = \frac{k(k-1)}{2(p^2+p+1)}$. Connectivity through two-hop paths between nodes is given by $Pr_2 = (1 - \frac{k(k-1)}{2(p^2+p+1)})(1 - (1 - \frac{\mu_2}{p^3-2})^\eta)$. Resiliency is given by $fail(s) = 1 - 2(1 - \frac{p^2-2}{p^3-2})^s + (1 - \frac{2p^2-p-2}{p^3-2})^s$.

The resiliency of the linear scheme is better than the quadratic scheme. A multiple space scheme has also been presented in [89].

2.7.3 Chakrabarti-Maitra-Roy Scheme

Chakrabarti, Maitra and Roy [23,24] proposed a hybrid key predistribution scheme by merging the blocks in combinatorial designs. They considered the blocks constructed from the transversal design proposed by Lee and Stinson and randomly selected them and merged them to form the sensor nodes. Though this scheme increases the number of the keys per node, it improves the resiliency of the network. The probability that two nodes share a common key is also more. Thus it has a better connectivity. They also presented a basic heuristic for this and showed that it provides slight improvement in terms of certain parameters than their basic random merging strategy. Using the merging strategy the following theorem can be stated.

Theorem 2.7.1. *Consider a Transversal design $TD(r,k)$. Let $v = rk$, $b = r^2$ and z blocks be randomly selected to form a sensor node. Then*

1. *There will be $N = \lfloor \frac{b}{z} \rfloor$ 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 distinct keys, where $zk - \binom{z}{2} \leq M \leq zk$. The average value of M is $\hat{a} = zk - \binom{z}{2}\frac{k}{r+1}$.*
5. *The expected number of links in the merged system is $\hat{L} = (\binom{r^2}{2} - \binom{z}{2}\lfloor \frac{r^2}{z} \rfloor)\frac{k}{r+1} - (r^2 \mod z)k$.*
6. *Each key will be present in Q nodes, where $\lceil \frac{r}{2} \rceil \geq Q \geq r$. The average value of Q is $\hat{Q} = \frac{1}{kr}(\lfloor \frac{b}{z} \rfloor)(zk - \binom{z}{2}\frac{k}{r+1})$.*
7. *Given that s nodes are compromised the fraction of links broken is given by*
   $$Fail(s) = \frac{\sum_{i=1}^{z^2} \frac{\binom{\gamma}{i}}{\binom{kr}{i}} \binom{z^2}{i} (\frac{k}{r+1})^i (1-\frac{k}{r+1})^{z^2-i}}{1-(1-\frac{k}{r+1})^2}$$
   where $\gamma = szk(1 - \frac{sz-1}{2(r+1)})$.

In this scheme each of the nodes has the identifiers of the blocks which form that node. These identifiers are stored as a list in the node. To find a shared common key between two blocks, say U_1 and U_2, U_1 compares each identifier with z identifiers of U_2. Suppose the identifiers are (i_a, j_a) and (i_b, j_b). Then a common key between U_1 and U_2 exists provided $x = (j_a - j_b)(i_b - i_a)^{-1} \mod r$ exists and is less than k. For this, $O(z)$ bits must be broadcasted by each node and z^2 inverse calculations are done. So the overall time complexity of the algorithm is $O(z^2)$.

2.7.4 Dong, Pei and Wang's scheme

The key predistribution scheme proposed by Dong et al in [44] makes use of $3-$ *designs*. In particular they use inversive planes to assign keys in the sensor nodes. Let q be a prime. We use an irreducible polynomial $f(x)$ of order 2 to construct a field $F_{q^2} = Z_q/(f(x))$. Let $f(x) = x^2 + f_1'x + f_0'$.

Let the field elements be $f_0 = 0$, $f_1 = 1$, $f_2, \cdots, f_{q^2-1}$. We choose $a, b, c, d \in F_{q^2}$, such that $ad - bc \neq 0$. Let $\infty \notin F_q$. We define a function

$$\pi\begin{pmatrix} a & b \\ c & d \end{pmatrix}(x) = \begin{cases} \frac{ax+b}{cx+d} & \text{if } x \in F_q \text{ and } cx + d \neq 0 \\ \infty & \text{if } x \in F_q,\ cx + d = 0 \text{ and } ax + b \neq 0 \\ \frac{a}{c} & \text{if } x = \infty \text{ and } c \neq 0 \\ \infty & \text{if } x = \infty,\ c = 0 \text{ and } a \neq 0 \end{cases}$$

Let $PGL(2, q^2)$ denote all distinct permutations $\pi\begin{pmatrix} a & b \\ c & d \end{pmatrix}$, where $a, b, c, d \in F_{q^2}$, such that $ad - bc \neq 0$. It can be proved as in [154, Lemma 9.25] that there are $q^6 - q^2$ such permutations.

Blocks are created in the following way. For each permutation π_i, $(i = 0, 1, 2, \cdots, q^6 - q^2 - 1)$ block B_{π_i} consists of elements $\pi_i(j)$, $j = 0, 1, \cdots, q-1, \infty$. So each block consists of $q + 1$ elements. The resulting design gives rise to a $3 - (q^2 + 1, q + 1, 1)$ design.

We consider the distinct blocks and map the blocks to the nodes and preload each node with the keys contained in that particular block. Since the number of distinct blocks is $q^3 + q$, the number of nodes supported by the network is $q^3 + q$. Let the key chain belonging to node U_i be denoted by $\{k_t^{(i)} : 0 \leq t \leq q\}$. Any two nodes can share at most two keys. The disadvantage of this method is that resiliency reduces drastically as the number of nodes compromised increases.

2.7.5 Product construction of Wei and Wu

Du et al [48] and Liu and Ning [91] had proposed a key distribution scheme which is an extension of Blom scheme. Wei and Wu [171] formalize their method. They use construction from set systems which optimize the product construction of [48, 91]. The scheme is based on the product of key distribution scheme and set systems. They deduce conditions of the set systems that provide optimum connectivity and resiliency of the network.

2.7.6 Combinatorial key predistribution using deployment knowledge

Till now we have considered networks in which nodes are placed randomly in the deployment area. Now we look at some schemes which employ deployment knowledge. There are many application where sensors are placed in grid structure.

Blackburn et al used Costas arrays and distinct-difference configuration to predistribute keys in sensor nodes. This results in a very high resiliency. However their designs are applicable given that suitable Costas Arrays and Distinct Difference Configurations exists. Hence they proposed the constructions of several Distinct Difference Configuration in [8].

Another grid-based deployment scheme was proposed by Chakrabarti [22]. Keys are distributed following the transversal design as done by Lee and Stinson. The nodes are placed in grids such that node (i, j) is placed at the (i, j) th intersection point in the grid.

2.8 Key predistribution using Deployment knowledge

Sometimes sensor nodes may be dropped from a airplane. The sensors are first grouped together. These groups are then dropped sequentially as the plane moves forward. The sensors that are in the same group have a high probability of being deployed close to each other. Although it may not be possible to precisely pinpoint sensor's positions, it is often possible to approximately determine their locations.

There are several schemes available in literature, which employ deployment knowledge during key predistribution (eg: [49, 69, 70, 92, 149, 174, 179]). According to these schemes the sensors are deployed according to some given pattern. The pattern is then exploited in key predistribution of keys. We discuss each of these schemes briefly.

2.8.1 Liu-Ning Scheme

Location-based key predistribution were first introduced by Liu and Ning [92]. They proposed two predistribution schemes both of which took advantage of the deployment knowledge of sensor nodes. The first scheme called the *closest pairwise scheme* was a modification of the pairwise key predistribution scheme. For any node U_i, the setup server establishes pairwise keys with sensor nodes which are close to U_i. The key establishment takes place as done in [28]. When nodes need to be added to the network, pairwise keys are established with nodes close to the new node. Revocation also takes place as given in [28].

The second predistribution scheme uses the polynomial-based key predistribution scheme of Blundo et al [10]. The deployment region is broken down into equal sized squares $\{C_{i_c,i_r}\}_{i_c=0,1,...,C-1,i_r=0,1,...,R-1}$, each of which is a *cell* with coordinates (i_c, i_r) denoting row i_r and column i_c. Each of the cells is associated with a bivariate polynomial. For a $R \times C$ grid the setup server generates RC t-degree polynomials $\{f_{i_c,i_r}(x, y)\}_{i_c=0,1,...,C-1,i_r=0,1,...,R-1}$, and assigns $f_{i_c,i_r}(x, y)$ to cell C_{i_c,i_r}.

For each sensor the setup server determine its *home cell* and its four neighboring cells which lie adjacent to the home cell in the same row and column. The setup

server distributes to the sensor the coordinates of the home cell and the polynomial shares of the home cell and its neighboring cell. For example for a sensor U_u in the cell with coordinate (r', c'), the polynomial shares $f_{r'-1,c'}(u, y)$, $f_{r',c'-1}(u, y)$, $f_{r'+1,c'}(u, y)$, $f_{r',c'+1}(u, y)$, $f_{r,c}(u, y)$ are given. For direct key establishment a node broadcasts the coordinates of its home cell. From this coordinate the destination node finds out the common polynomial that it shares with the broadcasting node if at all. Now the common key can be calculated using the same method as [28].

2.8.2 Du et al's Scheme

Almost independently Du et al proposed a key predistribution scheme using deployment knowledge in [47], which they extended in [49]. This scheme uses a grid-group based deployment scheme in which sensors are deployed in groups, such that a group of sensors are deployed at a single deployment point, and the pdf (probability distribution functions) of the final resident points of all the sensors in a group are the same. The key predistribution scheme uses multiple space Blom scheme as in [48, 50]. The deployment model is as follows [49]

1. N sensors are divided into $t \times n$ equal sized group, such that each group $G_{i,j}$ for $i = 1, 2, \ldots, t$, $j = 1, 2, \ldots, n$ is deployed from the deployment point with index (i, j).

2. The deployment points are arranged in a grid

3. During deployment, the resident points of the node U_l in group $G_{i,j}$ follow the pdf $f(x, y|U_l \in G_{i,j})$. Any node r in the group $G_{i,j}$ follows a two dimensional Gaussian distribution. For a point (x_1, y_1), $f(x, y|U_l \in G_{i,j})$ is given by

$$f(x, y|U_l \in G_{i,j}) = \frac{1}{2\pi\sigma^2} e^{-[(x-x_1)^2+(y-y_1)^2]/2\sigma^2}.$$

Blom's scheme is used for key predistribution. However instead of one key space, there are multiple key spaces. At first ω key spaces are constructed. Each sensor carries key information from τ $(2 \leq \tau \leq \omega)$ key pools, randomly selected from the ω key spaces. If two nodes carry key information from the same key space, then they can compute a shared key. However there is not guarantee that two nodes will have some key in common. Let $X_{i,j}$ represent the key space pool or the set of key spaces used by group $G_{i,j}$. Then the union of $X_{i,j}$ (for $i = 1, 2, \ldots, t$ and $j = 1, 2, \ldots, n$) equals the entire key space pool X. The deployment knowledge can be used to assign the keys from same key spaces to sensors which are close to each other. On the other hand if the groups G_{i_1,j_1} and G_{i_2,j_2} are further away from each other, the amount of overlap between X_{i_1,j_1} and X_{i_2,j_2} becomes smaller or even zero.

Advantages of this scheme include the number of keys are minimized and resiliency is improved. Overall connectivity is also very high. Disadvantages include complexity.

2.8.3 Yu-Guan Scheme

Yu and Guan [175, 176] studied key predistribution schemes using deployment knowledge and compared the effect of deployment on triangular, hexagonal and square grids. They show that hexagonal grids give better connectivity and resiliency than triangular and square grids. They make use of Blom's scheme for key predistribution. Their schemes have low storage and full connectivity between nodes within the communication range. If the number of nodes compromised is less than some threshold value, then the communication between any other nodes is secure.

According to their scheme, the entire deployment area S_f is broken down into t grids equally, where shapes of grids may be various. N nodes are also divided equally into t groups; group i is deployed in grid i. The center of the grid is a deployment point, which is the desired location of a group of nodes. It is assumed that the location of the nodes of each group i follows some probability distribution function (pdf) $f_i(x, y) = f(x, y, \mu_{x_i}, \mu_{y_i})$, where $(\mu_{x_i}, \mu_{y_i}) \in S_f$ is the coordinate of the deployment point of the group. Either a uniform distribution ($f_i(x, y) = 1/l^2$, where l is the distance between two neighboring deployment points and $x \in [\mu_{x_i} - l/2, \mu_{x_i} + l/2]$ and $y \in [\mu_{y_i} - l/2, \mu_{y_i} + l/2]$) or a Gaussian distribution ($f_i(x, y) = \frac{1}{2\pi\sigma^2} e^{-[(x-\mu_{x_i})^2+(y-\mu_{y_i})^2]/2\sigma^2}$, where σ^2 is the variance of the distribution) can be assumed.

The detailed key predistribution scheme is as follows [175].

1. A public matrix G is generated for all groups and a secret matrix A_i for each group i ($i = 1, 2, \ldots, t$). Each node U_j ($j = 1, 2, \ldots, n$) of group i picks the jth row from A_i , where j is called node ID.

2. Suppose there are in total $t_1 \times t_2$ grids, where $t_1 \times t_2 = t$. Each group i can be located by a pair of row and column index (r_i, c_i), where $r_i = 1, 2, \ldots, t_1$ and $c_i = 1, 2, \ldots, t_2$.

3. For each group i, if $r_i \mod 2 = 0$ and $c_i \mod 2 = 0$, but $r_i \mod 4 = 0$, or, $r_i \mod 4 = 0$ and $c_i \mod 2 = 1$, it is selected as the basic group and assigned a distinct matrix F. This step is repeated for all groups until all the basic groups are found.

4. For each non-basic group, all F matrices are looked up and assigned their neighboring basic groups and F matrices are assigned to the non-basic group.

5. Each node U_i of a group picks the i-th row from every matrix F assigned to its group.

6. An identical transmission range t_r is set for all nodes and deployed into the sensor field.

Assuming that the maximum number of rows that can be stored in one node is w and the memory size is M, the threshold value ν would be derived as $\nu = \lfloor \frac{M}{w} \rfloor - 1$.

The next phase shared key discovery is executed as follows.

1. Each node broadcasts its group ID, F IDs and column of G in plain text and receives these from its neighbors within the transmission range t_r.

2. Each node checks the IDs of every neighbor to see if any ID equals to one of its own. If two nodes have the same group IDs, then either of them derives a pairwise key by computing the dot product of its row of A and the column of the others.

3. If two nodes only have one common F ID, then either of them derives a pairwise key by computing the dot product of its row of that shared F and the column of the others.

4. If two nodes have more than one common F IDs, then they randomly select a shared F and derive a pairwise key from the selected F.

5. If no ID equals, two nodes stop the direct communications with each other.

The assignment of Fs is done keeping in view two metrics, resiliency against node capture and memory requirement. Given one row of some F compromised, all links that may be compromised by adversaries exists between groups sharing F. These groups are called affected groups and their number is used roughly as a measure of resiliency. Given some nodes of a group compromised, the set of affected groups is the union of all affected groups by each F assigned to this group. Since the number of Fs assigned to a group is that of its neighboring basic groups, to measure resilience against node capture of different assignments of F s one only needs to consider about the number of neighboring basic groups of each group. Let a cluster of groups denote a central group and all its neighboring groups. When nodes are deployed in hexagonal (or square) grids, each cluster consists of 7 (or 9) groups. Now one needs to consider how to assign basic groups in each cluster and determine the optimal number of basic groups needed in each cluster. It has been shown in [175, 176] that hexagonal layout has less number of affected groups and rows stored in nodes. Thus dividing sensor field into hexagonal grids is better in security and memory requirements.

The connectivity of this scheme is 1. When one node is compromised, the fraction of all links compromised in hexagonal grids is given by

$$\frac{(\pi t_r^2 q - 1) + 26 L_{comm}}{\frac{N}{2}(\pi t_r^2 q - 1)},$$

where each node has $\pi t_r^2 q - 1$ neighbors, q is the node density over the field and

$$L = \frac{q}{t_r} \int_0^{t_r} (r^2 cos^{-1} \frac{x}{t_r} - x\sqrt{t_r^2 - x^2}) dx \text{ and}$$
$$L_{comm} = \frac{4t_r}{3\sqrt{3}l} L$$

2.8.4 Huang et al's scheme

Huang, Mehta, Medhi and Harn [69, 70] discuss grid-group based key predistribution schemes which are perfectly secure to random node capture as well as perfectly secure to selective node capture. According to the scheme the deployment region is broken into smaller regions or zones. Groups of sensors are placed in these zones. This scheme is similar to Du et al's [49] scheme.

The target area is a rectangular region with size $(i.a) \times (j.a)$, divided into $i.j$ zones of equal size, such that $Area(\mathcal{Z}(i,j)) = a^2$. The total number of sensor nodes is N divided into $i.j$ groups of n_z sensors each. The group of sensors deployed in zone $\mathcal{Z}_{i,j}$ is defined as $G_{i,j}$. Each node in group $G_{i,j}$ is assigned an identifier $[(i,j),b]$, where $b = 1, 2, \ldots, N$. Each sensor is assigned k keys. The sensors in group $G_{i,j}$ are deployed uniformly in zone $\mathcal{Z}(i,j)$.

The keys in the sensors are deployed following multiple-space Blom scheme similar to Du et al's scheme [49]. Each sensor node chooses keys from two key spaces such that no more than c sensors are chosen from the same key space. Thus the number of sensors in each group is $|G_{i,j}| = c\omega/\tau$.

The key pool X is composed by $X = L \times M$ sub-key pools (a sub-key pool is represented as $P(i,j)$ where $i = 1, 2, \ldots, L$, $j = 1, 2, \ldots, M$). Each sub-key pool is divided into ω sub-key spaces. A sub-key space is a $N \times (c+1)$ key matrix A. Each element of A is a unique key. The N sensors are divided in $L \times M$ groups. For sensor $[(i,j),b]$, randomly select τ sub-key spaces from ω sub-key spaces in $P(i,j)$ while making sure that the selected sub-key space is not already selected c times. The sensor is then loaded with the bth row of matrix A for each sub-key space selected.

For distributing keys in the neighboring groups each sensor say U_i in group G_{i_1,j_1} randomly selects one sensor say j in group G_{i_2,j_2}. Groups G_{i_1,j_1} and G_{i_2,j_2} are neighbors if $|i_1 - i_2| \leq 1$ or $|j_1 - j_2| \leq 1$.

The duple $< k_{ij}, id_j >$ in installed in U_i and duple $< k_{ij}, id_i >$ in U_j, where key k_{ij} is unique and id_i , id_j are the identifiers of node U_i, U_j respectively. Once node U_i selects a peer node U_j in group $G(i_2, j_2)$, it cannot select another node in the same group. This process goes on until all sensors have chosen a node in each of its neighboring groups.

The probability $p(i^*, j^*)$ that sensor $[(i,j),b]$ can connect to the neighboring zone with the help of its neighbors is given by

$$p(i^*,j^*) = 1 - \frac{\binom{n_z - N_b(i,j)}{N_b(i^*,j^*)}\binom{n_z - N_b(i^*,j^*)}{N_b(i,j)}}{\binom{n_z}{N_b(i^*,j^*)}\binom{n_z}{N_b(i,j)}},$$

where $N_b(i,j)$ is the number of neighbors of sensor $[(i,j),b]$ within zone $Z(i,j)$. $N_b(i^*,j^*)$ is the number of neighbors of sensor $[(i,j),b]$ within zone $Z(i,j)$ or in adjacent zones and n_z is the number of sensors in the zone.

While key establishment within the same zone, each sensor, say $[(i,j),b]$, broadcasts its identifier $[(i,j),b]$ and its key space identifiers $[\tau_1, \tau_2]$. Based on the received ids and corresponding key spaces, a sensor builds a key graph with ids of

all the neighbors as vertices. For each of the neighbors, say $[(i,j),u]$, the sensor checks if they share the same key space. If they do share a key space, they can derive the pairwise key K_{bu} using the key agreement given in Du et al's scheme [49]. The node $[(i,j),b]$ will then add a link between itself and node $[(i,j),u]$ in its key graph.

After receiving the identifiers from all neighbors and adding links in the key graph for the neighbors with shared key space, the sensor broadcasts a list of neighbors who share key space with it. After receiving the same type of list from the neighbors, each sensor updates its key graph by adding edges between vertices according to the received neighbor list. Finally, based on the derived key graph, the sensor can use source routing, by explicitly specifying the key path (in hop-by-hop fashion), to send request and establish pairwise keys with all its remaining neighbors.

When a sensor wants to set up keys with its neighbors in the adjacent zones, it broadcasts the desired node list. A neighbor of the requester within the same zone who already shares a key with the nodes in the requesters list acts as a proxy and does the following: 1. selects a pairwise key for the pair, 2. encrypts the selected pairwise key using the pairwise key already set up between itself and the requester and the pairwise key already shared between itself and the destination node, 3. sends the two encrypted messages to the requester. Upon receiving the response, the requester forwards the encrypted pairwise key to the destination. Since during the first phase, nodes have already set up pairwise keys to all their neighbors within the same zone, during the second phase, as long as there exists one node with a link to the neighboring zone, it can be used as a bridge to set up pairwise keys to the neighboring zone for all its neighbors. The probability that a node can connect to neighboring zones with the help of exactly m neighbors is given as follows:

$$p_m(i^*,j^*) = \frac{\binom{n_z}{m}\binom{n_z-N_b(i,j)}{N_b(i^*,j^*)-m}\binom{n_z-N_b(i^*,j^*)}{N_b(i,j)-m}}{\binom{n_z}{N_b(i^*,j^*)}\binom{n_z}{N_b(i,j)}}.$$

The probability that a node can connect to neighboring zones with the help of at least q neighbors is denoted by $p_{\hat{q}}(i^*,j^*)$ and is given as follows:

$$p_{\hat{q}}(i^*,j^*) = 1 - [p_0(i^*,j^*) + \ldots, p_q(i^*,j^*)].$$

The scheme has been shown to be perfectly secure against random node capture as well as selective node capture, as defined in Section 1.2.3. Since each sensor node is given a different identifier, the scheme is also secure against fabrication node capture.

2.8.5 Simonova-Ling-Wang Scheme

In [149], Simonova, Ling and Wang discuss two predistribution schemes one using deployment knowledge: one for a homogeneous network and the other for a

heterogeneous network. We discuss the homogeneous scheme here and discuss the heterogeneous scheme in the next section. In both the schemes the entire deployment region is broken down into grids. Sensors are deployed in groups in these grids similar to [49]. There are two types of key pool: the original key pool and the deployment key pool. Each grid has a different original key pool. All the original key pools corresponding to the different grids are disjoint. Let the original key pool belonging to the cell $h_{i,j}$ be denoted by $OKP_{i,j}$. Initially keys in the sensors are distributed from the original key pool and placed in the sensors. Then the deployment regions are grouped together into bigger cell each having m^2 cells each. For a deployment cell $h_{i,j}$ the original key pools from the m^2 cells are merged to form the deployment key pool. For the cell $h_{i,j}$ the corresponding deployment key pool is denoted by $DKP_{i,j} = \{\bigcup_{x,y} OKP_{x,y} | x = 1, 2, \ldots, (i+m), y = 1, 2, \ldots, (j+m)\}$. So the grid is augmented by $m-1$ cells on the horizontal and vertical sides before the key pools are constructed. Simonova, Ling and Wang state that any key predistribution can be used to assign keys from the key pool. However they consider the transversal designs used by Lee and Stinson [89]. For the quadratic transversal design $TD(3, k, p)$, the key pools are generated as follows. For each cell, deployment key pool acts as a finite set X (where X is the set of elements from which the blocks are constructed) and every original key pool, contributing to a cells deployment key pool acts as a group H_g. The group size l is equal to K_{orig} number of keys in the original key pool and the number of groups p is the number of the original key pools contributing to the deployment key pool that is equal to $m \times m = m^2$. Using the set of groups H (the set of the original key pools contributing to the deployment key pool), the set of blocks $\mathcal{A}$ is constructed using quadratic transversal design. The quadratic transversal design is chosen due to its enhanced properties [89]. Each block $A_j \in \mathcal{A}$ (where $\mathcal{A}$ is the set of blocks) serves as a sensor key ring. Each cell further distributes key rings to its sensors by randomly picking a block. After a block is assigned to a sensor, the block is deleted from the set. So, every sensor gets unique block. Important relationships between parameters of the scheme are:

1. For each cell deployment key pool X is the union of the original key pools of m^2 cells. This means that the size of the finite set X is $K_{depl} = |X| = m^2 K_{orig}$.

2. Each original key pool contributing to the deployment key pool serves as a group H_g, so the set of groups consists of $k = m^2$ groups.

3. Each group H_g consists of $p = K_{orig}$ elements.

In order for overlapping deployment key pools to intersect in a known set system and provide deterministic performance, the following property should hold. Whenever deployment key pools of different cells contain original key pool of the same cell $h_{i,j}$ and use it as a group H_g, the group H_g should be assigned with the same index g in all the occurrences of the original key pool of the cell $h_{i,j}$.

Let the length of a cell side be $2h$ and R be the communication radius. Let $f_{in-cell}$ be the fraction of sensors inside the communication circle of a sensor belonging to the same cell, f_{side} be the fraction of sensors belonging to the side neighbors, f_{diag} be the fraction of sensors belonging to the diagonal neighbors. Since any sensor can reach only these three types of sensors, the following equation holds:
$f_{in-cell} + f_{side} + f_{diag} = 1$.
The fraction of in-cell neighbors is given by
$f_{in-cell}(h, R) = \frac{9\pi-5}{12\pi} \times \frac{R^2}{h^2} + \frac{2}{3}\frac{(h-R)R}{h^2} \times \frac{3\pi-2}{\pi} + \frac{(h-R)^2}{h^2} - \frac{R^2(6+4-3\pi)}{12h^2\pi}$
and the fraction of diagonal neighbors is given by
$f_{diag}(h, R) = \frac{R^2}{8h^2\pi}$.
The fraction of side neighbors can then be easily calculated. The in-cell connectivity is given by
$P_{in-cell} = \frac{m^2(m^2-1)}{2(k^2_{orig}+K_{orig}+1)}$.
The diagonal and side connectivities are given by
$P_{diag} = \frac{(m^2-2m+1)(m^2-2m)}{2(k^2_{orig}+K_{orig}+1)}$ and
$P_{side} = \frac{(m^2-m)(m^2-m-1)}{2(k^2_{orig}+K_{orig}+1)}$.
The overall connectivity is given by $P_1 = P_{in-cell}f_{in-cell} + P_{diag}f_{diag} + P_{side}f_{side}$.

The resiliency against node capture is given by
$fail_{grid}(s) = \sum_{i=0}^{s} fail_{orig}(i)\binom{s}{i}\left(\frac{m^2}{N^2}\right)^i (1 - \frac{m^2}{N^2})^{s-i}$, where $fail_{orig}(s)$ is the resiliency of Lee and Stinson's scheme [86].

2.8.6 Zhou-Ni-Ravishankar scheme

Uptil now we discussed only static nodes. In [179] Zhou, Ni and Ravishankar discussed a key predistribution scheme where sensor nodes are mobile. There are static sensor which are deployed in groups. There are mobile collectors which are used to collect and aggregate sensor data and forward to the base station. A number of schemes have been discussed in literature [76, 164, 173, 177, 178] which use mobile collectors within static sensor network to facilitate data collection. The impact of compromised mobile collectors on security is studied. It has been shown that data consistency when collectors are compromised may be improved when collectors are mobile.

For key predistribution it is assumed that there are n_s sensor nodes and n_m mobile collectors. The static sensors are arranged in g groups G_i, $1 \leq i \leq g$, each of which has $\gamma = n_s/g$ sensors. Group G_u will comprise sensors s_i such that $(u-1)\gamma < i \leq u\gamma$. The pairwise key between a pair of sensor nodes is denoted by $S-S$ key and pairwise key between a mobile collector and a sensor as an $M-S$ key. All senor nodes within a group are connected to each other using pairwise keys. If sensors U_i and U_j belong to the same group they start off associated. If they are in different groups then communication takes place by agents. Groups

G_u and G_v are said to be t-associated if they share t agents for each other. Each pair of agents share a pairwise key

If there are g groups, and each sensor has enough memory to hold μ inter-group pairwise keys, each group can have up to $t = \lceil \frac{\mu\gamma}{g-1} \rceil$ agents in each of the other groups. The functions $F_i(1 \leq i \leq t)$ are used which uniformly map group pairs from $[1, g] \times [1, g]$ to $[1, n_s]$. $F_i(G_u, G_v)$ selects the ith agent for G_v in G_u, and is defined as follows $F_i(G_u, G_v) = (t(v-1)+i)(\mod \gamma) + (u-1)\gamma$. G_u comprises of the sensors s_i with $(u-1)\gamma < i \leq u\gamma$. Hence $F_1(G_u, G_v), \ldots, F_t(G_u, G_v)$ select t sensors, with indices between $(t(v-1)+1) \mod \gamma + (u-1)\gamma$ and $tv \mod \gamma + (u-1)\gamma$ as agents for G_v. For each pair of groups G_u, G_v and for $i = 1$ to t, U_x and U_y are calculated as $U_x = F_i(G_u, G_v)$ and $U_y = F_i(G_v, G_u)$. A unique pairwise key is then assigned to U_x and U_y. The important features of the scheme are that :

1. Each sensor is agent for the same number of groups. This balances load and creates no high-value targets, since no sensor holds more keys than any other.

2. Multiple agents improve resilience for establishing path keys.

3. Agents can be discovered using the functions $F_1, F_2, \ldots, F_t$, rather than by lookups.

Sensor $< G_u, U_i >$ and $< G_v, U_j >$ generate path key in the following way.

1. $< G_u, U_i >$, first computes $H(U_i, U_j, p)$ for $1 \leq p \leq t$, and selects the p that yields the biggest H value. It now uses the function F_p to pick an associated sensor pair $< G_u, U_x >$ and $< G_v, U_y >$ for path key generation. U_i then randomly generates a key K_{ij} and sends it to agent U_x , encrypted with the association key K_{ix} it shares with U_x.

$$U_i \rightarrow U_x : (K_{ij}, G_v)_{K_{ix}}$$

2. Upon receipt, U_x decrypts this message and re-encrypts it with the association key K_{xy} it shares with U_y, and sends it to U_y.

$$U_x \rightarrow U_y : (K_{ij})_{K_{xy}}$$

3. U_y decrypts this packet, re-encrypts it with the key K_{jy} it shares with U_j , and sends it to U_j.

$$U_y \rightarrow U_j : (K_{ij})_{K_{jy}}$$

4. U_j first applies H to select the same associated pair G_u, U_x and G_v, U_y that U_i selected for path key establishment. It then recovers K_{ij} using K_{jy} , its preloaded association key with U_y.

Mobile Collectors have $O(n_s)$ Memory, each sensor U_j is preloaded with a secret key K_{U_j} shared pairwise with the base station. Sensor U_j communicates securely with mobile collector U_m using key $K_{ij} = R(K_{U_j}, U_m)$, where R is a pseudo-random function (PRF) [62]. Each mobile collector U_m is preloaded with the set of keys $\{K_{ij}\}$ for all sensors U_j. Sensor U_j can compute a unique pairwise key shared with every mobile collector U_m on-demand. However, mobile collectors have enough memory to store the keys they need. While R may be easy to compute, the overhead can be high if the number of mobile collectors is high.

When mobile collectors with limited memory, associations between each mobile collector U_m and sensors from some selected g groups are created, in a pattern that ensures that U_m is t-associated with each of the g groups. The g groups can be selected using g functions analogous to the F_i functions to ensure that each group is likely to be chosen by the same number of mobile collectors. This balances loads and creates no high-value targets, since no group of sensors hold more keys than any other. Also, agents for mobile collectors can be chosen using functions F_i' analogous to the functions F_i, in whose definition we can treat U_m as a group. The function $F_i'(G_u, U_m)$ is used to select the ith agent for U_m in G_u .

Mobile collector - sensor key establishment: A mobile collector U_m and one of its non-associated neighbor $< G_u, U_i >$ generate a path key as follows. If U_m has agents in G_u , Highest Random Weight technique is used to choose an agent for path key generation. Otherwise, U_m finds an agent in an adjacent group (say G_v), and uses that agent and the agent pair between G_u and G_v as intermediaries to establish path keys. To further reduce the communication overheads at sensors, U_m may be allowed to move to the agent. The advantage of this scheme is that it improves data consistency and has good connectivity and communication overhead.

Other schemes where Deployment knowledge has been employed by mobile sensors have been studied in [168, 183].

2.9 Heterogeneous Networks

We have discussed key predistribution schemes in which all the sensor nodes have equal capacities in terms of memory and battery power. Such nodes are capable of operating within the same RF region, have equal number of keys in the sensor networks and behave in the same way for broadcasting information and to node compromise. In this section we discuss heterogeneous networks, networks in which nodes vary in memory and battery capacities. Some nodes are tamper proof and other are not. They also differ in the mode of operation and are placed in different environments.

In a hierarchical networks there are three types of nodes: *sensor nodes*, *cluster heads* (CH) and *base station* in increasing order of battery and memory capacities. The base station is very powerful and is placed in a safe place and is thus

assumed to be fully resilient to adversarial attacks. The cluster heads and sensors nodes are deployed in the adversarial region. The cluster heads are responsible for aggregating the information from the sensor nodes and forwarding to the base station. Thus the cluster heads are more powerful than the sensor nodes and fewer in number compared to the sensor nodes.

2.9.1 Perrig et al's SPINS protocol

Perrig, Szeweczk, Wen, Culler and Tygar [121] presented a suite of security protocol optimized for sensor networks called SPINS. It has two components : SNEP and μTESLA. SNEP provides data confidentiality and two-party data authentication. μTESLA provides efficient broadcast for severely resource-constrained environments. μTESLA is a an improvement of $TESLA$ (Time Efficient Stream Loss-Tolerant Authentication) proposed by Perrig, Canetti, Tygar and Song [120]. μTESLA was improved to suit the requirements of resource constrained sensor nodes. In SPINS the base station acts as key distribution center. μTESLA requires that base station and sensor nodes be loosely time synchronized. Basically, 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, K_1, ...,K_{n-1} such that $K_i = H(K_i + 1)$. Given K_i , every sensor node can generate the sequence K_0, K_1, ..., $K_i - 1$. However, given K_i, K_{i+1} cannot be generated. At the 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. Sensor nodes can verify 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 sensor nodes to bootstrap from the Base Station; that is, they receive the first key of the chain which is called key chain commitment. Bootstrapping procedure requires unicast communication, and can be secured with pair-wise keys. Also, TESLA is used in [31,41,42] to authenticate message broadcasts from base station, in [150] to authenticate route update broadcasts, and in LEAP [181] to update pre-deployed network-wise keys in case of a node compromise. In this scheme, the Central Authority generates certificate $Cert(ID_A, t_{i+d}, \ldots, MACK_i(\ldots))$ 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. Advantages of this scheme include that it is one of the best memory efficient schemes discussed, provides strong security features with low complexity, universal design allows use in many low-end devices, incurs low communication cost, and offers authentication and strong data freshness with a minimum overhead. Disadvantages of this scheme include μTESLA overhead from releasing keys after a certain delay, possible message delay.

2.9.2 Zhu, Setia and Jajodia's LEAP protocol

A key distribution scheme was proposed by Zhu, Setia and Jajodia [181]. This scheme is called LEAP (Localized Encryption and Authentication Protocol). LEAP is a protocol, which was designed to support in-network processing, while at the same time restricting the security impact of a node compromise to immediate neighborhood of the compromised node.

The design of the protocol is motivated by the observation that different types of messages exchanged between sensor nodes have different security requirement and that a single keying mechanism is not suitable for meeting these different security requirements. LEAP supports the establishment of four types of keys for each sensor node: an individual key shared with the base station, a pairwise key shared with another sensor node, a cluster key shared with multiple neighboring nodes, and a group key that is shared by all the nodes in the network. LEAP also includes an efficient protocol for inter-node traffic authentication based on the use of one-way key chains. An important feature of the authentication protocol is that it supports source authentication without precluding in-network processing and passive participation. Research has shown that in-network processing techniques are very important for saving energy consumption in sensor networks [73, 77, 99]. There four types of keys are:

1. Individual key: This is a unique key that is shared between the base station and each sensor node. Sensor nodes use this key to calculate the MACs on their messages to the base station like alert signals (reports on abnormal nodes). In the same way, a base station can use an individual key to send messages to each and every node in the network.

2. Pairwise shared key: This is a unique key that is shared between each node and its neighboring node in the network. A node can use it to transfer individual messages like sharing a cluster key or sending data to an aggregator node.

3. Cluster key: This is a key that is shared between a node and its neighboring nodes, and is very important since it supports In-network Processing and Passive Participation. A node may elect not to send a message to the base station if its neighboring node is sending the same message with a better signal, a discovery that is only possible to implement if a node shares a common key with its neighboring nodes. With such a cluster key, a node can select which messages to transfer, thereby reducing the system communication overhead.

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

Local Broadcast Authentication: Local broadcast is different from global broadcast in that in local broadcast a node generally does not know what packet it is going to generate next and messages generally consist of aggregated sensor readings or routing protocols. Fortunately, Zhu, Setia, and Jajodia have designed a scheme called *one-way key chain-based authentication* for inter-node traffic authentication. This scheme is based on μTESLA in that each node generates a one-way key chain and sends the commitment of it to their neighbors. This transferring is done using the pairwise keys already shared with neighbors.

Resilience to Attack: An adversary might launch a selective forwarding attack in which a compromised node drops the packets containing the routing information of selected nodes and forwards the other packets normally. LEAP can minimize the effects of the scheme by minimizing this problem to a local area. As LEAP uses local broadcast, the attack's effects will not transfer to more than 2-hops, which will result in defeating the purpose of such an attack. LEAP can also prevent a HELLO attack in which an adversary attacks the network by repeatedly transmitting HELLO messages and thereby depletes the networks resources. This attack is averted since the nodes in a LEAP scheme accept packets only from authenticated neighbors. The sinkhole and wormhole attacks, however, are difficult to solve. In the sinkhole attack, a compromised node attracts packets by advertising information like high battery power, etc., then later drops all the packets. In the wormhole attack an adversary launches two nodes in the network, one near the target of interest and the other near the base station. The adversary then convinces the nodes near the target, which would generally be multiple hops away from the base station, that they are only two hops away thereby creating a sinkhole. Also, nodes that are far away think that they are neighbors because of the wormhole created. In LEAP an adversary cannot launch a wormhole attack after key establishment as at that point every node has knowledge about its neighbors so it is not easy to convince a node that it is near a particular compromised node. An insider node must then succeed in compromising two nodes for creating a wormhole and those nodes must be near the target of interest and the base station after the key establishment phase is complete. Although an adversary may try, it is difficult to create an attractive sinkhole without being detected

Advantages of this scheme include that it offers efficient protocols for supporting four types of key schemes for different types of messages broadcasted, reduces battery usage and communication overhead through in-network processing, and uses a variant of μTESLA to provide local broadcast authentication. Disadvantages of this scheme include that it requires excessive storage with each node storing four types of keys and a one-way key chain, computation and communication overhead dependant upon network density (the more dense a network, the more overhead it has).

2.9.3 Jolly et al's scheme

A predistribution scheme was proposed in this hierarchical framework by Jolly, Kusçu, Kokate and Younis [74]. According to their model there is a *command node* (base station) which is in charge of the network's mission. There are cluster heads which they call *gateways* and sensor nodes. The deployed nodes apart from the command node are divided into clusters - each cluster consists of a gateway and a set of sensor nodes (distinct from other sets). The clusters may be formed as in [64]. The sensors gather information and transmit to the gateways which fuses the data and performs other related task and sends it to the command node via a long-haul transmission. The clusters are randomly scattered in the adversarial region. Each sensor stores only two keys - one key is shared with a gateway and the other is shared with the command node. The gateways stores a key with each of the sensor nodes in its cluster, exactly one key with one other gateway and also a group key shared with all the gateways. The command node is assumed to be secure and is assigned $|G| + |S|$, where $|G|$ and $|S|$ are the number of gateways and sensor nodes respectively. In the initialization phase each gateway is assigned $|S|/|G|$ keys. Each gateway forms a cluster using a cluster forming algorithm, and therefore acquires the keys of the sensors in its cluster from the other gateways.

The scheme has the advantage that only two keys have to be kept in each sensor node. However the compromise of any gateway compromises the entire cluster, since all nodes in a cluster share keys with the gateway.

2.9.4 Cheng and Agrawal's modification of Jolly et al's scheme

Jolly et al's scheme was modified by Cheng and Agrawal [32]. They introduced the improved key distribution mechanism (IKDM). Like the previous scheme [74], sensors communicate with each others via cluster heads (termed as gateways in the previous scheme). Cluster heads can communicate with one another directly and relay data between its cluster members and the base station (command node in the previous scheme). Like the previous scheme, the base station is assumed to be powerful and secure to attacks. The base station may be positioned either in the center or at one corner of the network. Suppose there are N sensor nodes and m cluster heads. Thus each cluster contains $\lceil N/m \rceil$sensor inside. Now Blundo's scheme [10] is applied in the following way. Two bivariate polynomials are used, on $f_{CH}(x, y)$ which is used to establish pairwise keys with the cluster heads and the other $f_{CH_i}(x, y)$ $(1 \leq i \leq m)$ which is used by the cluster head in the ith cluster to calculate a secret share for an intended sensor node. Thus each cluster head contains two polynomial shares in its memory, one of which it shares with the other cluster heads and one of which it shares with the sensor nodes in its cluster. Each cluster head also shares a pairwise key with the base station.

Each sensor node contains two keys, one of which it shares with the base station

and the other with the intended cluster head after deployment.

1. Key Distribution Server (KDS) randomly selects l ($l \geq 1$) polynomials from the m polynomials $f_{CH_i}(x, y)$, $(1 \leq i \leq m)$. To achieve sufficient security, large l is desired. Let us assume $l = 2$ in this example and polynomials $f_{CH_a}(x, y)$ and $f_{CH_b}(x, y)$ are randomly selected.

2. KDS evaluates $f_{CH_a}(x, y)$ at $(x = CH_a, y = U_i)$ and $f_{CH_b}(x, y)$ at $(x = CH_b, y = U_i)$ respectively to get the two secret shares k_1 and k_2 of K_{U_i-CH}. $k_1 = f_{CH_a}(CH_a, U_i)$, $k_2 = f_{CH_b}(CH_b, U_i)$.

3. KDS calculates key K_{U_i-CH} by
$K_{U_i-CH} = k_1 \oplus k_2$.

4. KDS preloads key K_{U_i-CH} with the two cluster head id CH_a and CH_b into sensor node U_i. K_{U_i-CH} will be the pairwise key between node U_i and its intended cluster head after the deployment.

Thus the sensor node U_i stores only two key, one which it shares with the base station (K_{U_i-BS}) and the other is the pairwise key shared with the cluster head (K_{U_i-CH}) and is calculated as above. Any two sensors within the same cluster communicate via the cluster head.

The intercluster pairwise key establishment is done as in all schemes using Blundo's scheme [10] by evaluating the polynomials at the derived point.

The scheme has the advantage that each sensor contains only two keys and since Blundo's scheme is used, at least t cluster heads must be compromised to compromise the entire network.

2.9.5 Paterson-Stinson attacks on Cheng and Agrawal's scheme

Paterson and Stinson [118] presented two attacks on the above scheme. The first attack is called the *interpolation attack* and takes place for certain parameters. The second attack, though weaker can always take place and is called the *reconstruction attack*. Both the attacks exploit the construction of the pairwise key between sensor nodes and nearest cluster head. During the formation of the pairwise keys, the sensor nodes contains the list of l cluster heads. This list is sent in the clear to a cluster head, so it is known to the adversary. By compromising s cluster heads, the adversary gets sN/m such l-subsets(blocks). The average number of occurrences of a point $x \in \{1, ..., m\}$ in the sN/m blocks is sNl/m^2. If $\frac{sNl}{m^2} = 1.25l$, then it can be shown that almost every point will occur more than t times. Thus if an adversary compromises $s = 1.25tm^2/(Nl)$ cluster heads after the IKDM process, then the expected number of CH-polynomials (Cluster-Head polynomials) that can be reconstructed using the interpolation attack is at least $m(1.0e^{.025t})$.

It has already been stated in [32] that if s cluster heads are compromised, then sN/m sensors are exposed. However Paterson and Stinson [118] shows that actually $\frac{N}{m}\left(s+\frac{(m-s)\binom{s}{l}}{\binom{m}{l}}\right)$ sensors are exposed. Suppose $\mathcal{J}=\{j_1,j_2,\ldots,j_s\}$ cluster heads are compromised, then a sensor node which has a list B_i such that $B_i \subseteq \mathcal{J}$ is exposed, since all the keys k_i are compromised. The probability that $B_i \subseteq \mathcal{J}$ is $\frac{\binom{s}{l}}{\binom{m}{l}}$. There are $N - sN/m = N(m-s)/m$ sensor nodes whose nearest cluster head has not been compromised. Therefore, the expected number of reconstructed sensor node keys is $\frac{N(m-s)\binom{s}{l}}{m\binom{m}{l}}$.

2.9.6 Das-Sengupta scheme

The two schemes [32, 74] suffer from the disadvantage that sensor nodes cannot communicate with each other directly and can only communicate via the cluster head. As such if the cluster heads become compromised, then all the sensor nodes in the cluster are compromised. To alleviate these problems Das and Sengupta proposed a scheme [40] which also uses bivariate polynomials similar to Cheng and Agrawal's scheme [32]. The target field is divided into m equal sized disjoint groups (clusters). Each group consists of a cluster head. Each sensor node is given a share of a t-degree bivariate polynomial similar to Blundo's scheme [10]. The number of sensors in each cluster is less than t so that even if all the sensors are compromised, the bivariate polynomial will not be disclosed to the enemy. The base station may be placed either at the center or at one corner of the deployment region.

Key predistribution is carried out as follows.

1. The set-up server assigns each cluster head and each sensor node an unique identifier.

2. The setup server generates randomly a unique master key MK_{GH} to each group head GH_i which is shared with the base station only. Similarly each sensor node shares a key with the base station.

3. For all the m deployed group heads GH_i $(1 \leq i \leq m)$, the setup server randomly generates a t-degree bivariate polynomial $f(x,y) \in F_q[x,y]$ over a finite field F_q such that $f(x,y)$ is symmetric i.e., $f(x,y) = f(y,x)$ and such that $t >> m$. This ensures that even if all the cluster heads are compromised, the polynomials will not be known to the adversary. The setup server then computes a polynomial share $f(GH_i,y)$ for each deployed cluster head GH_i and loads into its memory.

4. For each deployment group $G_i(i = 1,2,...,m)$, the setup server randomly generates a unique t degree bivariate polynomial $f_{G_i}(x,y) \in F_q[x,y]$ over a

finite field F_q , such that $f_{G_i}(x, y)$ is symmetric. For each regular sensor node U_j in a deployment group G_i , the setup server computes a polynomial share $f_{G_i}(U_j, y)$ and loads this into its memory.

5. Then the setup server loads the following information into the memory of each group head $GH_i(i = 1, 2, ..., m)$: (i) its own identifier, (ii) its own master key MK_{GH_i} , (iii) the polynomial share $f(GH_i, y)$ and (iv) also the polynomial share $f_{G_i}(GH_i, y)$. Each deployed regular sensor node U_j in a deployment group G_i is loaded with the following information: (i) its own identifier, (ii) its own master key MK_{U_j} and (iii) the polynomial share $f_{G_i}(U_j, y)$.

For intergroup communication any two cluster heads GH_i and GH_j can calculate their pairwise key k_{GH_i,GH_j} by evaluating the polynomial $f(GH_i, GH_j) = f(GH_j, GH_i)$. For intragroup communication any two sensor nodes U_u and U_v in cluster $G_i$can calculate their pairwise key k_{U_u,U_v} by evaluating the polynomial $f_{G_i}(U_u, U_v) = f_{G_i}(U_v, U_u)$. In a similar way a senor node can calculate its pairwise key with cluster head GH_i by evaluating evaluating $f_{G_i}(U_u, G_i) = f_{G_i}(G_i, U_u)$.

Nodes may not be placed in their own cluster at the time of deployment. Suppose a node U_u belonging to cluster G_i is deployed in another cluster G_j. Let U_u want to communicate with node U_v which belongs to cluster G_j. Then U_u sends a request to U_v. This request consists of its own id and a randomly generated nonce RN_u . After receiving such a request, node U_v also generates a random nonce RN_v and sends a request consisting of its own id as well as the id of U_u, the random nonce RN_u and RN_v to its own group head GH_j protected by its own master key MK_{U_v} . The group head GH_j forwards this request to its neighbor group head and finally to the base station. The base station first validates this request by decrypting the request by the master key MK_{U_v} of the node U_v. If the validation passes, the base station generates a random key k_{U_u,U_v} to be shared by the nodes U_u and U_v, makes two protected copies $E_{MK_{U_u}}(k_{U_u,U_v} \oplus u \oplus RN_u)$ and $E_{MK_v}(k_{U_u,U_v} \oplus v \oplus RN_v)$ where $E_k(\mathcal{M})$ denotes the encryption of the data $\mathcal{M}$ using the key k, and the first one is sent to node U_u and the later copy is sent to node U_v via group heads. Nodes U_u and U_v first decrypt their protected copies. Node U_u retrieves the secret key k_{U_u,U_v} using its own id and its own random nonce RN_u as $k_{U_u,U_v} = (k_{U_u,U_v} \oplus u \oplus RN_u) \oplus (u \oplus RN_u)$. Similarly, node U_v also uses its own id and random nonce RN_v in order to retrieve the secret key $k_{u,v}$ as $k_{u,v} = (k_{u,v} \oplus v \oplus RN_v) \oplus (v \oplus RN_v)$.

The transaction is uniquely determined by attaching the random nonces of the nodes in the message by the base station. It can be noted that the communication overhead is not much due to involvement of the group heads during this process. In fact, such a scenario is unlikely to occur, because the probability of having a smaller deployment error is typically higher than the probability of having a larger one when the nodes are randomly deployed in a deployment group. In a similar fashion, node U_u can also establish a secret key with the group head GH_j if GH_j is neighbor of U_u.

The advantage of this scheme is that even if all the cluster heads are compromised, the polynomial is not disclosed. Within a cluster, even if all the sensors are compromised, the polynomials are not compromised. This helps to add more sensors in the network. In fact we can add sensors till the number of sensors in each cluster is t. Thus the system is scalable.

The disadvantage of the scheme is that the cluster size is small compared to the storage required. For example if $t = 198$, then 200 keys have to be placed in the sensors. And the number of sensors can be at most 200.

2.9.7 The SecLEACH protocol

Another cluster based approach in heterogeneous networks was taken by Oliveira, Ferreira, Vilaça, Wong, Bern, Dahab and Loureiro [116,117]. The scheme is known as SecLEACH , which is a modification of the LEACH [67] (proposed by Heinzelman,Chandrakasan,Balakrishnan) and F-LEACH [56](proposed Ferreira, Vilacca, Oliveira, Habib, Wong and Loureiro). LEACH (Low Energy Adaptive Clustering Hierarchy) works in rounds where cluster heads are rotated among all nodes in the network from time to time, thus distributing aggregation and routing-related energy consumption among all node in the network. In each round it uses a distributed algorithm to elect Cluster Heads (CHs) and dynamically cluster the remaining nodes around the CHs. The sensor nodes send their messages to the CHs which aggregate the information and forwards to the base station.

Rounds in LEACH consists of two steps *setup* phase and *steady-state* phase. There are synchronized, to mark the beginning of a round. The setup consists of three steps. The first step is called *advertisement* step and here nodes decide whether to become a CH for that round. Those which decide to do so, broadcast a message *adv* advertising this fact at a level where it can be heard by everyone in the network. To avoid collision, a carrier sense multiple access control is used. In the second step (called the *cluster joining step*, the remaining nodes pick up a cluster based on the largest signal to noise ratio and communicate their intension to join the CH by sending a $join\_req$ message. In the last step (the *confirmation* step) the CHs broadcast a confirmation message that is used for communication during the steady state phase.

After the *setup* phase the system moves to the *steady-state* phase in which actual communication between sensor nodes and base station takes place. The CHs collect messages from the sensor nodes, aggregate these data and send the data to the base station. The steady-state phase consists of multiple reporting cycles and lasts much longer than the setup phase.

Like most routing protocols for WSNs, LEACH is vulnerable to a number of security attacks [78], including jamming, spoofing, replay, etc. However, because it is a cluster- based protocol, relying fundamentally on the CHs for data aggregation and routing, attacks involving CHs are the most damaging. If an intruder manages to become a CH, it can stage attacks such as sinkhole and selective forwarding,

thus disrupting the workings of the network. Of course, the intruder may leave the routing alone, and try to inject bogus sensor data into the network, one way or another. A third type of attack is (passive) eavesdropping

Given the communication patterns in LEACH, two different types of authentication are required: authenticated broadcast, for broadcasts from the CHs to the rest of the network; and pairwise authentication for the remaining (node-to-CH and CH-to-BS) communications.

Symmetric-key authenticated broadcasts for WSNs, both global (μTESLA [121]) and local (LEAP [181]), share the core idea of using a one-way key chain (a sequence of keys $k_1, k_2 \ldots, k_n$, where k_{i+1} is generated from k_i by applying a one-way hash function $f()$, i.e., $k_{i+1} = f(k_i)$) to achieve authentication. These schemes cannot be applied, as is, to LEACH because: 1) the key chain would require significant storage space in the broadcasting CHs; and more importantly, 2) all nodes in the network would need to store one key for each node in the network, which is neither practical nor scalable. (Each node needs to store one key for every other node in the network because an ordinary node needs to be able to authenticate the CHs in each round, which can be arbitrary nodes in the network.)

Pairwise authentication is also challenging to implement in LEACH, because of key distribution issues. Given that any node needs to be ready to join any CH (which could be any node in the network), it would need to have shared pairwise keys with every other node in the network. Just like in authenticated broadcast, this is neither practical, nor scalable.

Ferreira et al. [56] proposed a scheme (henceforth refereed as F-LEACH) where each node has two symmetric keys: a pairwise key shared with the BS; and the last key of a key chain held by the BS, used in authenticated broadcast.

F-LEACH implements authentication for CH's broadcasts in two smaller steps, leveraging on the BS, who is trusted and has more resources. Briefly, each CH sends a slightly modified *adv* message consisting of: the id of the CH in plaintext, used by the ordinary nodes as before and a MAC produced using the key the CH shares with the BS (which will be used by the BS for the purpose of authentication). The BS waits to hear and authenticate (modified) *adv* messages from all CHs; compiles the list of legitimate CHs; and sends the list to the network using the μTESLA [121](SPINS protocol) broadcast authentication scheme. Ordinary nodes now know which of the *adv* messages they received are from legitimate nodes, and can proceed with the rest of the original protocol, choosing the CH from the list broadcasted by the BS. The other broadcast by the CHs is authenticated the same way. Using only two keys per node, F-LEACH does not manage to provide a complete and efficient solution for node-to- CH authentication. In particular, *join-req* messages in the setup protocol are not authenticated; and ordinary nodes only share keys with the BS. This means that the CHs are prevented from verifying the sensing report's MACs and, in turn, have to forward them, which incurs a considerable energy consumption.

To overcome this problem SecLEACH was proposed by Oliveira et al [116,117].

SecLEACH uses random key predistribution, to set up keys for securing node-to-CH communication in LEACH. In SecLEACH, a large pool X keys and their ids are generated prior to network deployment. Each node is then assigned a ring of k keys drawn from the pool pseudorandomly [62], without replacement, as follows. For each node U_i, we use a pseudorandom function (PRF) to generate its unique id id_i. id_i is then used to seed a pseudorandom number generator (PRNG) of a large enough period to produce a sequence of k numbers. $P(U_i)$, the set of key ids assigned to U_i, can then be obtained by mapping each number in the sequence to its correspondent value modulus. Also prior to deployment, for each node is assigned a pairwise key shared with the BS. The LEACH clustering algorithm can then be run with the following modifications: when a self-elected CH broadcasts its *adv* message , it includes the ids of the keys in its key ring; the remaining nodes now cluster around the closest CH with whom they share a key.

The memory usages, energy efficiency and security is better than previously proposed schemes.

2.9.8 Du et al's Scheme

Other heterogeneous schemes include a scheme by Du, Xiao, Guizani and Chen [52] in which there are two types of sensors the H-nodes which have more battery power, communication range and storage capacity and L-nodes which are less powerful than the H-nodes. There is also a base station which is powerful and secure. The H-nodes can be thought of as cluster heads and the L-nodes as small sensor nodes. The keys in the H-nodes and L-nodes are chosen randomly from a key pool such that the L-nodes contain k keys each and the H-nodes contain M keys each such that $M >> k$. The shared key discovery and path key establishment are the same as homogeneous networks. An additional property of this network is that new nodes can added in the network.

2.9.9 Modification of Du et al's Scheme by Hussain, Kauser and Masood

A modification of Du et al's scheme [52] was proposed by Hussain, Kauser and Masood [71]. In their scheme the key pool X consists of M different key chains C_i, such that

$$X = C_0|C_1|\cdots|C_{M-1}$$

where $C_i \bigcap C_j = \emptyset$ $(\forall i \neq j)$. The n-th key of the key chain C_i is computed as

$$k_{C_{i,n}} = HASH^n(S, g_i),$$

where g_i is a unique generation key and S is a publicly known seed. The total number of keys in a key chain is $|X|/M$.

The key predistribution algorithm is as follows.

1. Each L-sensor node is assigned with k randomly selected generation keys of corresponding key chains. From these k generation keys $k \times N$ random keys can be calculated effectively.

2. Each H-sensor node is pre-loaded with S randomly selected generation keys of corresponding key chains, where $S >> k$.

The shared key discovery algorithm is the same as [52]. The connectivity of this scheme is shown to be better than Du et al's scheme [52].

2.9.10 HERO protocol

HERO [97](stands for Hierarchical kEy management pRotocol for heterOgeneous WSN) (proposed by Maala, Challal and Bouabdullah) has been proposed for hierarchical networks in which a secure tree rooted at the sink is created. Limiting the key sharing requirement to the child-parent relationship allows to reduce the key storage overhead to few keys per mote. The authors try to create a tree instead of a connected graph. There are two types of nodes The powerful supernodes Sn and the less powerful Normal nodes Nn. Each node has the information stating the type of node it is.

For key predistribution a large key pool X is created. Each super node Sn is preloaded with M keys and a normal node is preloaded with k keys such that $M >> k$. Any two normal node or super node have some keys in common.

HERO aims at constructing a secure path from the nodes to the sink. The sink node (can also be thought of as a base station) contains all the X keys of the key pool and thus shares keys with every node.

Each attached node U_i (Sn or Nn) authenticates the request through computing one Message Authentication Code (MAC), over its identifier, using each key of its key ring. For instance, the "tree construction request" forwarded by node U_i will be:
$\{id_i, MAC(K_1, id_i), MAC(K_2, id_i), \ldots, MAC(K_k, id_i), ind_i\}$, where k is the size of nodes key ring and ind_i refers to the type of node U_i. Then, node U_i broadcasts the request. When a node receives this message, it checks the type of the node. Thus, there are two different cases:

1. Receiving node is an Sn node: Suppose that U_A is the receiving node and U_B is the sending node. Then, U_A checks the type of node U_B.

   (a) If U_B is an Sn node. Then U_A checks the request message in order to discover if it has a shared key with U_B or not. This is done by verifying the corresponding MAC. Here, if U_A finds a shared key with U_B, it notes node U_B as its parent and notes their shared key in its table of routing. Otherwise, U_A does nothing and it waits receiving a request from another node.

(b) If U_B is an Nn node. Then U_A waits wishing receiving a request from an Sn node.

2. Receiving node is an Nn node: If U_A finds a shared key with U_B, it notes U_B as its parent in its routing table and their shared key. Otherwise, it waits receiving another request.

To attach a new node U_A to the secure tree, the new node must search for a parent, already attached to the secure tree, with whom it shares a key. Thus, U_A is firstly preloaded with its key ring of a keys which are selected randomly from the key pool. Then, it broadcasts a request message as follows: $\{Id_A, MAC(K_1, id_A), MAC(K_2, id_A), \ldots, MAC(K_k, id_A), ind_A\}$. When a deployed sensor node U_B, which is already an attached node, receives this message, it broadcasts the following message:
$\{Id_B, MAC(K_1, id_B), MAC(K_2, id_B), \ldots, MAC(K_k, id_B), ind_B\}$. When U_A receives this message, it checks if it has a shared key with U_B. If U_A and U_B have a shared key, U_A notes U_B as its parent. Otherwise U_A waits receiving a message from another node. Furthermore, due to one of the reasons which are cited above, a deployed node U_C may be still unattached to the secure tree. So, upon receiving the request of the new node U_A, U_C attempts to profit from this new node to attach to the secure tree. Thus, it checks if there is a shared key with U_A, if a shared key is found, U_C considers U_A as its parent.

Probability of key sharing is better than Du et al's scheme [52].

2.9.11 SHELL protocol of Younis, Ghumman and Eltoweissy

Another hierarchical location aware key predistribution scheme was proposed by Younis, Ghumman and Eltoweissy [174]. This scheme is called SHELL because it is Scalable, Hierarchical, Efficient, Location-aware and Light-weight. The underlying key predistribution scheme is Exclusion Basis System (EBS), developed by Eltoweissy, Heydari, Morales and Sadborough [53]. SHELL has three types of nodes, the sensor nodes, which sense and collect information from the deployment field, the gateways which aggregate the information, process it and sends to the command node. The command node is extremely powerful. Sensors are deployed in clusters. These clusters are formed based on various criteria such as capability, location and communication range [64]. The sensor nodes and gateways are assumed to be static.

The EBS is a combinatorial optimization methodology for key management of group communication setups. It exploits the trade-off between the number of administrative keys, k and the number of rekeying messages m. A set of $k + m$ administrative keys is used to support a set of N nodes, where each node is assigned a distinct combination of k keys. A node can be simply admitted to the group by assigning one of the unused set of k keys out of the total of $\binom{k+m}{k}$, distinct

combinations. Eviction of a compromised node can be performed by broadcasting replacement of the k keys that the evicted node knows using the m keys that the node does not know. The EBS approach proves to be very scalable for large networks and enables great flexibility in network management by controlling the values of k and m. Large k increases the storage requirements at the node, while large m increases communication overhead for key management. An EBS is defined as a collection Γ of subsets of the set of members. Each subset corresponds to a key and the elements of a subset $A \in \Gamma$ are the nodes that have that key. An EBS Γ of dimension (N, k, m); represents a situation in a secure group where there are N members numbered 1 through N, and where a key server holds a distinct key for each subset in Γ. If the subset A_i is in Γ, then each of the members whose number appears in the subset A_i know the distinct key (provided by the key server) for that subset. Furthermore, for each $t \in [1, N]$, there are m elements in Γ whose union is $[1, N] - \{t\}$. From this, it follows that the key server can evict any member t, rekey, and let all remaining members know the replacement keys for the k keys they are entitled to know, by multicasting m messages encrypted by the keys corresponding to the m elements in Γ whose union is $[1, N] - \{t\}$. Each new key is encrypted by its predecessor in order to limit decipher ability only to the appropriate members.

Each sensor has a preloaded discovery key K_{sg}, as well as a one-way hash function to recompute K_{sg}. Each node also has two preloaded keys KS_{CH} and KS_{Key} for initial key distribution. In addition, every node should have the capacity for storing K administrative keys and ck communication keys. The keys preloaded in each gateway are $K_{gc,i}$ which is used for direct communication between the gateway and the command node, the keys K_{g_i,g_j} - the inter gateway key for communication between gateway i and gateway j. The inter gateway keys are supplied by the command node during the bootstrapping phase. The sensor discovery key which is provided to each gateway by the command node. The tasks of the gateway are to form the EBS matrix and generate the communication key of its own cluster, generate administrative keys for other clusters as and when required, refresh data keys of the cluster after network setup, detect and evict compromised sensor nodes in its cluster. The command node acts as a repository for valid IDs and preloaded keys $(K_{sg}, KS_{CH}, KS_{key})$ of all sensor in the region, authenticates the gateways and sensors, distributes keys for gateway communications and for detecting the compromised nodes, perform key renewal for the inter gateway communication and trigger renewal of gateway-to-sensor communication keys in order to counter potential on-going spoofing.

When the network is set up, each gateway i establishes connection with the command node. The command nodes broadcasts inter gateway keys encrypted with $K_{gc,i}$. After this step sensor discovery starts. The sensor nodes broadcasts their location to the gateways. Once the gateways obtain this information, it tabulates the sensor's ID and location. The sensors are then distributed in clusters each managed by a gateway $G_{CH}[i]$.

Once the gateway has information about all the sensors in its cluster, it distributes keys according to the EBS. The EBS table with the gateways list of sensors is then send to the command node. Only the key identifiers are sent and not the keys themselves because of security reasons. The command node designates for each cluster some other node (apart from the gateway) that will generate the administrative keys for the cluster. Let these be denoted by $G_{K1}[i]$ and $G_{K2}[i]$. The cluster head sends the relevant portion to these gateways, such that any of the gateways doesn't know the part of the EBS assigned to another gateway.

Upon generation of the keys, each sensor node is informed about the set of administrative keys that it was assigned.

New sensor nodes can be added to the network at any time. When a new sensor node is added the command node lets the gateway node to broaden its power range and gives it the key to communicate with the new node. When compromised sensor are detected, the sensors are revoked and keys are redistributed in sensor nodes. To ensure that two compromised sensors do not have a large number of common keys a heuristic method is used. This method uses the Hamming distance between the two strings which corresponds to the two columns in the EBS matrix given to two sensors. For more details on the algorithm one may refer to [174][Section 4.2]. When the gateway id is compromised, the command node assumes the responsibility of initializing a rekeying of the inter gateway nodes. There are two ways of of accommodating for a compromised gateway node. The first is to deploy a new gateway node. the other is to distribute the nodes by the compromised gateway node to other clusters. If the option to replace the gateway is used and the nodes in its governed cluster cannot communicate with it, a key redistribution is required.

2.9.12 Simonova, Ling and Wang's scheme using deployment knowledge

We have said previously that Simonova, Ling and Wang [149] had proposed two deployment knowledge based schemes. We have discussed one of the schemes. The second scheme is heterogeneous in which there are two types of nodes : the strong nodes and the weak nodes. The strong node contain more number of keys(mem_s keys in each sensor), compared to the weak nodes (mem_w keys in each sensor) and have higher communication range than the weak nodes. Initially mem_w keys are distributed in each of the weak nodes. Once the weak nodes are deployed, the cells are grouped together as super cells. The $mem_s - mem_w$ keys are now placed in the strong nodes. The rest of the predistribution scheme is similar to the one given in the Section 2.8.5.

2.10 Comparison of different key predistribution schemes

We compare in Table 2.1 and Table 2.2 the scalability, key connectivity, resiliency, key storage, communication overhead and computation required for key establishment. We also state the nature of the key predistribution scheme - probabilistic, deterministic and hybrid. To compute the the resiliency we consider either of the two aspects.

1. Probability that a link is broken when a node is compromised instead of $E(s)$
2. We also consider t-secure resiliency meaning that the network remains connected if t or less nodes are compromised.

For larger expressions we refer to the respective section in the papers where they appear. A resiliency of 0 means that the rest of the network is not affected when a node is compromised.

To compute the key storage, we give either the exact number of keys and when comparing the number of keys with the size of the network, we give the order in terms of the size of the network. In some cases we simply write the number of keys as k meaning that the number of keys can be chosen arbitrarily, that there is no relation with the parameters of the design.

The same holds for the communication overhead, where we either give the exact number of bits required or the order of the same.

The computation required for shared key discovery is calculated in the same way. The symbols have their usual meaning as discussed in the chapter.

Scheme	Type	Scalability	Key Connectivity	Resiliency	Key Storage	Overhead	Computation
Blom	Probabilistic	Not Scalable	1	t-secure	$t+1$	$t+1$	$t+1$
Blundo	Probabilistic	Scalable	1	t-secure	$(t+1)\log q$	$O(\log N)$	$t+1$
ES	Probabilistic	Scalable	$\frac{((\lvert X\rvert-k)!)^2}{(\lvert X\rvert-2k)!\lvert X\rvert!}$	$k/\lvert X\rvert$	k	$O(k\log\lvert X\rvert)$	$k\log\lvert k\rvert$
Q-Composite	Probabilistic	Limited Scalability	Given in [28, Section 5.2]	$\binom{k}{q}$	k	$O(k\log\lvert X\rvert)$	$k\log\lvert k\rvert$
Random Pair-wise Schemes							
Chan-Perrig-Song [28]	Probabilistic	Scalable	$\frac{((\lvert X\rvert-k)!)^2}{(\lvert X\rvert-2k)!\lvert X\rvert!}$	$k/\lvert X\rvert$	Np_c	$O(k\log\lvert X\rvert)$	$k\log\lvert k\rvert$
Liu-Ning-Li [91, 96]	Hybrid	Scalable	Given in [91, Section 4.1]	t-secure	$s'(t+1)\log q$	$s'\log\lvert\mathcal{F}\rvert$	$t+1$
Zhu et al [182]	Probabilistic	Scalable	1	Given in [182, Section 4.1]	k	$O(\log N)$	n, n number of shares
Grid-based Schemes							
PIKE [27]	Deterministic	Not Scalable	$1/\sqrt{N}$	$1/\sqrt{N}$	$O(\sqrt{N})$	$O(\log N)$	$O(1)$
Kalindi et al [75]	Deterministic	Not Scalable	Given in [75, Section IIB]	$1-\frac{12m-6}{N}$	$6(\sqrt{N}/l)$	$O(\log N)$	1
Sadi-Kim-park [139]	Probabilistic	Not Scalable	$2/(\sqrt{N}+1)$	t-secure	$O(\tau t\log q)$	$O(\tau\log N)$	$O(\tau\log\sqrt{N\omega})$
Mohaisen-Maeng-Nyang [110]	Probabilistic	Not Scalable	$\frac{3}{\sqrt[3]{n}+1}$	Given in [110, Section 3.6]	$O(\sqrt[3]{n})t\log q$	$O(\log N)$	$O(\tau\log\tau)$
Group-based Schemes							
Liu-Ning-Du [95](Hash-key)	Deterministic	Not Scalable	Given in [95, Section 5.2]	Given in [95, Section 5.4]	$(m+n)/2$	$O(\log N)$	$O(H)$, H is the time to compute hash functions
Liu-Ning-Du [95](Polynomial)	Hybrid	Scalable	Given in [95, Section 5.4]	Given in [95, Section 5.4]	$\frac{(m+n)/2}{t}\log q$	$O(t+1)$	$O(t+1)$
Martin-Paterson-Stinson [104]	Deterministic	Not Scalable	k/n if in the same group $\frac{n-1}{n^{2/\lambda}-1}+\frac{n^{2/\lambda}-n}{n^{2/\lambda}-1}mn$ if in different group	Given in [104]	$(m+1)(t+1)$	$O(\log N)$	$O(1)$
Combinatorial design based Schemes							
Çamtepe and Yener [18, 20] (Symmetric)	Deterministic	Not Scalable	1	$\frac{q-1}{q^2+q+1}$	$O(\sqrt{N})$	$O(\log N)$	$O(1)$
Lee-Stinson [86]	Deterministic	Not Scalable	$\frac{k}{r+1}$	$\frac{r-2}{b-2}$	k	$O(\log N)$	$O(1)$
Chakrabarti, Maitra and Roy [23, 24]	Hybrid	Not Scalable	Given in [23, 24]	Given in [23, 24]	$zk-\binom{z}{2}\frac{k}{r+1}$	$O(z\log N)$	$z\log\lvert X\rvert$
Dong et al [44]	Deterministic	Not Scalable	≈ 0.5	Given in [44]	$O(\sqrt[3]{N})$	1	$O(\sqrt[3]{N})$
Deployment-knowledge based Schemes							
Liu-Ning [92]	Deterministic	Not Scalable	Given in [44, Section 4.3]	Given in [44, Section 4.3]	$O((t+1)\log q)$	$O(1)$	$O(t)$
Du et al [47]	Hybrid	Scalable	Given in [47, Section 6]	t-secure	$\omega\tau\log q$	$\tau\log\omega$	$(t+1)\omega\log\omega$
Yu-Guan [175, 176]	Deterministic	Not Scalable	1	t-secure	$N(t+1)$	$t\log t$	$Nt\log t$
Huang et al [69, 70]	Probabistic	Scalable	Given in [69, 70]	Given in [69, 70]	$\tau(t+1)$	$O(\tau)$	ωN
Simonova et al [149]	Deterministic	Scalable	Given in [149]	Given in [149]	$O(\sqrt{N/k})$	$O(\log p')$	$O(1)$
Zhou et al [179]	Hybrid	Not Scalable	Given in [179](Section VIIB)	Given in [179](Section VIIA)	Four types of nodes	$O(N)$	Depends on the type of node

Table 2.1: A table that compares the different key predistribution schemes.

2.11 Traitor tracing

We now discuss several traitor tracing schemes and point out their areas of applications.

2.11.1 Chor-Fiat-Naor scheme

Traitor tracing schemes were first introduced by Chor, Fiat and Naor [33] which was extended in [34]. They presented three schemes, two open schemes and one secret scheme. The open schemes are based on hash functions. Each hash function maps the N users into a set of $2c^2$ decryption keys. The user's personal set consists of $O(c^2 \log N)$ decryption keys and the enabling block consists of $O(c^4 \log N)$ encrypted keys. The second scheme is based one "two level" hash function. This scheme has $O(c^2 \log^2 c \log N)$ keys per user and an enabling block of $O(c^3 \log^4 c \log N)$. The scheme can find at least one traitor with a probability of $1 - p$ $(0 < p < 1)$. Each user receives $O(c \log(N/p))$ decryption keys and has $O(c^2 \log(N/p))$ encrypted keys per enabling block. The tracing is probabilistic in all the three cases.

2.12 Threshold traitor tracing

Most tracing schemes are designed to operate against decoder which decrypts with a non-negligible success probability. Suppose a TV program is divided into one minute segments which are separately encrypted. A decoder which decrypts with a probability of 90% is expected to fail in the decoding of one out of ten minutes. Very few customers will be willing to pay for such a decoder. For this reason *threshold traitor tracing* was introduced by Naor and Pinkas [112]. Such schemes are designed to trace the source of piracy, provided the decoders decrypt with a probability greater than some threshold q (which is a parameter). Threshold tracing schemes find application where the pirate decoders have a decryption probability close to 1. The schemes however have no guarantee for their success against decoders with success probability smaller than q.

Threshold traitor tracing schemes find application in pay-per-view TV, online services or databases, where a charge is levied for access to all or certain records. In the threshold tracing scheme proposed by Naor and Pinkas [112], the communicated content is divided into blocks which are independently encrypted. A valid decoder contains keys which enable it to decrypt each block. These keys identify the decoder. If a pirate decoder contains sufficient keys to enable it to decrypt more than q fraction of the blocks, these keys are sufficient to identify at least one of the traitors. It is assumed that a pirate decoder which decrypts less than a q fraction of blocks is not useful and therefore it is not important to trace the source of its keys.

Scheme	Type	Scalability	Key Connectivity	Resiliency	Key Storage	Overhead	Computation
Heterogeneous Schemes							
SPINS [121]	Probabilistic	Scalable	1	0	k	Low	Low
LEAP [181]	Probabilistic	Not Scalable	1	Prevent Sybil attack	Given in [181, Section 4.3d]	$O(d^2/N)$	$O(N)$
Jolly et al [181]	Probabilistic	Not Scalable	S-S communicate through the gateways G-G communicate through the nodes	Given in [74](Simulation)	Sensor: 2 Gateway: $\|S\|+1$ Command node: $\|G\|+\|S\|$	Sensor $O(\log(\|S\|/\|G\|))$ Gateway $O(\log\|G\|)$	$O(1)$
Cheng-Agrawal [32]	Probabilistic	Scalable	"	t-secure	Sensor: 2 CH : $t\log q$ Sink node: $N+m$	Sensor: $\log\lceil N/m\rceil$ CH: $\log m$	$O(1)$
Das-Sengupta [40]	Deterministic	Scalable	"	t-secure	Sensor: $t\log q$ CH : $t\log q$ Sink node: $\lceil N/m\rceil+m$	Sensor: $\log\lceil N/m\rceil$ CH: $\log m$	$O(1)$
SecLEACH [116, 117]	Probabilistic	Scalable	Given in [117, Section 5.1]	Given in [117, Section 5.1]	Sensor: m	$\lceil N/m\rceil$	$O(H)$, H : time to compute hash function
Du et al [52]	Probabilistic	Scalable	Given in [52, Section 4]	Given in [52, Section 5]	L:K H >> k	$k\log\|X\|$	$k\log k$
HERO [97]	Probabilistic	Scalable	Given in [97, Section 4]	Given in [97, Section 5]	$N_n :\leq k$ $S_n : k$	k	$k\log k$
SHELL [174]	Deterministic	Scalable	1	Given in [174, Section 3.4]	Sensor : $c+k$	$k\log\|X\|$	$O(1)$

Table 2.2: A table that compares the different key predistribution schemes (Contd.).

The complexity of the q-threshold schemes depends on q. These schemes are more efficient for larger q. Naor and Pinkas devised two threshold schemes, first is a one-level threshold scheme and the second, a two-level scheme. Both the schemes make use of hash functions to map the users $\{1, 2, \ldots, N\}$ to the keys. The pirate decryption process is considered to be a black box. To carry on the tracing process, only probes are to be made to the decryption box, without breaking it open. The one level scheme has personal keys of length $\frac{4c}{3q}\log(N/p)$ and communication overhead of $4c$, whereas the length of the keys in the two-level scheme is $O(\log(c/p)\log(N/p))$. p is the probability that a traitor cannot be traced. The communication overhead for the two-level scheme is $O(c\log(\frac{c}{q\log(c/p)}))$.

2.13 Asymmetric traitor tracing

Till now we had assumed that the data supplier is honest. Asymmetric traitor tracing schemes are those in which the data supplier, when confronted with treachery obtains the information that she could not have produced on her own and is therefore a much better evidence. Sometimes the data supplier may forge some keys and give unauthorized access to some unauthorized person. When such a person is convicted how should we prove whether the keys she possessed were given by a coalition of users or the data distributor herself. Sometimes a situation may arise that the redistributed or unauthorized information is provided by some traitor who has access to the data supplier's equipment. Asymmetric traitor tracing was proposed by Pfitzmann [122] to solve this problem and is similar to asymmetric fingerprinting [123, 124]. In the asymmetric tracing scheme there is a third party called a *judge* or *arbiter*. In a trial protocol devised by Pfitzmann, the data supplier tries to convince the judge of a traced user being a traitor. The scheme works as follows.

1. No provider initialization is needed.

2. User initialization. The user generates a key pair of an asymmetric encryption scheme, gives the encryption key to the information provider, and signs that the decryption key corresponding to this encryption key will be used exclusively in the traitor tracing scheme. The information provider verifies this signature with the public key of this user.

3. Session sending. The enabling block contains the session key, encrypted under each encryption key of a legitimate user of the session.

4. Tracing. The simpler assumption is that a pirate decoder has been opened and the secret decryption key of a traitor has been found inside. The information provider finds out to which encryption key it belongs; if there is no simple predicate to decide this, she can use the ability to decrypt encrypted random messages.

5. Trial. The information provider first shows the judge the accused user's signature about the use of her encryption key, and the judge verifies it. Next, if the information provider has actually found a decryption key, she shows it as a proof, and the judge verifies that the decryption and encryption keys belong together in the same way as the information provider did. If the information provider has only found the index of a key, she has to hand the pirate decoder to the judge.

The above scheme results in enabling block linear in size N of users. The other construction is based on one way functions and 2-party computation and has shorter size of enabling block. Pfitzmann also introduced 3 party trials in which the accused user also takes part in the trial.

Pfitzmann also devised a way in which a symmetric scheme can be made fully frameproof, which means that no coalition of members can frame any other member not in the coalition. In Chor, Fiat and Naor's [33] scheme, the user who has contributed at least k/c keys is the traitor, where k is the number of keys in the pirate decoder. In a fully frameproof scheme, each user from whom at least k/c keys are found in the pirate decoder is called a traitor. If there are no such traitors then the output of the tracing is *failed*. By doing this if a coalition of more than c people collude, they cannot frame a innocent user.

Other works on asymmetric traitor tracing include one by Pfitzmann and Schunter [123], Pfitzmann and Waidner [124], Kurosawa and Desmedt [83], Komaki, Watanade, Hanaoka and Imai [82], Watanbe, Hanaoka and Imai [170] and Kiayias and Yung [81].

Pfitzmann and Waidner [124] showed a asymmetric scheme by combining the symmetric scheme of [33] and a two party protocol [29, 63]. Kurosawa and Desmedt [83] were the first to use public-key setting to the asymmetric traitor tracing problem. They derive lower bounds on sizes of keys and ciphertext for symmetric traceability schemes. The optimum schemes have strong connections to orthogonal arrays. They also show two practical asymmetric traitor tracing schemes with trusted third party called *agents* or *arbiter*. They show that no authorized user can be framed as a traitor if the data supplier and $c-1$ agents collude and the data supplier can be framed as a traitor even if the data supplier can detect a traitor and convince a judge without the help of agents.

2.13.1 Public-key asymmetric schemes

Watanabe, Hanaoka and Imai [170] proposed an asymmetric public key traitor tracing without trusted agents. The protocol has important properties that include non-repudiation, full frameproof, black-box traceability for asymmetric scheme. The scheme is based on the primitive proposed by Naor and Pinkas [113] called *oblivious polynomial evaluation*. *Oblivious polynomial evaluation* is a primitive in which a polynomial f is known only to one party say Bob and he lets another party say Alice to compute the value of $f(x)$ for an unknown input x, in such a way that

Bob does not learn about x and Alice does not learn anything about the function f. The intuition behind their scheme is that in a c-resilient traitor tracing scheme, the function required for generating the personal-key of the subscriber U_i is $f(i)$. In order to change it into an asymmetric scheme, the authors used the bivariate polynomial function $f(x, y)$ as the key generation function, where f is a polynomial of degree c in x and of degree 1 in y. The subscriber U_i chooses an integer α_i randomly and computes $f(i, \alpha_i)$, but the user U_i does not gain any additional information for $f(i, y)$. The user U_i is able to deduce its publicly verifiable proof from her knowledge of α_i without revealing the content of α_i. The scheme requires only two personal keys, $O(c)$ encryption keys and $o(c)$ ciphertexts.

Another asymmetric public-key traitor tracing was proposed by Kiayias and Yung [81]. They present an efficient asymmetric public-key traitor tracing scheme for which the traceability is proved in details. The system is capable of implicating *all* traitors that participate in the construction of the pirate-key.

Their scheme is also based on oblivious polynomial evaluation and involves agents. The scheme has the following properties (i) frameproof, the system manager is incapable of implicating innocent users in the construction of pirate decoder, (ii)non-repudiation: tracing should produce indisputable proof for the implication of traitors in the construction of the pirate decoders, such a proof should be impossible to forge by the system-manager and any interested third party (the judge) can check its validity without the participation of the subscribers of the system.

They show that previous proposals of asymmetric public key traitor tracing schemes are flawed. Previous schemes could be attacked using techniques to combine user-key information in an arbitrary fashion thus disabling *direct* tracing by merely observing the keys found inside the pirate-decoder. Though Watanabe et al [170] claimed that such technique do not apply to their scheme, Kiayias and Yung [81] show that their assumption is false. Besides they also show that the scheme given in [82] takes exponential time to trace the traitors.

2.13.2 Combinatorial Asymmetric Traitor Tracing Scheme

Another asymmetric traitor tracing was proposed by Safavi-Naini and Wang [140]. In this, only a part M of the fingerprint is inserted by the data supplier and part by the arbiter or the judge. When a pirate copy is found, the data supplier finds the user with which there is maximum matching with the copy. The data supplier then convicts the person as guilty. If the user U_j denies the fact, then the arbiter is called and the rest of the mark is also checked. Suppose the copy F matches at a_F places. If $a_F - |F \bigcup P(U_j)| \leq m$, where m is a predefined threshold, then arbiter accepts the acquisition, else rejects it. The authors used two constructions, one based on polynomials over finite fields and other based on orthogonal arrays for asymmetric traitor tracing.

2.14 Dynamic traitor tracing schemes

In all the above schemes the underlying assumption was that the unauthorized users are able to access decoder boxes and decrypt the original content that is broadcasted. Sometimes the traitors may rebroadcast the content. In such a case, the above schemes fail. This is where dynamic traitor tracing schemes are used - where the content is rebroadcasted. Dynamic traitor tracing scheme was first introduced by Fiat and Tassa in [58] which was later extended in [59]. The dynamic traitor tracing schemes find out the traitors and disconnect them from the system.

To facilitate this, the entire content is broken down into segments. *Watermarking schemes* are used to each segment. The different watermarks for each segment are called *versions* of the segment. The watermarks present in the broadcasted content help to trace the traitors and such traitors are disconnected from the system. Cox, Kilian, Leighton and Shamoon [38] introduced methods to create secure and robust watermarks. The basic idea of dynamic traitor tracing schemes is that marks are generated on the fly based on the feedback from the pirate network. The traitors are found on the fly, contrary to the static schemes where a bound is preset based on the number of traitors.

The segments may be one minute of video. Each segment has different variants depending on the mark it possesses. The dynamic traitors tracing schemes consists of the following two steps.

1. Watermark distribution : an algorithm that assigns each subscriber a watermarked copy of the content.

2. Tracing and incrimination : an algorithm that given an illegal copy of the content, the watermarks embedded in it are used to trace back the traitors who had broadcasted the contents.

The dynamic tracing scheme incriminates all traitors and does not convict any innocent user and is based on real time calculation of marked sequences. The center distributes the copies to the users. Hidden within the copies are watermarks. The marks are selected from the marking alphabet $\sum = \{\sigma_1, \sigma_2, \ldots, \sigma_r\}$. For a given segment $1 \leq j \leq m$ and a mark σ_k, $1 \leq k \leq r$, $S_k^j \subset U$ denotes the subset of subscribers that got variant σ_k of segment j. One way of assigning marks will be that each segment is marked with N versions and given to N users. However this involves a huge bandwidth.

Every user has a unique symmetric key in common with the center. Prior to the segment transmission, the center distributes keys to users using individually encrypted transmissions : if user U_i is to get variant l of segment j, then the center sends an individually encrypted transmission to user U_i containing key k_l^j, where all such keys are generated at random. The center then transmits multiple variants of the jth segments, where variant l is encrypted under key K_l^i. The overhead consists of two parts.

1. The center needs to transmit individual messages that contain the relevant keys to every user.

2. The center needs to broadcast multiple variants of every segment, this is a high overhead component, because it multiplies the total bandwidth by the number of different variants.

Two things are to be considered in any dynamic tracing scheme.

1. The number of versions must be kept low, to reduce the bandwidth of communication.

2. The time taken to find out all the traitors i.e. the convergence time should be kept low.

There is a trade off between the above two parameters. Fiat and Tassa proposed three dynamic traitor tracing schemes [58, 59]. The first scheme makes used of $r = c + 1$ versions and has very high convergence time. Here c is the number of traitors. The second scheme makes use of $r = 2p + 1$ versions and converges in $c \log n + c$ time steps. The third scheme makes use of $r = c + 1$ versions and converges in $2.3^c c \log N + c$ time steps.

The dynamic traitor tracing scheme was improved by Berkman, Pranas and Sgall [5, 6]. They reduced the problem to a graph theoretic problem, in which $G = (V, E)$ is a (t, l)-graph, if

1. G contains $t+l+1$ vertices, one of which is the special vertex I. The vertices are subsets of users, all of them except possibly I are non-empty and every user belongs to exactly one vertex.

2. For any edge $(X, Y) \in E$, the subset $X \cup Y$ contains a traitor.

3. The vertices in $V \setminus I$ are partitioned into k disjoint cliques $\{Q_1, Q_2, \ldots, Q_l\}$, where $|Q_i| = t \geq 2$, for each $1 \leq i \leq l$.

Let G be a (t, l)-graph. A clique Q of t_i vertices contains at least $t_i - 1$ traitors and G contains at least t traitors. The number of vertices of a (t, l)-graph is at most $2t + 1$.

Berkman, Pranas and Sgall [5,6] gave three graph based algorithm. In the first algorithm using $c + 1$ versions, all traitors can be found in $\Theta(c^2 + c \log N)$ steps. The second algorithm uses $c+a+1$, $a \geq 1$ and convergence time $O(c^2/a+c \log N)$ whereas the third algorithm has a size $ca - 1$ and converges in $O(c \log_a N)$ steps. The authors also proved a lower bound of $\Omega(c^2/a - c \log_{a+1} N)$ steps.

2.15 Sequential Traitor Tracing

Dynamic traitor tracing suffers from two drawbacks. Firstly it is completely inefficient against *delayed broadcast*, in which the attackers rebroadcast the content with some delay. When such a condition arises, then not even a single colluder can be found. Secondly, it requires high real time computation and so is not suitable for large number of users. To alleviate these two problems, Safavi-Naini and Wang introduced sequential traitor tracing in [141] which was extended in [143]. The main difference between dynamic and sequential traitor tracing is that, while in dynamic traitor tracing the segments are marked depending upon the feedback from the pirate network, the marks are assigned according to a precomputed table in sequential traitor tracing. The feedback is only used to trace the traitors. So on one hand it does not involve real time computations and on the other that even if there is delay in broadcasting, the traitors are found.

In sequential traitor tracing first the data supplier creates a mark allocation table. As in dynamic traitor tracing, the whole content is broken down into segments. Each segment has several versions each having a mark allocated to it. The mark allocation table contains marks to be assigned to each version in each segment. The feedback from the broadcast is noted. By observing the feedback sequence and counting the number of matches with the users, the traitors can be found. For N users, if there is a q-ary marking alphabet $\mathcal{W}$, then a mark allocation table is an $N \times L$ array with entries from $\mathcal{W}$. L is the *convergence length* of the system. Given the feedback sequence $\mathcal{F}_k = (f_1, f_2, \ldots, f_k)$, each mark is f_i extracted. If the number of matches between $\mathcal{F}_k$ and user U_i is $c + 1$, then u is a traitor. It has been shown that to find one traitor, maximum number of steps taken is $c^2 + 1$ steps. Once a traitor is found, she is disconnected from the system and the feedback is again observed. All the traitors can be found in $c^2 + c$ steps, and are discarded, one at each step.

Safavi-Naini and Wang used two constructions for the mark allocation table, one using function family and the other using error correcting codes. They also show that the mark allocation table in a sequential TA scheme gives a sequence of c-TA schemes.

2.16 Public traitor tracing schemes

Boneh and Franklin [11] were the first to propose a public traitor tracing scheme. The tracing method is deterministic and all the traitors can be found, contrary to finding at least one traitor in previous schemes. Also innocent users are never accused as long as less than or equal to c colluders collude. The tracing method is partially "black box", where the pirate decoder can be queried but private keys cannot be extracted.

In the public scheme, there is one public encryption key e and each user has its own personal decryption key $d_1, d_2, \ldots d_k$. The scheme is a open scheme. Each

private key is a different solution vector for the discrete log representation problem with respect to a fixed base of field elements. The pirate can form new keys which are convex combination of stolen keys. If every set of $2c$ keys is linearly independent, then every convex combination of c keys can be traced uniquely. The keys are derived from Reed Solomon code so takes advantage of efficient error correction methods to trace uniquely and efficiently. The main assumption of the traceability scheme is that discrete log problem is hard. The encryption scheme is secure, if the decision Diffie-Hellman problem is hard.

Kurosawa and Desmedt [83] proposed another public-key traitor tracing scheme. According to their scheme, the key distribution center chooses a polynomial $f(x)$ of degree c over Z_q and gives user U_i the value of the polynomial $f(i)$. Now if a group of c users collude, they cannot construct the value of the private key). The scheme has been broken [11, 159].

2.16.1 Public-key schemes using pairing and bilinear maps

Several traitor tracing schemes have been proposed using pairings and bilinear maps. Mitsunari, Sakai and Kasahara [146] were the first to use Weil pairing in elliptic curves for traitor tracing. The advantage of the scheme is that the ciphertext size is independent of the number of traitors. It is shown that the problem of constructing a pirate key by c colluders is as hard as the so-called "c-weak Diffie-Hellman problem".

Mitsunari et al's [146] scheme was broken by Tô, Safavi-Naini and Zhang [165]. In this paper they also propose three new traitor tracing schemes using bilinear maps. The first scheme is a modification over Mitsunari et al's scheme. It has the added advantage that the construction of the pirate decoder is more complex thus deterring traitors to collude and construct the pirate decoder. The scheme is a public key traitor tracing scheme, as opposed to Mitsunari et al scheme which was secret key tracing scheme. The new scheme has black box tracing, while the original scheme used open box tracing algorithm. Tô et al give a security proof, which was not present in Mitsunari et al's scheme. The second scheme proposed by Tô et al is a generic scheme and can be used with any linear error-correcting codes. The third scheme uses Shamir's secret sharing scheme and has the added property that the encrypted message can be targeted to a subset of users.

All the proposed schemes are shown to be semantically secure against passive adversary assuming the hardness of the decision bilinear Diffie-Hellman problem. The size of ciphertext component in the schemes are smaller because each component is a point on the elliptic curve. The first and the second schemes use one pairing operation, while the third scheme needs two pairing operations. Since pairing is a very expensive operation, the first two schemes have the best performance if the size of the group is fixed.

2.16.2 Other Public-key Schemes

Boneh, Sahai and Waters [13] presented a traitor tracing system that uses bilinear maps in groups of composite order. The system generates ciphertexts of size $O(\sqrt{N})$ (where N is the number of users) and private keys of size $O(1)$.They first construct a *private linear broadcast encryption (PLBE)* and then show that any PLBE gives a traitor tracing system with the same parameters. The decryption time is constant (i.e. depends on the security parameter, but not on N). Other properties of this system include:

1. The broadcasters key BK is public, but the tracers key TK must be kept secret,

2. The system is black-box traceable, and

3. Is designed for stateless pirate decoders.

The PLBE technique is conceptually a simpler primitive than traitor tracing. The authors show that any secure PLBE gives a (black-box) traitor tracing system. Roughly speaking, a PLBE is a broadcast encryption system [57] that can only broadcast to "linear" sets, that is sets of the form $\{U_i, U_{i+1}, ..., U_N\}$ for some $i = 1, \ldots, N+1$. Thus, a PLBE enables the broadcaster to create ciphertexts that can only be decrypted properly under keys K_i, K_{i+1}, ..., K_N. A broadcast to everyone, for example, is encrypted using $i = 1$. The main security requirement is that the system should be private [4]: a ciphertext should reveal no non-trivial information about the recipient set. That is, a broadcast to users $\{U_i, U_{i+1}, \ldots, U_N\}$ should reveal no non-trivial information about U_i. A traitor tracing scheme with constant size ciphertext was proposed by Boneh and Naor in [12]. The ciphertext size is independent of the number of users in the system and the collusion bound. The ciphertext consists of only two elements, the length of which depends on the security parameter. If λ be the security parameter, the the ciphertext length is $O(\lambda)$, the secret key length is $O(c^2\lambda^2 \log N)$ and the tracing time is $O(c^2\lambda \log N)$.

2.17 Combinatorial traitor tracing schemes

The first traitor tracing scheme proposed by Chor, Fiat and Naor [33] was combinatorial scheme. Thereafter several authors have proposed traitor tracing schemes using combinatorial methods. Stinson proposed a unconditional method for secure key distribution pattern in [153]. There after Stinson, Wei and Trung proposed several combinatorial methods of key predistribution and traitor tracing schemes in [156,158–160]. Later Staddon, Stinson and Wei [151] studied the different properties of related structures like traceability codes (TA-codes) [158], Frameproof codes (FP-codes) [14, 15], Secure Frameproof codes (SFP-codes) [157], codes with Identifiable Parent Property (IPP-codes) [68]. These results were improved by

Safavi-Naini and Wang [142]. Most of the techniques are based on set systems or designs. They several designs like BIBD designs, transversal designs, orthogonal arrays, perfect hash functions, separating families, cover free families.

Other combinatorial techniques include resolvable BIBDs, PBIBDs by McNicol, Boztaz and Rao [105–107]. List decoding techniques were introduced by Silverberg, Staddon and Walker in [147].

2.17.1 Related combinatorial structures

Several key distribution patterns were presented by Stinson in [153]. These methods are used by truster authorities to distribute keys and broadcast messages over a network, such that each member of the privileged set of users can compute a specified key and decrypt the message, whereas no person not in the privileged set can do so. The problems were studied using tools of information theory. The first part of the article was dedicated to existing schemes of Blom [9], Blundo et al [10], Fiat and Naor [57] schemes. They studied the information rate of each of these schemes and proposed a efficient improvement using resilient functions. Two constructions of one-time broadcast encryption were presented. Then a broadcast encryption scheme was constructed combining several key predistribution schemes with an ideal secret sharing scheme.

In [158] Stinson and Wei presented combinatorial properties and constructions of traceability schemes and frameproof codes. They show that the existence of a c-traceability scheme implies the existence of a c-Frameproof code (c-FP code). Frameproof codes were first introduced by Boneh and Shaw in [14]. A code is c-FP if no coalition of size at most c can frame another user not in the coalition by producing the codeword held by that user.

Suppose any exposed user U is a member of the coalition $\mathcal{C}$ whenever a pirate decoder F is produced by $\mathcal{C}$ and $|\mathcal{C}| \leq c$. Then the scheme is called a c-traceability scheme and is denoted by $c - TS(k, b, v)$, where k is the number of keys given to each of the b users. A c-TA code (traceability code) is one such that no coalition of size at most c can produce a N tuple that word be traced to at least one member of the coalition. Also it has the property that allows an efficient algorithm to determine the identifiable parent. Stinson and Wei gave several construction of frameproof and traceability schemes from combinatorial designs. They used t-designs, inversive planes, packing designs, error-correcting codes and perfect hash families. They also investigated embeddings of frameproof codes and traceability codes so that a given scheme can be expanded at a later date to accommodate more users. They further proved the following bound on traceability schemes. If a $c - TS(k, b, v)$ exists, then the following bound holds.

$$b \leq \frac{\binom{v}{t}}{\binom{k-1}{t-1}},$$

where $t = \lceil \frac{k}{c} \rceil$.

$$
\begin{array}{ccccc}
 & & \text{Code with } d > N(1-1/c^2) & & \\
 & & \Downarrow & & \\
 & & (c, 1-1/c)\text{-CF} & & \\
 & & \Downarrow & & \\
 & & c\text{-TA} & & \\
 & & \Downarrow & & \\
\lfloor \frac{(c+2)^2}{4} \rfloor\text{-PHF} & \Longrightarrow & c\text{-IPP} & \Longrightarrow & (c+1)\text{-PHF} \\
 & & \Downarrow & & \\
(c,c)\text{-SHF} & \Longleftrightarrow & c\text{-SFP} & & \Downarrow \\
 & & \Downarrow & & \\
 & & c\text{-FP} & \Longleftrightarrow & (c,1)\text{-SHF} \\
 & & \Downarrow & & \\
 & & c\text{-CFF} & &
\end{array}
$$

Table 2.3: Relation between different types of codes and combinatorial structures

Stinson and Wei's results [158] were extended by Staddon, Stinson and Wei [151]. They show the relation between FP-code, SFP-code, IPP-code, and TA-code. A code is c-SFP if no coalition of size at most c can frame a disjoint coalition of size at most c by producing a N tuple that could have been produced by the second coalition. A code has c-IPP property if no coalition of size at most c can produce a tuple N that cannot be traced back to at least one member of the coalition, where N is the length of the code. They show that c-IPP implies c-SFP and c-SFP implies c-FP. They also show that c-TA implies c-IPP. They show the correspondence between separating hash families, perfect hash families and traceability codes. The relation between FP-codes, SFP-code, IPP-code, TA-codes, Cover free code (CF-code), cover-free family (CFF), separating hash family (SHF) and perfect hash family (PHF) is given in Table 2.3.

2.17.2 Combinatorial threshold traitor tracing schemes

Key preassigned traceability was discussed by Stinson and Wei in [159]. In this paper the authors discuss methods to predistribute keys to users, so that only users in a specified privileged subset can decrypt. They also present a new threshold traceability scheme by using ramp scheme. In this paper the authors describe a new construction of traceability scheme in which the secret key is split into shares, issuing a threshold scheme (or a ramp scheme), and then the shares are encrypted, thus forming the enabling block. A threshold traitor tracing scheme has been constructed from orthogonal arrays. Other construction of traceability schemes have been done from set systems using structures like t-designs and inversive planes.

Another paper which discusses threshold traitor tracing is by McNicol, Boztas and Rao [107]. They embed digital patterns into the distributed content, so that

it is possible to trace the source of piracy. They propose new codes derived from combinatorial designs and develop a method for concatenating these codes and defend against some of the attacks. They preset four types of attacks on digital fingerprinting codes. The narrow attack creates codes such the alphabets at any position belongs to either of the codewords. In the erasure attack, the attacker can erase the marks where marks differ. In a wide attack, the attacker can put any arbitrary value in a position which does not belong to either of the codewords. Hybrid attack is a combination of the above attacks. The authors use resolvable BIBDs to construct new code families called VSBDC (Vector Space Block Design) codes. These codes with concatenation techniques are resistant to the above four attacks.

2.17.3 Traitor tracing using List decoding

Traitor tracing schemes using List decoding was presented by Silverberg, Staddon and Walker [147, 148]. They use techniques from error-correcting codes to trace traitors. They compare the TA and IPP traitor tracing algorithm and give evidence that when using an algebraic structure, the ability to trace traitors with IPP implies the ability to trace with the TA algorithm. In list decoding the input is a received word and the output is the list of all codewords within a given Hamming distance of the received codeword. The list decoding techniques result in very fast traitor tracing. The error correcting techniques are used to attain traceability.

2.18 Concluding Remarks

In this chapter we presented the state of art work in context of key predistribution in sensor network and traitor tracing. In the next Chapter we present our first key predistribution scheme for sensor networks. The scheme uses PBIBD as its basic building block.

Chapter 3

Key Predistribution using Partially Balanced Incomplete Block Designs

In this chapter we propose two deterministic key predistribution schemes in a Wireless Sensor Network (WSN), in which sensor nodes are randomly deployed. Both the schemes are based on Partially Balanced Incomplete Block Designs (PBIBD). An important feature of our scheme is that every pair of nodes can communicate directly, making communication faster and efficient. The number of keys per node is of the order of $\sqrt{N}$, where N is the number of nodes in the network. The second design has the added advantage that we can introduce new nodes in the network without redistributing the keys in the existing nodes. We study the resiliency of the network under node compromise and show that our designs perform better than the existing ones.

We will consider the dual of a PBIBD $PB[k, \lambda_1, \lambda_2; v]$. Let us consider a network containing v nodes containing r keys each. So any two nodes share either λ_1 or λ_2 ($\lambda_1, \lambda_2 \neq 0$) common keys. Unlike most of the previous designs, our design ensures that any two nodes can communicate directly thus making communication faster and efficient.

We can model of the network as a graph, in which the nodes are represented by vertices. An edge is said to be present between two nodes (vertices) if the nodes share at least one key in common. This helps us to define two measures of resiliency. One of the measures consider the proportion of nodes disconnected when a certain number of nodes are compromised. We recall the definitions of $V(s)$ and $E(s)$ from Section 1.2.4. We show that our design yields better connectivity and resiliency compared to other schemes

The rest of the Chapter is organized as follows. In Section 3.1 we discuss the triangular PBIBD design, which we use to construct the key predistribution schemes. We present a key predistribution scheme based on triangular PBIBD in Section 3.2. In Section 3.4 we study the resiliency of the network under node

capture. We give an upper bound for $E(s)$ and $V(s)$. In Section 3.5 we discuss another design and conduct an experimental study of $E(s)$ and $V(s)$ on that design. We compare our results with existing schemes in Section 3.6 and show that our design fares better in terms of resiliency and connectivity. This chapter is based on the papers [132, 134, 136].

3.1 Triangular Partially Balanced Incomplete Block Design

We recollect the definitions of Association schemes and Partially Balanced Incomplete Block Designs (PBIBD) from Definitions 1.5.8 and 1.5.9 in Section 1.5.

We use PBIBD with two associate classes in our design. We use a triangular association scheme [36, Page 14] in our design.

A *triangular association scheme* is a partially balanced design with two associate classes in which the number of varieties is $v = n(n-1)/2$ and the association scheme is an array A of n rows and n columns with the following properties:

1. The positions in the principal diagonal (from top left to bottom right) are left blank.
2. The $n(n-1)/2$ positions above the principal diagonal are filled by the numbers $1, 2, \cdots, n(n-1)/2$ corresponding to the varieties.
3. The $n(n-1)/2$ elements below the diagonal are filled so that the array is symmetrical about the principal diagonal.
4. For any variety i, the first associates are those elements which lie in the same row (or same column) as i, the second associates are the rest of the elements.

The parameters of the association scheme are

$$v = n(n-1)/2, \quad n_1 = 2(n-2), \quad n_2 = (n-2)(n-3)/2$$

and

$$P_1 = \begin{pmatrix} n-2 & n-3 \\ n-3 & (n-3)(n-4)/2 \end{pmatrix}, \; P_2 = \begin{pmatrix} 4 & 2n-8 \\ 2n-8 & (n-4)(n-5)/2 \end{pmatrix}.$$

3.2 A key predistribution scheme using triangular PBIBD

We use a triangular association scheme to predistribute the keys in the sensor network. Corresponding to any variety x ($x = 1, 2, \cdots, v$) of the triangular association scheme with parameter n ($n \geq 5$) we form a block of size n_1 by taking as elements in the block the n_1 first associates of the variety x. We use the association corresponding to the array A given by:

$$A = \begin{array}{cccccc} * & 1 & 2 & \cdots & n-2 & n-1 \\ 1 & * & n & n+1 & \cdots & 2n-3 \\ 2 & n & * & 2n-2 & \cdots & 3n-6 \\ \vdots & \vdots & \vdots & * & \vdots & n(n-1)/2 \\ n-1 & 2n-3 & \cdots & \cdots & n(n-1)/2 & * \end{array}$$

Thus $n(n-1)/2$ blocks are formed, where each element appears n_1 times and each block is of size n_1. We also note that the varieties that are first associates occur together in p_{11}^1 blocks. Thus

$$v = b = n(n-1)/2,\ r = k = n_1 = 2(n-2),$$
$$\lambda_1 = p_{11}^1 = (n-2),\ \lambda_2 = p_{11}^2 = 4.$$

This is a symmetric design so the number of common keys between the blocks will either be $\lambda_1 = p_{11}^1$ or $\lambda_2 = p_{11}^2$.

For the array A we denote the entry in the i-th row and j-th column by a_{ij}. We present an example to demonstrate the above scheme. We consider $n = 5$. Then the array A containing the varieties will be represented by

$$A = \begin{array}{ccccc} * & 1 & 2 & 3 & 4 \\ 1 & * & 5 & 6 & 7 \\ 2 & 5 & * & 8 & 9 \\ 3 & 6 & 8 & * & 10 \\ 4 & 7 & 9 & 10 & * \end{array}$$

In the above design, $v = b = 10$, $r = k = 6$, $\lambda_1 = 3$, $\lambda_2 = 4$. The blocks so formed are: $\{2,3,4,5,6,7\}$, $\{1,3,4,5,8,9\}$, $\{1,2,4,6,8,10\}$, $\{1,2,3,7,9,10\}$, $\{1,2,6,7,8,9\}$, $\{1,3,5,7,8,10\}$, $\{1,4,5,6,9,10\}$, $\{2,3,5,6,9,10\}$, $\{2,4,5,7,8,10\}$, $\{3,6,8,4,7,9\}$.

Let $f(x,y)$ denote the entry in the location (x,y) of the array A. Then $f(x,y)$ is given by

$$f(x,y) = \begin{cases} *, \text{ for } x = y \\ y - x, \text{ for } x = 1, x < y \\ x - y, \text{ for } y = 1, x > y \\ n + y - x - 1, \text{ for } x = 2, x < y \\ n + x - y - 1, \text{ for } y = 2, x > y \\ (x-1)n - (x+1)(x-2)/2 + (y - x - 1), \\ \qquad \text{for } x < y, x > 2 \\ (y-1)n - (y+1)(y-2)/2 + (x - y - 1). \\ \qquad \text{for } x > y, y > 2 \end{cases} \tag{3.2.1}$$

Designs	Sensor networks
Set of elements X	Pool of key identifiers
Blocks $\mathcal{A}$	Sensor nodes
Element $x \in X$	Key identifier
Elements in a Block	Key chain
k	Size of a key chain
Replication number r	Number of nodes in which a given key is present
λ	Number of nodes in which a given pair of keys are present

Table 3.1: Mapping between set systems and sensor networks

We now map this design to the our key predistribution scheme. This is done in Table 3.1. This same map can be applied to any network that uses designs for key predistribution. We assume a DSN where there are $N = n(n-1)/2$ nodes. The total number of keys in the key pool is also $n(n-1)/2$. Each sensor node contains $2(n-2)$ keys. Each node is identified by the position in the array A. Let $K^{(i)}$ denote the identifier of the i-th key in any node. Algorithm 1 presents the key predistribution scheme.

Algorithm 1 Key predistribution

```
1: for a node at position (x, y) in array A do
2:    j = 0
3:    for i = 1 to n do
4:       if i ≠ x and f(i, y) ≠ ∗ then
5:          K^(j) = f(i, y)
6:          j = j + 1
7:       end if
8:    end for
9:    for i = 1 to n do
10:      if i ≠ y and f(x, i) ≠ ∗ then
11:         K^(j) = f(x, i)
12:         j = j + 1
13:      end if
14:   end for
15: end for
```

Each block corresponds to a key chain. The elements in the blocks corresponds to the identifiers of the keys in the node. Since the above design is symmetric, we note that any two blocks will share either $n-2$ or 4 keys. Let U_P and U_Q be two nodes whose positions in the matrix A are given by (x_1, y_1) and (x_2, y_2). This implies that U_P lies in x_1th row and y_1th column and U_Q lies in x_2th row and y_2th column. If $x_1 = x_2$ (when nodes belong to the same row), then the

identifiers of the keys shared between them will be the set $\{a_{x_1 j} : 0 \leq j < n, j \neq x_1, y_1, y_2\} \bigcup \{a_{y_1 y_2}\}$. $n-2$ keys are similarly shared when two nodes belong to the same column. When U_P and U_Q belong to different rows and columns, then four keys having identifiers $a_{x_1 x_2}$, $a_{x_1 y_2}$, $a_{y_1 x_2}$ and $a_{y_1 y_2}$ are shared between U_P and U_Q. So every pair of node can communicate with each other directly. If we number the nodes $1, 2, \cdots, n(n-1)/2$, then any two nodes which lie in the same row (or column) of A will share $n-2$ keys. Any other pair of nodes will share four keys. We observe that if the number of sensor nodes is N, then the number of keys present in each sensor node is $O(\sqrt{N})$.

For our first design we consider a network containing a maximum of $N = n(n-1)/2$ sensor nodes, each node containing $k = 2(n-2)$ keys, and any two nodes have either $\lambda_1 = n-2$ or $\lambda_2 = 4$ keys in common.

3.3 Key establishment

In this section we present key establishment algorithm

Let nodes U_P and U_Q want to communicate with each other. For this purpose we store the location of the node in the array A. The nodes broadcast their position in the array A. We need to calculate a simple function which will give the identity of one or more common key between any two nodes. Given the position (x, y) of a node U_P the value in the matrix at position (x, y) is given by Equation 3.2.1.

Given any node U_P it can find the ids of the keys in common with another node U_Q at position (x', y') in the following way.

1. If $x = x'$, then $a_{f(x,t)}$ and $a_{f(y,y')}$ are the common keys between U_P and U_Q for $t = 1, 2, \cdots, n$ and $t \neq x, y, y'$.

2. If $y = y'$, then $a_{f(t,y)}$ and $a_{f(x,x')}$ are the common keys between U_P and U_Q for $t = 1, 2, \cdots, n$ and $t \neq x, y, x'$.

3. If $x \neq x'$ and $y \neq y'$, then the keys $a_{f(x,x')}$, $a_{f(x,y')}$, $a_{f(y,x')}$ and $a_{f(y,y')}$ are common between U_P and U_Q.

Since there are more than one key in common, the nodes can choose any of the common keys. Since $f(x, y)$ can be calculated in constant time, key agreement can be done in $O(1)$ time. Also the memory overhead is $O(log\ n) = O(log\sqrt{N})$ bits, since only the position of the node in the array is sent.

3.4 Study of resiliency

Sensor nodes when deployed in an hostile environment are prone to be captured or compromised by adversaries. When nodes are compromised, the keys contained therein are exposed and so cannot be used for communication. In some cases

only the links that share the exposed keys will be broken. In other cases, sensor nodes may contain all the keys that have been exposed. In such a situation, the node will be disconnected altogether. We first recall $V(s)$ and $E(s)$, defined in Section 1.2.4. We give an upper bound for $V(s)$ and $E(s)$ and compare them with the experimental values.

3.4.1 Analysis of $V(s)$

Recall the definition of $V(s)$ from Section 1.2.4. It has already been pointed out that any two sensor nodes share either $n-2$ keys or four keys. We recollect that any two nodes which lie in the same row (or column) of A will share $n-2$ keys. We consider any two nodes U_P and U_Q. Suppose they belong to positions (x_1, y_1) and (x_2, y_2) in the array A (as defined in Section 3). Since matrix A is symmetric about the principal diagonal, position of U_P and U_Q can also be stated as (y_1, x_1) and (y_2, x_2) respectively. If U_P and U_Q belong to the same row, then $x_1 = x_2$. Then the common keys between U_P and U_Q will be $\{a_{x_1 i} : 0 \leq i < n, i \neq x_1, y_1, y_2\} \bigcup \{a_{y_1 y_2}\}$. So $n-2$ keys are shared between U_P and U_Q. If U_P and U_Q do not belong to the same row (or column), then the four common keys will be in positions $a_{x_1 x_2}$, $a_{x_1 y_2}$, $a_{y_1 x_2}$ and $a_{y_1 y_2}$. It is easy to see that if only one node is compromised, then no node is disconnected.

Proposition 3.4.1. *For $n > 5$, when two nodes in the same row (or column) are compromised, then exactly one node will be disconnected.*

Proof. Suppose two nodes U_C and $U_{C'}$ at positions (x_1, y_1) and (x_1, y_2) are compromised. Due to the compromise of U_C, the keys, $a_{x_1 i}$ and $a_{j y_1}$ are lost, where $0 \leq i, j < n$, $i, j \neq x_1, y_1$. Also, due to the compromise of $U_{C'}$, the keys, $a_{x_1 l}$ and $a_{m y_2}$ are lost, where $0 \leq l, m < n$, $l, m \neq y_2, x_1$. So all keys, $a_{y_1 i}$, where $0 \leq i < n$ and $i \neq y_1$ and $a_{j y_2}$, where $0 \leq j < n$ and $j \neq y_2$ are lost. Thus all keys belonging to node at (y_1, y_2) are exposed. Hence the node at (y_1, y_2) is disconnected. ∎

For example if node at position $(2, 3)$ and at position $(2, 4)$ are compromised, then the node at location $(3, 4)$ is compromised.

Observation 1: It can be noted that any two nodes belonging to different row (or column) in array A will not disconnect any other node.

Observation 2: It follows from Proposition 3.4.1 and by Pigeon Hole Principle, that on compromising $\lceil \frac{n}{2} \rceil + 1$ nodes, at least two nodes will be in the same row or column. So at least one node will be disconnected.

Observation 3: If $\lceil \frac{n-2}{2} \rceil$ (n odd) nodes are so compromised that the row and column to which they belong are all distinct, then only one node is disconnected. Suppose, nodes at position (a_0, a_1), (a_2, a_3), $\cdots$, (a_i, a_{i+1}), (a_{i+3}, a_{i+4}), $\cdots$, (a_j, a_{j+1}), (a_{j+3}, a_{j+4}), $\cdots$, (a_{n-2}, a_{n-1}) are compromised. Then the node at position (a_{i+2}, a_{j+2}) has all the keys common to the compromised nodes and so it will be compromised.

n	N	k	s	$V(s)$ Experimental	Upper bound for $V(s)$
20	190	38	7	0.0765	0.1147
30	435	56	10	0.0753	0.1059
40	780	76	10	0.0351	0.0584
50	1225	96	10	0.0156	0.0370
60	1770	116	10	0.0085	0.0255
70	2415	136	10	0.0058	0.1871

Table 3.2: Experimental value of $V(s)$ for 100 runs and bound for $V(s)$, when number of nodes is $N = n(n-1)/2$ and keys per node is k

Theorem 3.4.2. *Maximum number of nodes disconnected when s ($s < n$) nodes are compromised is $s(s-1)/2$.*

Proof. From Proposition 3.4.1 when any two nodes which belong to the same row (column) of array A are compromised, one node is disconnected. For every pair of compromised nodes in the row (column), one node is disconnected. Hence, for $\binom{s}{2}$ pairs of compromised nodes $\binom{s}{2}$ nodes are disconnected. This is the maximum number of nodes disconnected, since by Observation 1, any two node not belonging to the same row (column) will not disconnect any other node. ■

We give some experimental values for $V(s)$. The results in Table 3.2 clearly show that very few nodes will be disconnected when nodes are compromised randomly. The values of $V(s)$ have been obtained by compromising s nodes randomly and the experiment is conducted for 100 runs for each s. The results for the upper bound show that even if nodes are compromised in a predetermined manner, very few nodes (other than the compromised nodes) will be disconnected.

3.4.2 Analysis of $E(s)$

Let L be the number of links broken when s nodes are compromised. $E(s)$ is the proportion of links disconnected when s nodes are compromised, that is,

$$E(s) = \frac{2L}{N(N-1)}.$$

Let the s nodes being compromised be $C_1, C_2, \cdots, C_s$ which belong to positions (x_1, y_1), (x_2, y_2), $\cdots$, (x_s, y_s) in array A.

Given any two nodes U_P and U_Q belonging to positions (x'_1, y'_1) and (x'_2, y'_2) in the array A, we can check whether the link PQ will be disconnected in the following way.

1. If U_P and U_Q do not belong to the same row(or column) in the matrix A, then

(a) We note the four keys shared by U_P and U_Q. As has been discussed in Section 3.4.1, the keys will have the identifiers $a_{x'_1,x'_2}$, $a_{x'_1,y'_2}$, $a_{y'_1,x'_2}$ and $a_{y'_1,y'_2}$.

(b) The link PQ will not be broken if any of the five conditions is satisfied.

i. If x'_1th and x'_2th row (column) do not contain a compromised node;

ii. If x'_1th and y'_2th row (column) do not contain a compromised node;

iii. If y'_1th and x'_2th and (column) do not contain a compromised node;

iv. If y'_1th and y'_2th and (column) do not contain a compromised node;

v. If any of the nodes (x'_1, x'_2), (x'_1, y'_2), (y'_1, x'_2) and (y'_1, y'_2) is a compromised node, and there is no other compromised node belonging to that row or column.

(c) For all other conditions, the link PQ will be broken.

2. Suppose U_P and U_Q belong to the same row ($x'_1 = x'_2$) (or column ($y'_1 = y'_2$)) in the array A: Suppose at least two compromised nodes (x'_1, t_1) and (x'_1, t_2) belong to the same row as P and Q. Then all the $a_{x_1 j}$ keys ($0 \leq j < n, j \neq x_1$) are exposed. Then the link PQ will be broken if $a_{y'_1 y'_2}$ is also exposed.

We give an upper bound on the number of links broken when s nodes are compromised.

3.4.3 Calculation of upper bound for L

Let the compromised s nodes $U_1, U_2, \cdots, U_s$ belong to positions (x_1, y_1), (x_2, y_2), $\cdots$, (x_s, y_s). Let there be d distinct values among $x_1, x_2, \cdots, x_s, y_1, y_2, \cdots, y_s$. Let us denote the set of rows (or columns) containing the compromised nodes by $S_1 = \{l_1, l_2, \cdots, l_d\}$. The rest of the rows (or columns) not containing any compromised node is denoted by $S_2 = \{m_1, m_2, \cdots, m_{n-d}\}$. Clearly, $S_1 \bigcap S_2 = \phi$. Let B_d denote the number of links broken when the compromised nodes belong to d distinct rows and columns. We first note that the link between any two nodes which do not share a row (or column) as any of the compromised nodes is not broken. This is because none of the keys are exposed. To calculate the number of links broken B_d, we consider the following two cases:

1. The number of links broken, such that the nodes forming the link both do not belong to a row (or column) containing a compromised node. This number is denoted by b'_d. So,

$$b'_d = |\{\{(x'_1, y'_1), (x'_2, y'_2)\} : x'_1 \neq x'_2 \neq y'_1 \neq y'_2, \text{ at least one of } x'_1, y'_1 \in S_1 \text{ and at least one of } x'_2, y'_2 \in S_1\}|$$

2. The number of links broken, such that the nodes forming the link both belong to the same row (or column) as a compromised node. This number is denoted by b''_d. So,

$$b''_d = |\{\{(x'_1, y'_1), (x'_2, y'_2)\} : x'_1 = x'_2 \text{ or } y'_1 = y'_2 \text{ or } x'_1 = y'_2 \text{ or } x'_2 = y'_1 \text{ and at least one of } x'_1, y'_1, x'_2, y'_2 \in S_1\}|$$

So, $B_d = b'_d + b''_d$. Let p_d denote the probability that the compromised nodes belong to exactly d distinct rows (or columns). Then

$$L = \sum_{d=s}^{2s} p_d(b'_d + b''_d) \tag{3.4.1}$$

Lemma 3.4.3. *Let the compromised nodes belong to the set of columns (rows) $S_1 = \{l_1, l_2, \cdots, l_d\}$. Let $S_2 = \{m_1, m_2, \cdots, m_{n-d}\}$ be the set of the columns (rows) not containing any compromised nodes, so $S_1 \bigcap S_2 = \phi$. Then, $b'_d \leq \binom{d}{2}\binom{n-d}{2} + \frac{1}{4}\binom{d-1}{2}(d-3)d + \binom{d}{2}(d-2)(n-d)$.*

Proof. The broken links connecting the nodes not belonging to the same row (or column) share four keys. By the definition of b'_d, such a shared key a_{ij} $(i \neq j)$ will be such that either $i \in S_1$ or $j \in S_1$ or $i, j \in S_1$. We distinguish two types of keys. Type I keys:

$$\begin{array}{cccc} a_{m_1l_1}, & a_{m_1l_2}, & \cdots, & a_{m_1l_d} \\ a_{m_2l_1}, & a_{m_2l_2}, & \cdots, & a_{m_2l_d} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m_{n-d}l_1}, & a_{m_{n-d}l_2}, & \cdots, & a_{m_{n-d}l_d} \end{array}$$

and Type II keys :

$$\begin{array}{ccccc} * & a_{l_1l_2}, & a_{l_1l_3}, & \cdots, & a_{l_1l_d} \\ a_{l_2l_1}, & * & a_{l_2l_3}, & \cdots, & a_{l_2l_d} \\ \vdots & \vdots & \vdots & \vdots & \\ a_{l_dl_1}, & a_{l_dl_2}, & \cdots, & a_{l_dl_{d-1}} & * \end{array}$$

The four common keys belonging to any two links must belong to the Type I and Type II keys. Also we note that two among the four keys belong to the same row and the other two to a different row of the matrix A but in the same column as the first pair. To calculate all such combinations of Type I and Type II nodes, we consider the following cases:
Case (a) : all four keys are of Type I (one pair from one row),
Case (b) : all four keys are of Type II (one pair from one row),
Case (c) : the four keys are a combinations Type I keys (two from same row)

and Type II keys (two others from a different row). We later show this is an overestimation.

To enumerate Case (a), we choose two keys from a row of Type I keys. This can be done in $\binom{d}{2}$ ways. It has been discussed earlier that the other two keys will also belong to the same column as the first pair. The first pair can be chosen from $n-d$ rows in $n-d$ ways. Corresponding to the $n-d$ ways, the second pair can be chosen in $n-d-1$ ways. However this involves double counting. So the two pairs (four keys)can be chosen in $\binom{d}{2}\binom{n-d}{2}$ ways.

To enumerate Case (b), we choose two keys from a row in $\binom{d-1}{2}$ ways (since there are $d-1$ Type II keys in each row as shown above). Since for each of these pairs the other pair must be chosen from the remaining $d-3$ rows (we note that a pair will not occur together in two rows as shown) in $\binom{d-1}{2}(d-3)$ ways. Since there are d rows, the four keys (all of Type II) can be chosen in $\binom{d-1}{2}(d-3)d$ ways. However this sort of counting considers each of the quadruples two times. Also because of the symmetry of Matrix A, this enumeration results in double counting. For example the key $a_{l_1l_2}$ is the same as $a_{l_2l_1}$. So the number of possible combinations for Case (b) is $\frac{1}{4}\binom{d-1}{2}(d-3)d$.

We next enumerate the number of possible combinations for Case (c). We can choose the first two shared keys from a rows of Type I keys in $\binom{d}{2}$. For each of ways the second pair must be chosen $d-2$ possible rows of Type II keys in $\binom{d}{2}(d-2)$ ways. Since there are $n-d$ rows of Type I keys, the possible combinations satisfying Case (c) will be $\binom{d}{2}(d-2)(n-d)$.

A few things should be observed here.

1. The calculation above gives the correct number of links affected for Case (a).

2. We over estimate the number of links affected for Case (b). We note that a key $a_{l_xl_y}$ will not be an exposed key, if only one node is compromised in the l_x-th row and l_y-th column of A. So all links which contain shared keys of this form will remain unaffected.

3. We also over estimate the number of links for Case (c). This is also because we over estimate the number of Type II keys as discussed above.

So $b'_d \leq \binom{d}{2}\binom{n-d}{2} + \frac{1}{4}\binom{d-1}{2}(d-3)d + \binom{d}{2}(d-2)(n-d)$. ■

We now find an upper bound for b''_d.

Lemma 3.4.4. *Let the compromised nodes belong to the set of columns (or rows) $S_1 = \{l_1, l_2, \cdots, l_d\}$. Let $S_2 = \{m_1, m_2, \cdots, m_{n-d}\}$ be the set of the columns (rows) not containing any compromised nodes, so $S_1 \bigcap S_2 = \phi$. Then, $b''_d \leq d(n-d)(d-1) + \binom{d-1}{2}d$*

Proof. Let us consider the i-th column, where $i \in S_1$. We consider the two nodes at positions (j,i) and (k,i), where $i \in S_1$ and $j,k \in \{1,2,\cdots,n\}$, $j,k \neq i$. Two cases will arise here:

1. If $j \in S_2$ and $k \in S_1$, the link between the nodes (j,i) and (k,i) will be broken, if there are at least two compromised nodes belonging to the i-th column. To obtain an upper bound we relax this constraint. So in this case, the maximum number of links broken will be $(n-d)(d-1)$ (Since we do not consider the node at (i,i) in our calculation).

2. If $j,k \in S_1$, then the number of links broken will be $\binom{d-1}{2}$. This is also an over estimate, since the common key a_{jk} may not be an exposed key. This happens when we have a compromised node at (j,k) but no other compromised node in the j-th and k-th row (or column).

So, for d columns the number of broken links will be given by $b''_d \leq d(n-d)(d-1) + \binom{d-1}{2}d$ ■

From Lemma 3.4.3 and Lemma 3.4.4 and the Equation (3.4.1) we arrive at the following theorem.

Theorem 3.4.5. *Let p_d denote the probability that the nodes compromised belong to d distinct rows (or columns). The total number of links broken, will be given by* $L \leq \sum_{d=s}^{2s} p_d(\binom{d}{2}\binom{n-d}{2} + \frac{1}{4}\binom{d-1}{2}(d-3)d + \binom{d}{2}(d-2)(n-d) + d(n-d)(d-1) + \binom{d-1}{2}d)$

Exact value of B_{2s}: We discuss a special case where all the compromised nodes belong to distinct rows (columns). So, $d = 2s$. We calculate the exact value of B_{2s} under such a circumstance.

Theorem 3.4.6. $B_{2s} = s(2s-1)(n-2s)(n-2s+1)/2 + s(s-1)(2s-3)/2 + s(s-1)(s-2)(2s-5) + 2s(s-1)(n-2s) + 4s(s-1)(s-2)(n-2s) + 2s(n-2)$.

Proof. $d = 2s$. Let the s compromised nodes belong to position (l_1, l'_1), (l_2, l'_2), $\cdots$, (l_s, l'_s). We define b'_{2s} and b''_{2s} as in the first part of this section. Proceeding as in the proof of Lemma 3.4.3, we find the possible common keys that two nodes belonging to a broken link can share. We find the exact keys that can be shared and define then as Type I' and Type II' keys (Similar to Type I and Type II keys). Type I' keys:

$$\begin{array}{cccccccc}
a_{m_1 l_1}, & a_{m_1 l'_1}, & a_{m_1 l_2} & a_{m_1 l'_2} & \cdots, & a_{m_1 l_s} & a_{m_1 l'_s} \\
a_{m_2 l_1}, & a_{m_2 l'_1}, & a_{m_2 l_2} & a_{m_2 l'_2}, & \cdots, & a_{m_2 l_s} & a_{m_2 l'_s} \\
\vdots & \vdots & \vdots & \vdots & & & \\
a_{m_{n-2s} l_1}, & a_{m_{n-2s} l'_1}, & a_{m_{n-2s} l_2}, & a_{m_{n-2s} l'_2}, & \cdots, & a_{m_{n-2s} l_s}, & a_{m_{n-2s} l'_s}
\end{array}$$

and Type II' keys :

$$\begin{array}{cccccccccc}
* & * & a_{l_1 l_2}, & a_{l_1 l'_2}, & a_{l_1 l_3}, & a_{l_1 l'_3}, & \cdots, & a_{l_1 l_s}, & a_{l_1 l'_s} \\
* & * & a_{l'_1 l_2}, & a_{l'_1 l'_2}, & a_{l'_1 l_3}, & a_{l'_1 l'_3}, & \cdots, & a_{l'_1 l_s} & a_{l'_1 l'_s} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & & \\
a_{l_s l_1}, & a_{l_s l'_1}, & a_{l_s l_2}, & a_{l_s l'_2}, & a_{l_s l_3}, & a_{l_s l'_3} & \cdots & * & * \\
a_{l'_s l_1} & a_{l'_s l'_1} & a_{l'_s l_2}, & a_{l'_s l'_2}, & a_{l'_s l_3}, & a_{l'_s l'_3} & \cdots & * & *
\end{array}$$

Proceeding as in the proof of Lemma 3.4.3 number of combinations satisfying Case (a) is given by $\binom{2s}{2}\binom{n-2s}{2}$.

For Case (b) we first consider the number of ways in which the pair of nodes chosen from the same row will be the pair (l_x, l'_x). Two such pairs can be chosen from a row in $s-1$ ways. Corresponding to this, any other pair can be chosen from the other rows in $2(s-2)+1$ ways. Since there are $2s$ rows, the number of ways of selecting the four keys will be $2s(s-1)(2s-3)$. However each quadruple is counted twice, and since each shared key is also counted twice the exact number of ways of selecting the four keys will be $s(s-1)(2s-3)/2$.

When the nodes chosen are not of the form (l_x, l'_x), then the number of ways of selecting a pair will be $4\binom{s-1}{2}$ ways. The second pair can be chosen from the remaining rows in $2(s-3)+1$ ways. Since there are $2s$ rows the number of four tuples will be $2s(4\binom{s-1}{2})(2s-5)$. However as before each quadruple is calculated four times. So the number of ways of selection is $s(s-1)(s-2)(2s-5)$. Hence the number of cases which satisfy Case (b) will be given by $s(s-1)(2s-3)/2 + s(s-1)(s-2)(2s-5)$.

By a similar argument we can show that the number of cases which satisfy Case (c) will be given by $2s(s-1)(n-2s) + 4s(s-1)(s-2)(n-2s)$. Hence $b'_{2s} = s(2s-1)(n-2s)(n-2s-1)/2 + s(s-1)(2s-3)/2 + s(s-1)(s-2)(2s-5) + 2s(s-1)(n-2s) + 4s(s-1)(s-2)(n-2s)$.

b''_{2s} can be calculated in the following way. We know that no two compromised nodes belong to the same row (or column). Suppose there is a compromised node at location (l_x, l'_x). Suppose we consider the nodes at positions (i, l'_x) and (j, l'_x), $i, j \in \{1, 2, \cdots, n\}, i, j \neq l_x, l'_x$. The key $a_{l_x l'_x}$ is shared by the nodes but not exposed. So all such links are unaffected. The only links affected within the column l'_x will be between the nodes (i, l_x) and (l_x, l'_x), where $i \in \{1, 2, \cdots, n\}, i \neq l'_x, l_x$. For each of the $2s$ columns there are $n-2$ such links. Hence $b''_{2s} = 2s(n-2)$. Hence the theorem. ∎

A tighter bound for L: From Theorem 3.4.2 we find an upper bound for L. We can show that B_d is an increasing function of d. The number of ways we can choose s nodes from the whole network consisting of $N = n(n-1)/2$ nodes, is $\binom{N}{s}$. To find the number of ways of choosing s nodes such that all the rows and columns are distinct we choose $2s$ rows (or columns) to which these nodes belong in $\binom{n}{2s}$ ways. Now we form s pairs from among these $2s$ rows (or columns) in $\frac{1}{s!}\binom{2s}{2}\binom{2s-2}{2}\binom{2s-4}{2}\cdots\binom{2}{2} = \frac{(2s)!}{2^s s!}$. So the number of ways of choosing s nodes such that all the rows and columns are distinct is $\frac{n!}{(n-2s)!2^s s!}$. So $p_{2s} = \frac{n!(N-s)!}{N!(n-2s)!2^s}$. We know the exact value of B_{2s} from Theorem 3.4.6. Hence we arrive at the theorem:

Theorem 3.4.7. $L \leq \frac{n!(N-s)!}{N!(n-2s)!2^s} B_{2s} + (1 - \frac{n!(N-s)!}{N!(n-2s)!2^s}) B_{2s-1}$

The Table 3.3 represent the experimental values of $E(s)$. The values of $E(s)$ have been obtained by compromising s nodes randomly and the experiment is conducted for 100 runs. The theoretical upper bound are also given.

n	N	k	s	$E(s)$ Experimental	$E(s)$ Upper Bound
30	435	56	10	0.3500	0.63
40	780	76	10	0.2510	0.3900
50	1225	96	10	0.1800	0.2417
60	1770	116	10	0.1314	0.1740
70	2415	136	10	0.07241	0.1382

Table 3.3: Experimental value of $E(s)$ and theoretical upper bound for $E(s)$, when number of nodes is $N = n(n-1)/2$ and keys per node is k

3.5 A second design: introducing new sensor nodes

In the previous design we saw that the size of the key pool is the same as the size of the network. Suppose we want to increase the size of the network, without adding new keys in the key pool. The design presented in this section augments the size of the network, keeping the same number of keys in each node. The keys in the key pool also remain the same. We show that network size can be increased in steps, keeping the same number of keys per node. However to ensure that any pair of nodes can communicate directly, we cannot go on adding nodes. We calculate the values of $E(s)$ and $V(s)$ when we double the size of the network. This design also uses triangular PBIBD. The main idea is to add new blocks to the existing PBIBD, thus increasing the number of sensor nodes.

3.5.1 Construction of the network

We use triangular PBIBD designs in the key predistribution scheme. We consider two triangular PBIBD schemes [36, Page 18] with parameters

$$v', r', k', b', n_i', \lambda_i', (p_{jk}^i)'$$

and

$$v'', r'', k'', b'', n_i'', \lambda_i'', (p_{jk}^i)'' \ (i, j, k = 1, 2)$$

and having two identical association schemes. Then we can construct another triangular PBIBD having the parameters

$$v = v' = v'', r = r' + r'', k = k' = k'', b = b' + b'',$$
$$\lambda_1 = \lambda_1' + \lambda_1'', \lambda_2 = \lambda_2' + \lambda_2''.$$

We consider two PBIBD schemes both having the same parameters:

$$v' = n(n-1)/2, b = n(n-1)/2, r' = 2(n-2), k' = 2(n-2),$$
$$\lambda'_1 = n-2, \lambda'_2 = 4,\ n \geq 5.$$

The first association scheme results from the array A as given in Section 3.2, whereas the second PBIBD has the association scheme resulting from the array

$$A' = \begin{array}{cccccc} * & 1 & n & \cdots & n(n-1)/2-2 & n(n-1)/2 \\ 1 & * & 2 & \cdots & \cdots & n(n-1)/2-1 \\ n & 2 & * & 3 & \cdots & n(n-1)/2-3 \\ \vdots & \vdots & \vdots & \vdots & & \\ \vdots & \vdots & \vdots & \vdots & * & n-1 \\ n(n-1)/2 & n(n-1)/2-1 & \cdots & \cdots & n-1 & * \end{array}$$

We can then construct a PBIBD having $n(n-1)$ blocks each containing $2(n-2)$ elements. When we map this to a sensor network we arrive at a network having $n(n-1)$ sensor nodes. As an example we consider a network having 20 nodes. Apart from the ten blocks (here sensor nodes) constructed in Section 3.2, we add the following ten nodes having the following keys: $\{2,5,6,9,8,10\}$, $\{1,3,5,6,7,9\}$, $\{2,4,5,6,7,8\}$, $\{3,6,7,8,9,10\}$, $\{1,2,3,7,8,10\}$, $\{1,2,3,4,8,9\}$, $\{2,3,4,5,9,10\}$, $\{1,3,4,5,6,10\}$, $\{1,2,4,6,7,10\}$, $\{1,4,5,7,8,9\}$. Each of the nodes contains $2(n-2)$ keys. Let $f'(x,y)$ be the entry in location (x,y) of the array A'. Then $f'(x,y)$ is given by

$$f'(x,y) = \begin{cases} *, \text{ for } x = y \\ (x-y-1)(2n-x+y)/2 + y, \text{ for } x > y \\ (y-x-1)(2n-y+x)/2 + x \text{ otherwise} \end{cases}$$

Key predistribution is similar to the first scheme.

3.5.2 Algorithm to find common key

Let the nodes U_i and U_j want to communicate with each other. Any node U_j broadcasts the following information.

1. Array s from which U_j was derived. $m[s] = 0$ if U_j is derived from A and $m[s] = 1$ if U_j is derived from A'. This requires one bit.

2. Position (x_j, y_j) of U_j in the array from which it has been derived. This requires $O(log\ \sqrt{N})$.

Given the above information node U_i can calculate the common keys using Algorithm 2.

3.5.2.1 Proof of Correctness of Algorithm 2

If both the nodes U_i and U_j are derived from the same array, then we follow the algorithm similar to that given in Section 3.3. We will consider the case when U_i and U_j are derived from arrays A' and A respectively. The case where U_i and U_j are derived from arrays A and A' respectively will follow similarly.

We consider the following example

Algorithm 2 Shared key Discovery for Scheme II

if $m[i] = m[j] = 0$ **then**
  if $x_i = x_j$ **then**
    Ids of the common keys are $a_{f(x_i,t)}$ and $a_{f(y_i,y_j)}$, for $t = 1, 2, \cdots, n$ and $t \neq x_i, y_i, y_j$.
  else if $y_i = y_j$ **then**
    Ids of the common keys are $a_{f(t,y_i)}$ and $a_{f(x_i,x_j)}$, for $t = 1, 2, \cdots, n$ and $t \neq x_i, y_i, x_j$.
  else
    Ids of the common keys $a_{f(x_i,x_j)}$, $a_{f(x_i,y_j)}$, $a_{f(y_i,x_j)}$ and $a_{f(y_i,y_j)}$.
  end if
else if $m[i] = m[j] = 1$ **then**
  if $x_i = x_j$ **then**
    Ids of the common keys are $a_{f'(x_i,t)}$ and $a_{f'(y_i,y_j)}$, for $t = 1, 2, \cdots, n$ and $t \neq x_i, y_i, y_j$.
  else if $y_i = y_j$ **then**
    Ids of the common keys are $a_{f'(t,y_i)}$ and $a_{f'(x_i,x_j)}$, for $t = 1, 2, \cdots, n$ and $t \neq x_i, y_i, x_j$.
  else
    Ids of the common keys $a_{f'(x_i,x_j)}$, $a_{f'(x_i,y_j)}$, $a_{f'(y_i,x_j)}$ and $a_{f'(y_i,y_j)}$.
  end if
else if $m[i] = 1$ and $m[j] = 0$ **then**
  Ids of the common keys as calculated by i will be $a_{f'(a,b)}$ where

  1. $(a, b) = (y_i - x_j, y_i), (y_i - y_j, y_i), (y_i + x_j, y_i), (y_i + y_j, y_i), (x_i, x_i - x_j), (x_i, x_i - y_j), (x_i, x_i + x_j), (x_i, x_i + y_j)$, such that $0 < a, b \leq n$ and $a \neq b$.
  2. $(a, b) = (x_i, x_j), (x_i, y_j)$ if $a < b$, $x_i \neq x_j$
  3. $(a, b) = (x_j, y_i), (y_i, y_i)$ if $a > b$, $y_i \neq y_i$
  4. $(x_i, 1), (x_i, 2), \cdots, (x_i, x_{i-1})$ if $x_j = x_i$.
  5. $(1, y_i), (2, y_i), \cdots, (y_{i-1}, y_i)$ if $y_j = y_i$.

else
  Ids of the common keys will be calculated as above except that the $f_{(a,b)}$ will be calculated instead of $f'_{(a,b)}$.
end if

Example 3.5.1. *Suppose position of* $U_i = (5, 7)$ *in array* A' *and that of* $U_j = (4, 6)$ *in array* A. *Ids of keys belonging to* U_j *are* $3, 9, 14, 19, 21, 22$ *and* $5, 11, 16, 23, 26, 27$. *Arrays* A *and* A' *have been shown in Figures 3.1(a) and 3.1(b) respectively.*

*	1	2	3	4	5	6	7
1	*	8	9	10	11	12	13
2	8	*	14	15	16	17	18
3	9	14	*	19	20	21	22
4	10	15	19	*	23	24	25
5	11	16	20	23	*	26	27
6	12	17	21	24	26	*	28
7	13	18	22	25	27	28	*

(a) Array A

*	1	8	14	19	23	26	28
1	*	2	9	15	20	24	27
8	2	*	3	10	16	21	25
14	9	3	*	4	11	17	22
19	15	10	4	*	5	12	18
23	20	16	11	5	*	6	13
26	24	21	17	12	6	*	7
28	27	15	22	18	13	7	*

(b) Array A'

Figure 3.1: Array A and Array A'

We mark four diagonal lines and two vertical lines and two horizontal lines starting from 5th and 7th column and 5th and 7th rows. We find all the crossed elements that lie in the 7*th column. These are the elements common between* U_i *and* U_j *which occur along the two diagonals. These are* 26 *and* 21. *Similarly, all the crossed elements that lie in the* 5*th row are the common elements between* U_i *and* U_j. *These are* 19 *and* 5 *in the above example. So the common keys have identifiers* 5, 19, 21 *and* 26.

Proceeding as in the example above there will be at most four diagonal lines and two vertical lines and two horizontal lines. The position of the elements along the marked diagonals that lie on the same column as U_i will be given by, $(a, b) = (y_i - x_j, y_i), (y_i - y_j, y_i), (y_i + x_j, y_i), (y_i + y_j, y_i)$ such that $0 < a, b \leq n$ and $a \neq b$. Similarly, all the positions of the elements along the marked diagonals that lie on the same row as U_i and is given by $(a, b) = (x_i, x_i - x_j), (x_i, x_i - y_j), (x_i, x_i + x_j), (x_i, x_i + y_j)$, such that $0 < a, b \leq n$ and $a \neq b$.

If both U_i and U_j belong to the same row (ie, $x_i = x_j$), then the position of the common elements will be $(a, b) = (x_i, 1), (x_i, 2), \cdots, (x_i, x_{i-1})$. These elements lie on one of the marked rows. If both i and j belong to the same column (ie, $y_i = y_j$), then the position of the common elements will be $(a, b) = (1, y_i), (2, y_i), \cdots, (y_{i-1}, y_i)$. These elements lie on one of the marked columns. If i and j do not belong to the same row or column, then the positions will be given by $(a, b) = (x_i, x_j), (x_i, y_j)$ if $a < b$ and $(a, b) = (x_j, y_i), (y_i, y_i)$ if $a > b$. So the ids of the common keys are given by $f'_{(a,b)}$. To communicate, the nodes can choose any of the common keys.

3.5.2.2 Time complexity of Algorithm 2

All the steps take $O(1)$ to be done. Hence the overall time complexity is $O(1)$. Each node broadcasts the value $m(s)$, the information of the array to which it

belongs (this requires just one bit) and its position in the array from which it is derived. Since the order of each array is $O(\sqrt{N})$ (where N is the number of nodes) $O(log\ \sqrt{N})$ bits have to be broadcasted.

3.5.3 Resiliency of Scheme II

n	N	k	s	$V(s)(Experimental)$
20	380	36	7	0.0400
30	870	56	10	0.0313
40	1560	76	10	0.0116
50	2450	96	10	0.0066
60	3540	116	10	0.0031
70	4830	136	10	0.0019

Table 3.4: Experimental value of $V(s)$, when number of nodes is $N = n(n-1)$ and keys per node is k

n	N	k	s	$E(s)(Experimental)$
30	870	56	8	0.2043
40	1560	76	10	0.1855
50	2450	96	10	0.1174
60	3540	116	10	0.08942
70	4830	136	10	0.06742

Table 3.5: Experimental value of $E(s)$, when number of nodes is $N = n(n-1)$ and keys per node is k

We study the effect of node compromise on such a network. We give the experimental results for $V(s)$ and $E(s)$ for this network in Table 3.4 and Table 3.5 respectively. The experiments are performed 100 times with s nodes being randomly compromised each time. We observe that for networks of comparable sizes, the design given in Section 3.4 gives better results in terms of $E(s)$, however at the cost of more number of keys per node. This design also results in a network in which every pair of nodes share at least one common key. Thus all pairs of nodes can communicate directly, making communication faster. We can also add more nodes in the network in a recursive manner.

Schemes	N	k	p_c
Basic Scheme [55]	2415	136	0.99
Çamtepe-Yener [18, 20]	2257	48	1
Linear Scheme [86]	2209	30	0.64
Quadratic Scheme [89]	2197	12	0.36
Scheme I	2415	136	1
Scheme II	2450	96	1

Table 3.6: Comparing the connectivity ratio p_c of our schemes with the Basic scheme, Çamtepe-Yener Scheme, Lee-Stinson Linear and Quadratic schemes

3.6 Comparison with existing schemes

Our design performs better than the basic scheme [55] for identical number of keys per node and the same size of the network. We compare the resiliency of our scheme with different schemes like the basic scheme [55], Çamtepe-Yener Scheme [18, 20], Lee-Stinson linear scheme [86] and quadratic scheme [89]. The parameters of the schemes are given in Table 3.6. From the table we see that our scheme always results in full connectivity of the network. Though the basic scheme has very high resiliency, there is no guarantee that all nodes are directly connected. The graph (Figure 3.2) compares our scheme with the above schemes. We see that our schemes give better resiliency compared to other schemes. However this comes at the cost of a large key ring size.

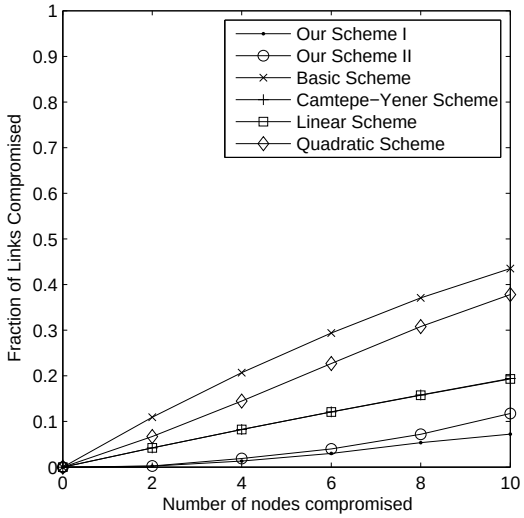


Figure 3.2: Comparing resiliency of our schemes, (Scheme I, Scheme II), Basic scheme, Çamtepe-Yener Scheme, Lee-Stinson Linear and Quadratic schemes

3.7 Concluding remarks

This work has been modified by Das and Roy [39]. They scaled the network to four times the original size. In the next chapter we discuss a key predistribution scheme using codes, with particular reference to Reed-Solomon codes. The number of keys is smaller than the schemes in this chapter, however the resiliency is poorer.

Chapter 4

Key predistribution using Codes

In this chapter we propose novel deterministic key predistribution schemes using codes. The motivation behind this is that there is a close resemblance between designs and codes. One can be obtained from the other. Our construction is generic and can be applied to any code, in specific we study the effect of predistribution using Reed-Solomon codes. Our scheme is the first proposed key predistribution scheme in a key pool based setting. We show that the scheme is resilient to selective node capture attack because there is no clever way of knowing which nodes to compromise. The scheme is also resilient to node fabrication attack. An important advantage of our scheme is that it permits the scalability of the network without redistributing keys in the existing nodes.

As we have already discussed in Chapter 2 that Eschenauer and Gligor [55] proposed a random key predistribution scheme in which keys are randomly drawn from a key pool and placed in the sensors prior to deployment. Such schemes are random key pool based schemes. Other key pool based schemes were proposed by Chan, Perrig and Song [28] and Liu and Ning [91]. In another approach dynamic keys are generated from a given matrix. This approach was first proposed by Blom [9]. The idea was extended by Du et al [49].

Whereas in the schemes proposed by Eschenauer and Gligor [55], Chan, Perrig and Song [28] the keys are selected randomly from a key pool and placed in sensors, in the combinatorial design based schemes there is a pool of key identifiers from which sets of identifiers are assigned to each sensor. We call both the types of schemes the key pool based schemes.

According to our technique, codewords are mapped to the sensor nodes and (x, i) is mapped to the keys, where x is the i-th symbol of a codeword. If we consider a code of length $k = q - 1$, dimension dim with alphabet in Q (such that $|Q| = q$), then the minimum distance is given by $k - dim + 1$. The number of codewords is q^{dim}. For the i-th codeword $(a_1, a_2, \cdots, a_k)$ we assign the keys (a_j, j) $(j = 1, 2, \cdots, k)$ to the i-th sensor. We map this design to sensor networks. So the network consists of q^{dim} sensors, each having $k = q - 1$ keys. We consider the problem of node compromise and calculate the resiliency of the network as the

fraction of links broken and nodes disconnected. We present experimental results for these parameters and support them by calculating the upper bounds. We show that our scheme is better than Lee and Stinson's scheme [86] that uses Transversal Designs. The greatest advantage of our scheme is that new nodes can be added to the network without redistributing the keys in the existing nodes. The earlier schemes using combinatorial designs were not scalable, except the one discussed in the previous chapter, where the network could be scaled to twice its original size. According to our scheme the system can be scaled to more than $q-1$ times the size of the existing network.

In [3] Al-Shurman and Yoo presented a key management system based on codes. According to their scheme a publicly known MDS (Maximum distance separable) generator matrix is available to every node. Every node in the network uses this matrix and a random vector to generate codeword which is the secret key chain. The design satisfies the Cover-free-family (CFF) properties with certain probability. They did not address the problem of connectivity in a network and did not consider the scenario where nodes are compromised. We not only present a novel predistribution scheme based on codes but also study the connectivity of the network. We make a detailed study of the resiliency of the system under node compromise and establish upper bounds for the same. Further, our system is scalable. According to Al-Shurman and Yoo's scheme, a codeword is a key chain itself where as in our scheme the codewords are used to derive the identifiers of the keys. So our basic setting, techniques and results are completely different from that presented by Al-Shurman and Yoo [3].

Rest of the chapter is organized as follows. In Section 4.1 we define a few terms and concepts. In Section 4.2 we describe our technique to map codes to key predistribution. Section 4.3 presents a method of finding the shared keys between nodes. We present the analysis on connectivity and resiliency in Sections 4.4 and 4.5 respectively. In Section 4.6 we show how to make the system scalable. We compare our scheme with existing schemes in Section 4.7. Section 4.8 shows that according to our construction Orthogonal Arrays can be used instead of codes to design predistribution schemes. We conclude with some open problems in Section 4.9. This chapter is the outcome of the paper [137].

4.1 Reed-Solomon codes

We recollect the definition of codes 1.5.17 from 1.5.

We use Reed Solomon codes. If k be the length of the code and dim be the dimension of the code, then the distance of the Reed Solomon codes is $k-dim+1$. The number of codewords $N = q^{dim}$, where q is a prime power. The values of k, dim, d, q must satisfy the inequality $1 \leq dim < k < q$. The construction of these codes can be found in [163].

4.2 Construction of key predistribution schemes from codes

We consider (k, N, d, q)-code having length k, distance d and number of codewords equal to N. Let Q be the set of symbols. We consider the key pool to be consisting of keys $\{(i, j) : i \in Q, j = 1, 2 \cdots, k\}$. The size of the key pool is qk. We map each codeword to a sensor node. Suppose the i-th codeword be $(a_1^{(i)}, a_2^{(i)}, \cdots, a_k^{(i)})$. We assign the key identifiers $(a_j^{(i)}, j)$ for $j = 1, 2, \cdots, k$ to the i-th sensor. So each sensor consists of k keys. Suppose d be the distance between two codewords x and y, then the number of common keys between two sensor nodes U_x and U_y is $k - d$. Thus the network can support a maximum of N sensor nodes, each having k keys. The number of common keys between any two nodes is less than $k - d$. Since sensor networks consist of a large number of resource constrained nodes, our aim is to find codes which have large N and small k. It can be observed that a large key pool increases the resiliency of the network. It is to be noted that if nodes share a common key then communication is efficient and faster. However if nodes have too many shared keys then the compromise of nodes will render a large number of keys ineffective, thus adversely affecting the resiliency of the network. We look for codes which have large N, small k and large d. In this paper we use Reed Solomon codes for the construction of key predistribution schemes. We also discuss how the above construction can be used to construct key predistribution scheme using Orthogonal Arrays in Section 4.8.

4.2.1 Construction of key predistribution scheme from Reed Solomon codes

We consider a (k, q^{dim}, d, q) Reed Solomon code having alphabet in the finite field F_q $(q > 2)$. The length of the code is $k = q - 1$, distance is $d = k - dim + 1$ and dimension is dim. The number of codewords is $N = q^{dim}$. When we map this code to sensor network we get a sensor network consisting of q^{dim} nodes, each having $q - 1$ keys. The number of keys common between any two nodes is at the maximum $k - d = dim - 1$. For any codeword $x = (a_1, a_2, \cdots, a_k)$, we assign the keys $(a_1, 1)$, $(a_2, 2)$, $\cdots$, (a_k, k) to the node x. The key pool consists of qk keys $\{(a_i, i) : a_i \in F_q, i = 1, 2, \cdots, k\}$.

The construction of Reed Solomon code has been presented in [163]. For completeness we present it here. Let F_q be a finite field of $q > 2$ elements. Let $\mathcal{P}$ be the set of polynomials over F_q of degree at most $dim - 1$. Thus $|\mathcal{P}| = q^{dim}$. Let $F_q^* = \{\alpha_1, \alpha_2, \cdots, \alpha_{q-1}\}$ be the set of non-zero elements of F_q. For each polynomial $p_i(x) \in \mathcal{P}$, we define $c_{p_i} = (p_i(\alpha_1), p_i(\alpha_2), \cdots, p_i(\alpha_{q-1}))$ to be the i-th codeword of length $q - 1$. We define $\mathcal{C} = \{c_{p_i} : p_i(x) \in \mathcal{P}\}$. So $\mathcal{C}$ is a Reed Solomon Code.

Example 4.2.1. *Let us consider a Reed Solomon code having parameters $q = 4$, $k = 3$, $dim = 2$. Let the codewords be*

$$\begin{pmatrix}
0 & 0 & 0 \\
1 & 1 & 1 \\
2 & 2 & 2 \\
3 & 3 & 3 \\
1 & 2 & 3 \\
0 & 3 & 2 \\
3 & 0 & 1 \\
2 & 1 & 0 \\
2 & 3 & 1 \\
3 & 2 & 0 \\
0 & 1 & 3 \\
1 & 0 & 2 \\
3 & 1 & 2 \\
2 & 0 & 3 \\
1 & 3 & 0 \\
0 & 2 & 1
\end{pmatrix}$$

This results in a network having $q^2 = 16$ nodes, each having 3 keys. Thus $d = 2$. The node ids, polynomials corresponding to the nodes and keys belonging to the nodes are given in Table 4.1.

Any two nodes share a maximum of one key.

4.3 Key establishment between two nodes

Suppose nodes U_i and U_j want to communicate with each other. Then the nodes broadcast their identifiers. The identifier i is given by $a_0^{(i)} + a_1^{(i)}q + a_2^{(i)}q^2 + \cdots + a_{dim-1}^{(i)}q^{dim-1}$, where $a_j^{(i)} \in F_q$ for all $j = 0, 1, 2, \cdots, dim - 1$. Then the polynomial corresponding to the i-th codeword is $p_i(x) = a_0^{(i)} + a_1^{(i)}x + a_2^{(i)}x^2 + \cdots + a_{dim-1}^{(i)}x^{dim-1}$. For practical purposes dim is chosen to be small and can be regarded as a constant. Each node can find the polynomial corresponding to its identifier. Since dim is constant for the code, this can be done in constant time. Let the polynomials corresponding to the nodes U_i and U_j be represented by $p_i(x) = a_0^i + a_1^i x + a_2^i x^2 + \cdots + a_{dim-1}^i x^{dim-1}$ and $p_j(x) = a_0^j + a_1^j x + a_2^j x^2 + \cdots + a_{dim-1}^j x^{dim-1}$ respectively. Let (a, t) be the common key. Then

$$a_0^{(i)} + a_1^{(i)}x_t + a_2^{(i)}x_t^2 + \cdots + a_{dim-1}^{(i)}x_t^{dim-1} = a_0^{(j)} + a_1^{(j)}x_t + a_2^{(j)}x_t^2 + \cdots + a_{dim-1}^{(j)}x_t^{dim-1}$$

From the above equation we can find the value of x_t if it exists. Then the common key will be $(p_i(x_t), t)$. For $dim = 2, 3, 4$ this can be done in constant time, since there are known formulae for solving quadratic, cubic and quartic equations. As

Block id (i)	Polynomial ($p_i(x)$)	Keys
0	0	$\{(0,1),(0,2),(0,3)\}$
1	1	$\{(1,1),(1,2),(1,3)\}$
2	2	$\{(2,1),(2,2),(2,3)\}$
3	3	$\{(3,1),(3,2),(3,3)\}$
4	x	$\{(1,1),(2,2),(3,3)\}$
5	$1+x$	$\{(0,1),(3,2),(2,3)\}$
6	$2+x$	$\{(3,1),(0,2),(1,3)\}$
7	$3+x$	$\{(2,1),(1,2),(0,3)\}$
8	$2x$	$\{(2,1),(3,2),(1,3)\}$
9	$1+2x$	$\{(3,1),(2,2),(0,3)\}$
10	$2+2x$	$\{(0,1),(1,2),(3,3)\}$
11	$3+2x$	$\{(1,1),(0,2),(2,3)\}$
12	$3x$	$\{(3,1),(1,2),(2,3)\}$
13	$1+3x$	$\{(2,1),(0,2),(3,3)\}$
14	$2+3x$	$\{(1,1),(3,2),(0,3)\}$
15	$3+3x$	$\{(0,1),(2,2),(1,3)\}$

Table 4.1: Block, corresponding polynomial and keys

highlighted earlier in this section that large *dim* adversely affects the resiliency, so we can keep the value of $dim = 2, 3, 4$ for all practical purposes. Thus shared key discovery can be done in constant time. If the equations do not have a solution, then there is no common key between the nodes and a path needs to be established as given in Section 1.2.1.4. It is to be noted that any two nodes is connected by a maximum of two hop path.

4.4 Connectivity of the network

As defined in Section 1.2.4 the *Connection probability* or *Connectivity* p_c of the network to be the probability that two nodes are connected by one or more keys. The theorem below gives the value of connectivity.

Theorem 4.4.1. $p_c = \frac{\sum_{i=1}^{dim-1}(-1)^{i-1}q^i\binom{q^{dim-i}}{2}\binom{q-1}{i}}{\binom{q^{dim}}{2}}$

Proof. We know that the total number of nodes in a network is q^{dim} each containing $q-1$ keys. So $\binom{q^{dim}}{2}$ links are possible. We note that

$$p_c = \frac{\text{Number of links present in the network}}{\binom{q^{dim}}{2}}.$$

We now calculate the number of links that are actually present in the network. When $dim = 2$, any two nodes share a maximum of one key. The number of nodes

in which the key $(a_i, 1)$ $(i \in GF(q))$ occurs is $q^{dim-1} = q$. Since a maximum of one key is shared between any two nodes, the number of links connected by the key $(a_i, 1)$ is $\binom{q}{2}$. Similarly the number of links connected by the key $(a_i, 2)$ is $\binom{q}{2}$. So the total number of keys connected by the keys (a_i, j) $(j = 1, 2, \cdots, k)$ is $k\binom{q}{2}$. So the number of links connected before compromise is $kq\binom{q}{2} = q^2(q-1)^2/2$.

When $dim = 3$, any two nodes share a maximum of two keys. Proceeding as above we see that the number of nodes connected by any key (a_i, i) is $\binom{q^{dim-1}}{2}$. So the total number of nodes connected by common keys is $\binom{q^{dim-1}}{2}q(q-1)$. However since nodes may share two keys some nodes may be connected by a pair of keys. This results in double counting. So we subtract the number of links connected by two keys. The number of times a given pair occurs is $q^{dim-2} = q$. The number of such pairs is $q^2\binom{q-1}{2}$. Hence the number of links connected by two keys is $\binom{q^{dim-2}}{2}q^2\binom{q-1}{2}$. So the number of links connected before compromise is $\binom{q^{dim-1}}{2}q(q-1) - \binom{q^{dim-2}}{2}q^2\binom{q-1}{2}$.

Proceeding similarly for any arbitrary dim, the total number of links in the network is given by
$\binom{q^{dim-1}}{2}q(q-1) - \binom{q^{dim-2}}{2}q^2\binom{q-1}{2} + \binom{q^{dim-3}}{2}q^3\binom{q-1}{3} - \cdots + (-1)^{dim-2}\binom{q}{2}q^{dim-1}\binom{q-1}{dim-1}$.

Hence the theorem. ■

For example we consider a network where $q = 16$, $dim = 3$. The number of nodes in the network is $q^{dim} = 4096$. Each node contains 15 keys. The connectivity of the network is $p_c \simeq 0.6$.

4.5 Resiliency of the network

We have discussed about two types of node compromise in Section 1.2.3. We show that our scheme is secure against selective node capture attack and node fabrication attack.

4.5.1 Resiliency against selective node capture

During selective node capture the attacker compromises those nodes whose keys have not already been compromised. We note that any two nodes broadcast their node ids during the shared-key discovery phase. The key identifiers are not broadcasted. Thus at no stage the attacker can know what key identifier is present in which node. Thus there is no way of knowing which nodes are left to be compromised. Thus unless the attacker compromises the node, she cannot choose a node for compromise to maximize the number of keys compromised. Hence our scheme is secure against selective node capture.

4.5.2 Resiliency against node fabrication attack

In node fabrication attack, nodes may fabricate sensors with new identities. However since in our scheme every node has a distinct node identifier, malicious nodes cannot fabricate the old nodes with their identities. Thus our scheme is resilient to node fabrication attack.

4.5.3 Resiliency against random node capture

Mathematically, $V(s)$ is given by

$$V(s) = \frac{\text{Number of nodes disconnected when } s \text{ nodes are compromised}}{q^{dim}}$$

We refer to the Example 4.2.1. If the nodes having identifiers 0,1,2, and 3 are compromised then all the nodes are disconnected. This is because all the keys are exposed. By a similar argument we can see that if the nodes having identifier values $0, 1, 2, \cdots, q-1$ are compromised then all the nodes are disconnected. So a minimum of q nodes have to be disconnected to compromise the entire network.

We consider a node having the keys $\{(a_1, 1), (a_2, 2), \cdots, (a_k, k)\}$. This node is disconnected if the nodes containing the keys $(a_1, 1), (a_2, 2), \cdots, (a_k, k)$ are compromised. The maximum intersection between two nodes is $k - d = dim - 1$. So the minimum number of nodes to be compromised to disconnect a node is $\lceil k/(dim-1) \rceil$. This happens when $dim - 1$ keys are exposed between the disconnected node and each of the compromised node.

To break the entire network i.e. to disconnect all the nodes, all the keys must be exposed. The minimum number of nodes to be disconnected to expose all the keys is q. In this condition the nodes containing the keys $\{(0, 1), (0, 2), \cdots, (0, k)\}$, $\{(1, 1), (1, 2), \cdots, (1, k)\}$, $\cdots$, $\{(q-1, 1), (q-1, 2), \cdots, (q-1, k)\}$ are exposed.

We present experimental values of $V(s)$ in the Table 4.2. The experiment is performed 100 times, each time s nodes are compromised randomly. From the table we note that even when a large number of nodes are compromised, very few nodes are disconnected. Hence our system has high resiliency in terms of node disconnection.

4.5.4 Analysis of $E(s)$

We present experimental values for $E(s)$ and calculate the upper bounds for the same.

Upper bound for $E(s)$ The value of $E(s)$ depends on which s nodes are compromised. Accordingly some keys are exposed. When nodes are compromised randomly it is difficult to predict which set of s nodes are compromised. So we do not know which keys are exposed. For example if the nodes with id 0 and 1 are compromised, then the keys $(0, 1), (0, 2), (0, 3)$ $(1, 1), (1, 2), (1, 3)$ are exposed, where as if the nodes 5 and 6 are compromised then the keys

q	dim	N	k	s	$V(s)(Experimental)$
49	2	2041	48	100	0.0003867
49	2	2041	48	150	0.0981
71	2	5041	70	150	0.00283
71	2	5041	70	200	0.0762
101	2	10201	100	200	0.01139
101	2	10201	100	250	0.0779

Table 4.2: Experimental value of $V(s)$, when number of nodes is $N = q^{dim}$ and keys per node is k

$(0,1),(3,2),(2,3),(3,1),(0,2),(1,3)$ exposed. Hence the exact calculation of $E(s)$ is not possible. So we calculate the upper bound for $E(s)$. It is clear that the maximum number of links will be broken when keys exposed are all distinct. In this section we establish the theoretical upper bound for $E(s)$. We state the following theorem.

Theorem 4.5.1. *If s nodes are randomly compromised, then*

$$E(s) \leq 1 - \frac{\sum_{i=1}^{dim-1}(-1)^{i-1}(q-s)^i\binom{q^{dim-i}}{2}\binom{q-1}{i}}{\sum_{i=1}^{dim-1}(-1)^{i-1}q^i\binom{q^{dim-i}}{2}\binom{q-1}{i}}$$

Proof. Suppose s nodes are compromised. Let the compromised nodes be denoted by U_1, U_2, $\cdots$, U_s. Let the exposed keys be denoted by $(a_1^{U_1},1)$, $(a_2^{U_1},2)$, $\cdots$, $(a_k^{U_1},k)$, $(a_1^{U_2},1)$, $(a_2^{U_2},2)$, $\cdots$, $(a_k^{U_2},k)$, $\cdots$, $(a_1^{U_s},1)$, $(a_2^{U_s},2)$, $\cdots$, $(a_k^{U_s},k)$. It is clear that maximum nodes will be broken when all the exposed keys are distinct. So while calculating the upper bound of $E(s)$, we consider the number of links disconnected when all the exposed keys are distinct.

When $dim = 2$, any two nodes share a maximum of one key. The number of links before compromise $kq\binom{q}{2} = q^2(q-1)^2/2$ and has been calculated in Theorem 4.4.1 Suppose $(a_i^{U_j},1)$ be a compromised key. All the links that are connected by this key are broken. Since there are sk such exposed keys (since all the exposed keys are distinct), the number of links broken is $sk\binom{q}{2} = sq(q-1)^2/2$.

When $dim = 3$, any two nodes share a maximum of two keys. The number of links connected before compromise as calculated in Theorem 4.4.1 is $\binom{q^{dim-1}}{2}q(q-1) - \binom{q^{dim-2}}{2}q^2\binom{q-1}{2}$.

Now we calculate the number of links which remain connected when s nodes are compromised. The number of such links is given by $\binom{q^{dim-1}}{2}(q-s)(q-1)$. However this involves some links which are connected by two uncompromised keys. The number of such links is $\binom{q^{dim-2}}{2}(q-s)^2\binom{q-1}{2}$. So this number must be deducted from the total count. Thus the minimum number of links connected when s nodes are compromised is $\binom{q^{dim-1}}{2}(q-s)(q-1) - \binom{q^{dim-2}}{2}(q-s)^2\binom{q-1}{2}$.

Proceeding similarly for any arbitrary dim, the total number of links before compromise is given by $\binom{q^{dim-1}}{2}q(q-1) - \binom{q^{dim-2}}{2}q^2\binom{q-1}{2} + \binom{q^{dim-3}}{2}q^3\binom{q-1}{3} - \cdots + (-1)^{dim-2}\binom{q}{2}q^{dim-1}\binom{q-1}{dim-1}$. The number of links connected after s nodes are compromised is $\binom{q^{dim-1}}{2}(q-s)(q-1) - \binom{q^{dim-2}}{2}(q-s)^2\binom{q-1}{2} + \binom{q^{dim-3}}{2}(q-s)^3\binom{q-1}{3} - \cdots + (-1)^{dim-2}\binom{q}{2}(q-s)^{dim-1}\binom{q-1}{dim-1}$.

From this the result follows. ■

Table 4.3 represents the experimental results for $E(s)$. The experiment is conducted 100 times, each time s nodes are compromised randomly. The corresponding upper bounds are also shown.

q	dim	N	k	s	$E(s)(Experimental)$	Theoretical Upper Bound
49	2	2041	48	10	0.186564	0.2041
49	2	2041	48	15	0.2761	0.3061
71	2	5041	70	10	0.13154	0.1408
71	2	5041	70	20	0.2474	0.2817
101	2	10201	100	10	0.01139	0.0990
101	2	10201	100	30	0.2582	0.2970

Table 4.3: Experimental value of $E(s)$, when number of nodes is $N = q^{dim}$ and keys per node is k

4.6 Scalability of the network

The most important advantage of our scheme is that the network can be made scalable. This means we can introduce more nodes if need arises. However the keys in the already existing nodes need not be changed or redistributed. Suppose there are q^{dim} nodes in the network. If we increase the value of dim by 1, then there can be q^{dim+1} codewords. So we can introduce a maximum of $q^{dim+1} - q^{dim}$ nodes in the network without redistributing the keys by increasing the value of dim by 1. If we increase the value of dim by more than one, we can introduce more nodes. This is possible because q^{dim} polynomials corresponding to the earlier nodes remain the same. We note that a codeword is basically the evaluation of a polynomial of degree at most $dim - 1$ at k points. So whatever be the value of dim, the length of the codeword is $k = q - 1$. So the number of keys stored in each sensor is $q - 1$, whatever be the value of dim. Initially there are q^{dim} polynomials with a degree at most $dim - 1$. We can introduce $q^{dim+1} - q^{dim}$ polynomials with degree dim. The evaluation of these $q^{dim+1} - q^{dim}$ polynomials at k points will give rise to codewords from which we can derive the keys as described in Section 4.2.

For example let us assume that $q = 16$, $dim = 2$. So there are $q^2 = 256$ nodes in the network. The number of keys per node is $q - 1 = 15$. Corresponding to each node is a polynomial of degree at most $dim - 1 = 1$ with coefficients in $GF(16)$. Now we consider polynomials of degree 2 with coefficients in $GF(16)$. There are $q^3 - q^2 = 3840$ such polynomials. So we can add 3840 nodes each having 15 keys without changing the keys in the previous 256 nodes. Thus we can scale the network to 15 times is original size.

However one disadvantage will be that the resiliency will decrease on increasing k.

4.7 Comparison with existing schemes

We compare our scheme with other deterministic designs like Lee and Stinson's [85] scheme using Transversal Designs and Çamtepe and Yener's [18] symmetric scheme. Suppose there are q^2 sensors (where q is a prime or prime power). The sensors are denoted by $U_{i,j}$, $i, j = 0, 1, 2, \cdots, q-1$. Each sensor contains x keys, such that $x < q$. The sensor $U_{i,j}$ contains the keys $\{(x, xi + j \mod q) : x < q\}$. Assuming that all nodes are within communication radius of each other, the initial number of links is $kq\binom{q-1}{2} = kq^2(q-1)/2$. When the number of keys $k = q - 1$, then the initial number of links is $q^2(q-1)^2/2$.

Now we consider our scheme and assume that $dim = 2$. Then the number of nodes supported is q^2. As calculated in Section 4.5.4 the number of links is $q^2(q-1)^2/2$.

So when $dim = 2$, our scheme has the same connectivity as that of Lee and Stinson's scheme. On compromising nodes randomly, we found that the $E(s)$ remains the same in both the schemes. However we should note that the keys in the nodes are different.

When $dim > 2$, we consider a network containing q^{dim} sensor nodes, where keys are distributed following our scheme. The number of links initially present is of the order of $O(q^{2dim})$. We consider a network of the same order, where the keys are distributed according to Lee and Stinson's approach. There are $O(q^{dim/2})$ nodes each having $q - 1$ keys. The number of links present is $O((q-1)q^{dim}(q^{dim/2} - 1))$. When we compare the number of links present between nodes in our scheme with that of Lee and Stinson's we find that the number of links present in our scheme is $O(q^{dim/2-1})$ that of theirs. For example for a network containing $4096 = 16^3 = 64^2$ nodes, each having 16 keys, our scheme has a connectivity 2.34 times that of Lee and Stinson's scheme. Thus we have better connectivity for the same number of keys and same size of the network.

We also compare our scheme with that of the symmetric design of Çamtepe and Yener [18] in Table 4.4. We find our scheme gives slightly better resiliency compared to their scheme for almost same size of network and almost equal number of keys. For example, when we compare with Çamtepe and Yener(CY) scheme we

q	CY scheme			Our scheme		
	N	k	$E(s)$	N	k	$E(s)$
61	3783	62	0.1528	3721	60	0.1524
71	5113	72	0.1328	5041	70	0.1326
73	5403	74	0.1289	5329	72	0.1288
101	10303	102	0.0992	10201	100	0.0972

Table 4.4: Comparative study of $E(s)$ of our scheme with that of Çamtepe and Yener's (CY) scheme of $E(s)$ for 100 runs for $s = 10$ compromised nodes

find that for 10303 nodes with 102 keys each Çamtepe and Yener's scheme gives $E(10) = 0.0992$. The resiliency achieved in our scheme by 10201 nodes and only 100 keys is 0.0972. An added advantage of our scheme over Çamtepe and Yener is that our scheme is scalable.

4.8 Key predistribution scheme using Orthogonal Arrays

We refer to the definition of Orthogonal Arrays given in Section 1.5. We can extend our construction to design key predistribution schemes from Orthogonal Arrays instead of codes. If we consider each row of an array to be a codeword $(a_1, a_2, \cdots, a_n)$, then using the construction given in Section 4.2 we can design a key predistribution scheme where the number of keys per node is n and the size of the network which can be supported is λv^t. A key predistribution scheme using orthogonal arrays was given by Dong, Pei and Wang [45].

4.9 Concluding remarks

Uptil now we had discussed schemes in which nodes are randomly deployed. In the next two chapters we discuss schemes which use deployment knowledge. The first of these scheme uses a grid based deployment, in which the sensors are placed at the grid intersection points. In the second scheme the entire deployment region is broken down into smaller regions. Within each region there are two types of nodes with varying battery power and storage.

Chapter 5

Key Predistribution using Transversal Design on a Grid of Wireless Sensor Network

We propose a deterministic key predistribution scheme in a Wireless Sensor Network (WSN), in which each sensor node (identifiable with a block in transversal design) is placed at each grid intersection point. Any two sensor nodes, when they are in the Radio Frequency (RF) range of each other, are allowed to communicate (in an encrypted manner) if they share a common secret key. We analyze the connectivity of the network taking the RF radius into account. Further, we study the security of the network when some sensor nodes are compromised to the adversaries.

We have already mentioned in Section 2.8 that there are several schemes which use deployment knowledge. It has been shown that schemes using deployment knowledge result in better resiliency, compared to schemes where deployment is random. In fact the information of deployment position helps us to improve the resiliency of the network.

Grid based design is one such approach of deterministic deployment in which the points of intersection are accessible and the sensor devices may be placed at those points with reasonable precision. As mentioned in [144] and [127] a grid based deployment finds application in intrusion detection. An unreliable GPS enabled grid network has also been mentioned in [144]. As we have already seen that deployment knowledge has been used in [49, 69, 70, 92, 149, 174, 179].

In this chapter, we consider a grid-based deployment scheme and we predistribute the keys according to the Transversal Design $TD(k, r)$. We consider an unit-disk graph model as given in [161, Section 11.1.1]. We assume that the sensing and the transmission radii are equal and call this radius - *Radio Frequency (RF) radius*. We study the connectivity and resiliency of the network taking RF radius into account. The importance of such analysis of RF radius lies in the fact that we can change the RF radius according to power requirements. As already mentioned

in Section 1.2.4, the radio frequency (RF) region around a sensor is the circular region around the center with the given RF radius. At first we approximate this to a square. We calculate the resiliency and connectivity of the network using this square region. We then make a better approximation of the RF region by assuming the region to be a Lee sphere with appropriate Lee distance.

The r^2 blocks (identify them as sensor nodes) of the TD are placed on a deployment grid of dimension $r \times r$. The main idea where our proposal differs from that of Lee and Stinson [86] is that in [86] the sensor nodes are scattered randomly on an unknown geometry unlike our model where we consider a known grid based deployment. If one considers the idea of [86], given r^2 sensor nodes with k keys in each node, any two sensor nodes (without considering whether these two nodes are in RF communication range) can share a common secret key with probability $\frac{k}{r+1}$. This is acceptable as the geometry is not known a priori. However, if the geometry is known, one may try to place the nodes in such a manner that any two of them in the same RF communication range should have a common key with more probability than that between two nodes which are not in the RF communication range. If one follows the idea of [86] in the known grid based deployment, by placing any random block (identified as a sensor node) of $TD(k, r)$ at any grid location, then a node will share a common key with probability $\frac{k}{r+1}$ with any of its neighboring node in the RF communication range. We like to get a better probability than $\frac{k}{r+1}$ by a proper placement. The most natural idea in this area is to put the block indexed by (i, j) at the location (i, j) and that works successfully. We present that the probability of key sharing we achieve here is much better than $\frac{k}{r+1}$ for small k's. Thus with very small number of keys in each node (which is a very interesting property as the keys are to be stored in memory and memory is constrained in a low cost sensor devices) we get a very good connectivity. We supplement this practical motivation with detailed theoretical results based on combinatorial analysis.

The rest of this chapter is organized in the following way. In Section 5.1, we define basic concepts. In Section 5.2, we find the connectivity ratio and upper bounds for $E(s)$ when RF region is a square. We do the same estimation when RF region is a Lee sphere in Section 5.3. We compare our scheme with other schemes in Section 5.4. This chapter is based on papers [131, 133].

5.1 Preliminaries

We recall the definition of a transversal design from Section 1.5. We denote a transversal design with $\lambda = 1$ as $TD(k, r)$. It can be shown that if there exists a $TD(k, r)$, then there exists a (v, b, r, k) design with $v = kr$, $b = r^2$.

We use the following TD for predistribution. Let $(X, \mathcal{A})$ be a transversal design $TD(k, r)$.

1. $X = \{(x, y) : 0 \leq x < k, 0 \leq y < r\}$,

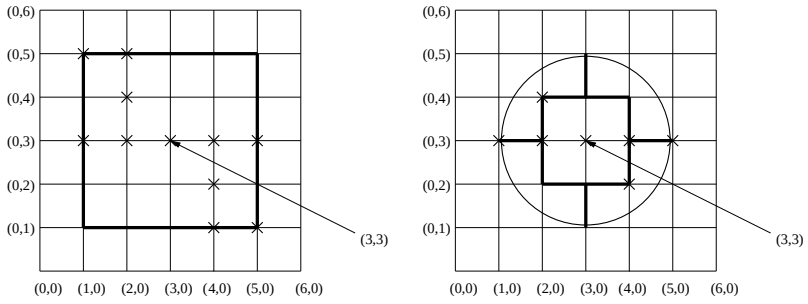
2. For all i, $G_i = \{(i, y) : 0 \leq y < r\}$,

3. $\mathcal{A} = \{A_{i,j} : 0 \leq i < r \ \& \ 0 \leq j < r\}$.

We define a block $A_{i,j}$ by

$$A_{i,j} = \{(x, xi + j \bmod r) : 0 \leq x < k\} \tag{5.1.1}$$

We consider an $r \times r$ grid such that there are r^2 points of intersection. For our purpose, we take a prime power r. We map the r^2 blocks to the r^2 sensor nodes and place block $A_{i,j}$ at the location (i, j) of the grid as shown in Figure 5.1(a). We represent the node at (i, j) as $U_{i,j}$. The varieties are mapped on to the secret keys in the sensor nodes. Thus we establish a correspondence between $TD(k, r)$ and the placement of sensor nodes on a $r \times r$ square grid. Note that any two blocks have either no key or one key in common and the algorithm to check whether the two nodes actually share a common secret key is efficient (see [86] for more details). Consider a square grid (as shown in figure 5.1(b)). A *Lee Sphere* [7] of radius ρ_l centered at a given point U_P consists of the set of points that lie at a Manhattan distance of at most ρ_l from U_P. ρ_l is called the Lee distance. The triangle inequality implies that the Manhattan distance between two nodes is greater than the Euclidean distance. This implies that all the nodes within the Lee sphere of radius ρ_l centered at a point U_P are also contained in the RF region of radius ρ_l centered at U_P. We see that a Lee sphere is a better approximation than a square RF region. (As given in 5.1(a) and 5.1(b)). We assume that two nodes can communicate with each other provided they are within Lee distance and have a common key.



(a) RF region is a square (b) RF region is a Lee Sphere

Figure 5.1: A 7×7 grid with $k = 3$, $\rho = \rho_l = 2$. The physical neighbors of the node at $(3, 3)$ occur along the dark lines and the key sharing neighbors are marked by crosses.

The sensor nodes can carry on effective communication only inside a particular range called the Radio Frequency (RF) range. The RF range with respect to a particular point is actually a circular region with center as that point and some radius around that. For our calculations, we assume that a node can communicate with other nodes within a square of dimension $2\rho \times 2\rho$, with the node itself at the

center. Thus the RF circle circumscribes the square. We refer to the RF radius as ρ.

Definition 5.1.1. Physical neighbor: *For a given node α located at (i, j) and a given radius ρ, a node β ($\neq \alpha$) located at (i', j') is said to be a physical neighbor of α, if $max(|i - i'|, |j - j'|) \leq \rho$.*

Note that the maximum number of physical neighbors is $(2\rho+1)^2-1 = 4\rho(\rho+1)$. For nodes at (or close to) the boundary, the number of physical neighbors are less.

Definition 5.1.2. Key sharing neighbor: *For a given node α located at (i, j) and its physical neighbor γ ($\neq \alpha$) located at (i', j') is said to be a key sharing neighbor of α, if γ has a common key with α.*

When we consider the RF region as a Lee sphere instead of a square, the mathematical definitions of physical and key-sharing neighbors change slightly. Lee distance is denoted by ρ_l.

Definition 5.1.3. Physical Lee neighbor : *For a given node α located at (i, j) and a given Lee Distance ρ_l, a node β ($\neq \alpha$) located at (i', j') is said to be a physical neighbor of α, if β is within the Lee sphere of radius ρ_l centered at α. Mathematically, $|j - j'| + |i - i'| \leq \rho_l$.*

Note that the maximum number of physical neighbors is $2\rho_l(\rho_l + 1)$. For nodes at (or close to) the boundary, the number of physical neighbors is less.

Definition 5.1.4. Key sharing Lee neighbor: *For a given node α located at (i, j) and its physical Lee neighbor γ ($\neq \alpha$) located at (i', j'), γ is said to be a key sharing neighbor of α, if γ has a key common with α.*

Note that one may think that why it is needed to assign the sensor node corresponding to the block $A_{i,j}$ at the grid location (i, j). In fact this natural correspondence cannot be the only justification and one may ask what will happen if one randomly assigns the block $A_{i,j}$ at the grid location (i', j'). That the natural correspondence is quite efficient than the random placement is theoretically justified in Subsection 5.2.2.2 later.

Fix a node α located at (i, j) and radius ρ. Consider the set $A_\rho^{(i,j)}$ of key sharing neighbors of α within a RF radius ρ and the set $B_\rho^{(i,j)}$ of physical neighbors of α within a RF radius ρ. We will calculate $|A_\rho^{(i,j)}|$ in Theorem 5.2.2 later.

In general, one may note that the number of physical neighbors,

$$|B_\rho^{(i,j)}| = (\min\{i, r - 1 - i, \rho\} + \rho + 1)(\min\{j, r - 1 - j, \rho\} + \rho + 1) - 1. \quad (5.1.2)$$

We call a node $U_{i,j}$ an *interior* node (not around the boundary), when $i \leq \rho$, $(r - 1 - i) \leq \rho$, $j \leq \rho$, $(r - 1 - j) \leq \rho$. For all the interior nodes $n_{i,j}$, $|B_\rho^{(i,j)}| = 4\rho(\rho + 1)$. We also note that $|B_{\rho_l}^{(i,j)}| = 2\rho_l(\rho_l + 1)$.

Definition 5.1.5. Connectivity Ratio: *The connectivity ratio $p_{c_\rho}^{(i,j)}$ of a node $U_{i,j}$ is defined as the ratio of the number of key sharing neighbors of $U_{i,j}$ and the number of physical neighbors of $U_{i,j}$. Mathematically, $p_{c_\rho}^{(i,j)} = \frac{|A_\rho^{(i,j)}|}{|B_\rho^{(i,j)}|}$.*

When the RF region is a Lee sphere then the ρ is replaced by ρ_l in the above definition.

5.2 When RF region is a square

We assume that the RF region is a square.

5.2.1 Key establishment

We now present the key establishment protocol between two sensor nodes. We consider a $r \times r$ grid, where r is prime, such that each node contains k keys. The RF radius ρ is small for practical purposes and can be assumed to be much less than $\frac{r+1}{2}$. We see that node $U_{i,j}$ has a common key $(0, j)$ with nodes $U_{0,j}$, $U_{1,j}$,$\cdots$, $U_{i-1,j}$, $U_{i+1,j}$, $\cdots$, $U_{r-1,j}$ and node $U_{i,j}$ do not share a common key with any of the nodes $U_{i,0}$, $U_{i,1}$, $\cdots$, $U_{i,j-1}$, $U_{i,j+1}$, $\cdots$, $U_{i,r-1}$. That is, all the nodes along a given row share a common key, and all nodes along a given column never share a common key. One may refer Figure 5.1 as an example. Two nodes $U_{i,j}$ and $U_{i',j'}$ share a common key [24, 86] if for some x, $0 \leq x < k$, $xi + j \equiv xi' + j' \bmod r$. It follows that, for $0 \leq x < k$, $x(i-i') \equiv j-j' \bmod r$ holds. So if $x \equiv (j'-j)(i-i')^{-1} \bmod r$, where $0 \leq x < k$, and $max\{|i-i'|, |j-j'|\} \leq \rho$, then the nodes $U_{i,j}$ and $U_{i',j'}$ will share a common key. Since r is prime, $(i-i')^{-1}$ always exists. If $i = j$, then $U_{i,j}$ and $U_{i',j'}$ do not share a common key. If $i \neq j$, $x = (j'-j)(i-i')^{-1} \bmod r$, where $0 \leq x < k$ is a common key. The common key can thus be efficiently calculated, since the inverse can be calculated efficiently by Extended Euclidean Algorithm in $O(log_2^2 r)$ time as shown in [155, Chapter 5].

5.2.2 Connectivity Analysis

In this section we present how one node shares the keys with the other nodes which are its physical neighbor.

We calculate the exact value of connectivity ratio.

5.2.2.1 Exact calculation of $p_{c_\rho}^{(i,j)}$

We give the exact value of $|A_\rho^{(i,j)}|$, when $\rho \leq \frac{r-1}{2}$. Consider a node $U_{i,j}$ that contains the keys indexed by $(0, j)$, $(1, (i + j) \bmod r)$,$\cdots$, $(k-1, (i(k-1) + j) \bmod r)$. According to our transversal design, any two nodes can share a maximum of one key. Therefore, to find the key sharing neighbors of the node $U_{i,j}$, it is sufficient

to find the number of nodes in which each of the $(x, (xi + j) \bmod r)$ keys occur, where $0 \leq x < k$. Suppose the node $U_{i,j}$ contains the key (x, y). We find the number of nodes within the RF radius ρ which also contain the key (x, y). Given a key (x, y), we find the nodes $U_{i,j}$ such that $y = (xi + j) \bmod r$. Hence the nodes $U_{i,j}$ must satisfy the equation

$$j = (y - xi) \bmod r \tag{5.2.1}$$

Note that the key (x, y) occurs in nodes $U_{0,y}$, $U_{1,y-x}$, $\cdots$, $U_{r-1,y-x(r-1) \bmod r}$.

The node $U_{i,j}$ contains the keys $(x, y = xi + j \bmod r)$ where $0 \leq x < k$. By Equation (5.3.2), if $U_{i+t,j'}$ is a key sharing node of $U_{i,j}$, then $j' = (y - x(i+t)) \bmod r$. So $j' = (j - xt) \bmod r$. The key sharing neighbors of $U_{i,j}$ which share the the key (x, y) are the following: $U_{i+1,j-x}$, $U_{i+2,j-2x}$, $\cdots$, $U_{i+t,j-xt}$ and $U_{i-1,j+x}$, $U_{i-2,j+2x}$, $\cdots$, $U_{i-t,j+xt}$.

To find the key sharing neighbors of $U_{i,j}$, we refer to the Figure 5.2. We find the key sharing neighbors in the four quadrants. The following cases arise.

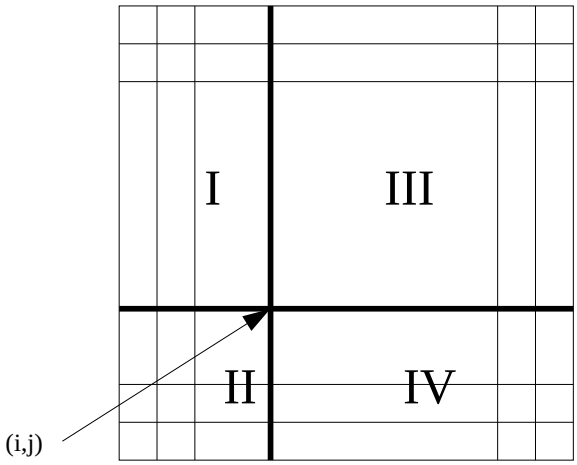


Figure 5.2: A $r \times r$ grid showing the four types of neighboring nodes

Case a: When $i' < i$, we consider the nodes $U_{i-1,j+x}$, $U_{i-2,j+2x}$, $\cdots$, $U_{i-t,j+tx}$, where $0 \leq x \leq k - 1$.

If $j \leq j'$ (when the neighboring nodes are in quadrant I), then the following conditions must be satisfied.

$$0 < t \leq \min\{i, \rho\}, 0 \leq x < k \; and \; tx \bmod r \leq \min\{\rho, r - 1 - j\} \tag{5.2.2a}$$

If $j > j'$ (when the neighboring nodes are in quadrant II), then the following conditions must be satisfied.

$$0 < t \leq \min\{i, \rho\}, 0 \leq x < k \; and \; r - (tx \bmod r) \leq \min\{\rho, j\} \tag{5.2.2b}$$

Case b: When $i' > i$, we consider the nodes $U_{i+1,j-x}$, $U_{i+2,j-2x}$, $\cdots$, $U_{i+t,j-tx}$, where $0 \leq x \leq k - 1$.

If $j \geq j'$ (when the neighboring nodes are in quadrant IV), then the following conditions must be satisfied.

$$0 < t \leq \min\{r-1-i, \rho\}, 0 \leq x \leq k-1 \text{ and } tx \bmod r \leq \min\{\rho, j\} \quad (5.2.3a)$$

If $j < j'$ (when the neighboring nodes are in quadrant III), then the following conditions must be satisfied.

$$0 < t \leq \min\{r-1-i, \rho\}, 0 \leq x \leq k-1 \text{ and } r-(tx \bmod r) \leq \min\{\rho, r-1-j\} \quad (5.2.3b)$$

So, the number of solutions (t, x) satisfying the above equations (5.2.2a), (5.2.2b), (5.2.3a) and (5.2.3b) give the number of key sharing neighbors of $U_{i,j}$ within the RF radius ρ.

The following lemma is crucial in finding the exact value of $|A_\rho^{(i,j)}|$.

Lemma 5.2.1. *Let w be a prime such that $w \geq 2T-1$. Then the number of solutions (t, x) satisfying the equation*

$$0 < S' \leq tx \bmod w \leq S \leq w-1 \quad (5.2.4)$$

where, $0 < t \leq T < w$ and $0 \leq x \leq X < w$, is given by $\sum_{t=1}^{T}\sum_{l=0}^{t-1} S_1$ where $u_1 = (wl+S')/t, u_2 = (wl+S)/t$ and

$$S_1 = \begin{cases} 0, & \text{if } X+1 \leq u_1 \leq u_2, \\ X+1-\lceil u_1 \rceil & \text{if } \lceil u_1 \rceil < X+1 \leq \lfloor u_2 \rfloor, \\ \lfloor u_2 \rfloor - \lceil u_1 \rceil + 1 & \text{if } \lfloor u_2 \rfloor < X+1. \end{cases}$$

Proof. When $t = 1$, three conditions can arise.

Case (i): If $X+1 \leq S'$, then there are no values of x which satisfy (5.2.4).
Case (ii): If $S' < X+1 \leq S$, $x = S', S'+1, S'+2, \cdots, X$ satisfy (5.2.4). So there are $X+1-S'$ solutions.
Case (iii): If $S \leq X$, $x = S', S'+1, S'+2, \cdots, S$ satisfy (5.2.4). So there are $S-S'+1$ solutions.

When $t = 2$, three conditions can arise.
Case (i): If $X+1 \leq \lceil \frac{S'}{2} \rceil$, then there are no values of x which satisfy (5.2.4).
Case (ii): If $\lceil \frac{S'}{2} \rceil < X+1 \leq \lfloor \frac{S}{2} \rfloor$, then $x = \lceil \frac{S'}{2} \rceil, \lceil \frac{S'}{2} \rceil + 1, \cdots, X$ satisfy (5.2.4). So, there are $X+1-\lceil \frac{S'}{2} \rceil$ solutions.
Case (iii): If $\lfloor \frac{S}{2} \rfloor \leq X$, then $x = \lceil \frac{S'}{2} \rceil, \lceil \frac{S'}{2} \rceil + 1, \cdots, \lfloor \frac{S}{2} \rfloor$ satisfy (5.2.4). So, there are $\lfloor \frac{S}{2} \rfloor - \lceil \frac{S'}{2} \rceil + 1$ solutions.

When Case (iii) arises, then again consider the three sub cases.
Case (iii a): If $X+1 \leq \lceil \frac{w+S'}{2} \rceil$, then there are no values of x which satisfy (5.2.4).
Case (iii b): If $\lceil \frac{w+S'}{2} \rceil < X+1 \leq \lfloor \frac{w+S}{2} \rfloor$, then $x = \lceil \frac{w+S'}{2} \rceil, \lceil \frac{w+S'}{2} \rceil + 1, \cdots, X$ satisfy (5.2.4). So there are $X - \lceil \frac{w+S'}{2} \rceil + 1$ such solutions.
Case (iii c): If $\lfloor \frac{w+S}{2} \rfloor \leq X$, then $x = \lceil \frac{w+S'}{2} \rceil, \lceil \frac{w+S'}{2} \rceil + 1, \cdots, \lfloor \frac{w+S}{2} \rfloor$ satisfy (5.2.4).

So there are $\lfloor\frac{w+S}{2}\rfloor - \lceil\frac{w+S'}{2}\rceil + 1$ such solutions.
Note that for $\lfloor\frac{S}{2}\rfloor < x < \lceil\frac{w+S'}{2}\rceil$, there is no solution when $t = 2$. The above cases give all the solutions when $t = 2$, since $\lceil\frac{lw+S'}{2}\rceil > w$, for $l > 1$. So, the number of solutions when $t = 2$, is $\sum_{l=0}^{1} S_1$, where S_1 is as given.

Proceeding as above, $t = l$, three conditions can arise.
Case (i): If $X + 1 \leq \lceil\frac{S'}{l}\rceil$, then there are no values of x which satisfy (5.2.4).
Case (ii): If $\lceil\frac{S'}{l}\rceil < X + 1 \leq \lfloor\frac{S}{l}\rfloor$, then, $x = \lceil\frac{S'}{l}\rceil, \lceil\frac{S'}{l}\rceil + 1, \cdots, X$ satisfy (5.2.4). So, there are $X + 1 - \lceil\frac{S'}{l}\rceil$ solutions.
Case (iii): If $\lfloor\frac{S}{l}\rfloor \leq X$, then, $x = \lceil\frac{S'}{l}\rceil, \lceil\frac{S'}{l}\rceil + 1, \cdots, \lfloor\frac{S}{l}\rfloor$ satisfy (5.2.4). So, there are $\lfloor\frac{S}{l}\rfloor - \lceil\frac{S'}{l}\rceil + 1$ solutions.

When Case (iii) arises, then again consider the three sub cases.
Case (iii a): If $X + 1 \leq \lceil\frac{w+S'}{l}\rceil$, then there are no values of x which satisfy (5.2.4).
Case (iii b): If $\lceil\frac{w+S'}{l}\rceil < X + 1 \leq \lfloor\frac{w+S}{l}\rfloor$, then $x = \lceil\frac{w+S'}{l}\rceil, \lceil\frac{w+S'}{l}\rceil + 1, \cdots, X$ satisfy (5.2.4). So there are $X - \lceil\frac{w+S'}{l}\rceil + 1$ such solutions.
Case (iii c): If $\lfloor\frac{w+S}{l}\rfloor \leq X$, then $x = \lceil\frac{w+S'}{l}\rceil, \lceil\frac{w+S'}{l}\rceil + 1, \cdots, \lfloor\frac{w+S}{l}\rfloor$ satisfy (5.2.4). So there are $\lfloor\frac{w+S}{l}\rfloor - \lceil\frac{w+S'}{l}\rceil + 1$ such solutions.

Note that for $\lfloor\frac{S}{l}\rfloor < x < \lceil\frac{w+S'}{l}\rceil$, there is no solution when $t = l$.

Again for Case (iii c), three cases can arise. Continuing similarly, we notice that if $\lfloor\frac{(l-2)w+S}{l}\rfloor \leq X$, then three conditions will arise.
Case (a): If $X+1 \leq \lceil\frac{(l-1)w+S'}{l}\rceil$, then there are no values of x which satisfy (5.2.4).
Case (b): If $\lceil\frac{(l-1)w+S'}{l}\rceil < X + 1 \leq \lfloor\frac{(l-1)w+S}{l}\rfloor$, then $x = \lceil\frac{(l-1)w+S'}{l}\rceil, \lceil\frac{(l-1)w+S'}{l}\rceil$ $+ 1, \cdots, X$ satisfy (5.2.4). So there are $X - \lceil\frac{(l-1)w+S'}{l}\rceil + 1$ such solutions.
Case (c): If $\lfloor\frac{(l-1)w+S}{l}\rfloor \leq X$, then $x = \lceil\frac{(l-1)w+S'}{l}\rceil, \lceil\frac{(l-1)w+S'}{l}\rceil + 1, \cdots, \lfloor\frac{(l-1)w+S}{l}\rfloor$ satisfy (5.2.4). So there are $\lfloor\frac{(l-1)w+S}{l}\rfloor - \lceil\frac{(l-1)w+S'}{l}\rceil + 1$ such solutions.

Note that for $\lfloor\frac{(l-2)w+S}{l}\rfloor < x < \lceil\frac{(l-1)w+S'}{l}\rceil$, there is no solution when $t = l$. These are the only solutions when $t = l$, since $\lceil\frac{l''w+S'}{l}\rceil > w$, for $l'' > l$. So, the number of solutions when $t = l$, is $\sum_{m=0}^{l-1} S_1$, where S_1 is as given.

Hence for all values of t, $1 \leq t \leq T$, there are $\sum_{t=1}^{T}\sum_{l=0}^{t-1} S_1$ solutions satisfying (5.2.4) ∎

The example below will help us understand the above lemma better. We consider the equation

$$0 < tx \bmod 7 \leq 5 \textit{ and } 0 < t \leq 3 \textit{ and } 0 \leq x \leq 4. \tag{5.2.5}$$

Note that w is a prime and $w \geq 2T - 1$. For $t = 1$, the tuples $(1, 1)$, $(1, 2)$, $(1, 3)$, $(1, 4)$ satisfy (5.2.5). So, there are four solutions when $t = 1$. Here $\lceil u_1\rceil = 1$, $\lfloor u_2\rfloor = 5$. Since $1 < X + 1 \leq 5$, from the formula in Lemma 5.2.1 there are $X + 1 - \lceil u_1\rceil = 4 + 1 - 1 = 4$ solutions.

For $t = 2$, the tuples $(2, 1)$, $(2, 2)$, $(2, 4)$ satisfy (5.2.5). So, there are three solutions when $t = 2$. When $l = 0$, $\lceil u_1\rceil = 1$, $\lfloor u_2\rfloor = 2$. Since $X > 2$, there are

$\lfloor u_2 \rfloor + 1 - \lceil u_1 \rceil = 2 + 1 - 1 = 2$ solutions. When $l = 1$, $\lceil u_1 \rceil = 4$, $\lfloor u_2 \rfloor = 6$. Since $\lceil u_1 \rceil < X + 1 \leq \lfloor u_2 \rfloor$, there are $X + 1 - \lceil u_1 \rceil = 4 + 1 - 4 = 1$ solutions.

For $t = 3$, the tuples $(3, 1)$, $(3, 3)$, $(3, 4)$ satisfy (5.2.5). So, there are three solutions when $t = 2$. When $l = 0$, $\lceil u_1 \rceil = 1$, $\lfloor u_2 \rfloor = 1$. Since $X > 1$, there is $\lfloor u_2 \rfloor + 1 - \lceil u_1 \rceil = 1 + 1 - 1 = 1$ solutions. When $l = 1$, $\lceil u_1 \rceil = 3$, $\lfloor u_2 \rfloor = 4$. Since $X \geq \lfloor u_2 \rfloor$, there are $\lfloor u_2 \rfloor - \lceil u_1 \rceil = 4 - 3 + 1 = 2$ solutions. When $l = 2$, $\lceil u_1 \rceil = 5$, $\lfloor u_2 \rfloor = 6$. Since $X + 1 \leq 5$, there is no solution. All these three cases provide the overall count.

Using Lemma 5.2.1 and conditions (5.2.2a), (5.2.2b), (5.2.3a) and (5.2.3b) we arrive at the following theorem.

Theorem 5.2.2. $|A_\rho^{(i,j)}| = \min\{i, \rho\} + \min\{r - 1 - i, \rho\} + \sum_{t=1}^{\min\{i,\rho\}} (\sum_{l=0}^{t-1} \Gamma + \sum_{l=1}^{t} \Delta) + \sum_{t=1}^{\min\{r-1-i,\rho\}} (\sum_{l=0}^{t-1} \Phi + \sum_{l=1}^{t} \Psi)$ *where* $\gamma_1 = (rl + 1)/t$, $\gamma_2 = (rl + \min\{r-1-j, \rho\})/t$, $\delta_1 = (rl - \min\{j, \rho\})/t$, $\delta_2 = (rl-1)/t$, $\phi_2 = (rl + \min\{j, \rho\})/t$, $\psi_1 = (rl - \min\{r - 1 - j, \rho\})/t$ *and*

$$\Gamma = \begin{cases} 0, & \text{if } k \leq \gamma_1 \leq \gamma_2, \\ k - \lceil \gamma_1 \rceil & \text{if } \lceil \gamma_1 \rceil < k \leq \lfloor \gamma_2 \rfloor, \\ \lfloor \gamma_2 \rfloor - \lceil \gamma_1 \rceil + 1 & \text{if } \lfloor \gamma_2 \rfloor < k, \end{cases}$$

$$\Delta = \begin{cases} 0, & \text{if } k \leq \delta_1 \leq \delta_2, \\ k - \lceil \delta_1 \rceil & \text{if } \lceil \delta_1 \rceil < k \leq \lfloor \delta_2 \rfloor, \\ \lfloor \delta_2 \rfloor - \lceil \delta_1 \rceil + 1 & \text{if } \lfloor \delta_2 \rfloor < k, \end{cases}$$

$$\Phi = \begin{cases} 0, & \text{if } k \leq \gamma_1 \leq \phi_2, \\ k - \lceil \gamma_1 \rceil & \text{if } \lceil \gamma_1 \rceil < k \leq \lfloor \phi_2 \rfloor, \\ \lfloor \phi_2 \rfloor - \lceil \gamma_1 \rceil + 1 & \text{if } \lfloor \phi_2 \rfloor < k, \end{cases}$$

$$\Psi = \begin{cases} 0, & \text{if } k \leq \psi_1 \leq \delta_2, \\ k - \lceil \psi_1 \rceil & \text{if } \lceil \psi_1 \rceil < k \leq \lfloor \delta_2 \rfloor, \\ \lfloor \delta_2 \rfloor - \lceil \psi_1 \rceil + 1 & \text{if } \lfloor \delta_2 \rfloor < k. \end{cases}$$

Proof. When $x = 0$, $(i - 1, 0)$, $(i - 2, 0)$,$\cdots$,$(i - \min\{i, \rho\}, 0)$ satisfy (5.2.2a).

For $x \neq 0$, we can map (5.2.2a) to Lemma 5.2.1. Here, $w = r$, $S' = 1$, $S = \min\{\rho, r - 1 - j\}$, $T = \min\{i, \rho\}$, $X = k - 1$. So number of solutions satisfying (5.2.2a) is given by $\min\{i, \rho\} + \sum_{t=1}^{\min\{i,\rho\}} \sum_{l=0}^{t-1} \Gamma$.

We map (5.2.2b) to Lemma 5.2.1. Here, $w = r$, $S' = r - \min\{\rho, j\}$, $S = r - 1$, $T = \min\{i, \rho\}$, $X = k - 1$. So number of solutions is given by $\sum_{t=1}^{\min\{i,\rho\}} \sum_{l=1}^{t} \Delta$.

When $x = 0$, $(i + 1, 0)$, $(i + 2, 0)$,$\cdots$,$(i - \min\{r - i - 1, \rho\}, 0)$ satisfy (5.2.3a).

For $x \neq 0$, we can map (5.2.3a) and (5.2.3b) to Lemma 5.2.1 as above. So the number of solutions to (5.2.3a) and (5.2.3b) is given by $\min(r - 1 - i, \rho) + \sum_{t=1}^{\min\{r-i-1,\rho\}} (\sum_{l=0}^{t-1} \Phi + \sum_{l=1}^{t} \Psi)$. Hence the theorem follows. ∎

ρ	1	2	3	4	5	6	7	8
p_{c_ρ}	0.5	0.41667	0.3333	0.3000	0.25000	0.23810	0.20536	0.20139

Table 5.1: Connectivity Ratio p_{c_ρ} for interior nodes with change in RF radius, for a 47×47 grid with 7 keys per node.

Note that the value of $|A_\rho^{(i,j)}|$ from the above theorem and the value of $|B_\rho^{(i,j)}|$ from Equation(5.3.1) directly provides $p_{c_\rho}^{(i,j)} = \frac{|A_\rho^{(i,j)}|}{|B_\rho^{(i,j)}|}$.

The Table 5.1 shows the connectivity ratio for an interior node corresponding to different RF radii.

5.2.2.2 Comparison of our design with random deployment

The deterministic assignment of a node (the block $A_{i,j}$ in transversal design) at the grid location (i, j) fares better than deploying the nodes randomly. This is because in random deployment for any node $U_{i,j}$, a key (x, y) occurs in $r - 1$ other nodes. Note that the node $U_{i,j}$ shares one key with each of other $k(r - 1)$ nodes. For any node chosen randomly, there are $r^2 - 1$ physical neighbors. Thus the connectivity ratio is $\frac{k}{r+1}$. So for small values of k, our approach gives better results than random deployment. The choice of small values of k can be fully justified by the fact that the sensor nodes have very limited storage capacities. For example, for $r = 47$, $k = 7$, random deployment gives a connectivity ratio of 0.14583. However, for grid based deployment connectivity ratio maybe be more than double, equal to 0.33 and 0.3 for RF radius equal to 3 and 4 respectively. Again, to obtain a connectivity ratio of 0.25 (which is for RF radius 5 in grid based design), we need 12 keys in case of random deployment. However, our design uses only 7 keys.

5.2.3 Security related parameters

In this section we calculate the upper bound for $E(s)$ and give some experimental results for $V(s)$.

5.2.4 Calculation of upper bound for $E(s)$

Let us denote the total number of links by T. Hence $T = \frac{1}{2}\sum_{i=0}^{r-1}\sum_{j=0}^{r-1}|A_\rho^{(i,j)}|$.

There are a total of rk keys. If nodes are compromised such that all rk keys are compromised, then all the links are broken. However for simplicity we assume that only a small fraction of the nodes are compromised. Suppose s nodes are compromised. Maximum links are broken when the nodes compromised have disjoint sets of keys and occur at the interior.

Let the number of links broken when s nodes are compromised be denoted by B_s.

Consider a node $U_{i,j}$ which has been compromised. Let it contain key (x, y). We find the nodes $U_{i',j'}$ within RF radius of $U_{i,j}$ which share the key (x, y). By the analysis of $A_\rho^{(i,j)}$, we see that the nodes $U_{i-1,j+x}$, $U_{i-2,j+2x}$, $\cdots$, $U_{i-t,j+tx}$ and $U_{i+1,j-x}$, $U_{i+2,j-2x}$, $\cdots$, $U_{i+t,j-tx}$ are the key sharing neighbors of the node $U_{i,j}$. So $|t| \leq \rho$ and either $tx \bmod r \leq \rho$ or $r - |tx \bmod r| \leq \rho$. Since x is known, the number of nodes sharing key (x, y) within the RF radius is the same as finding the number of values of t which satisfy the equations

$$|t| \leq \rho \text{ and } |tx \bmod r| \leq \rho \tag{5.2.6a}$$

and

$$|t| \leq \rho \text{ and } r - |tx \bmod r| \leq \rho \tag{5.2.6b}$$

The theorem below gives the maximum value of B_s.

Theorem 5.2.3. *$B_s \leq r\sum_{m=1}^{s}(\sum_{n=0}^{k-1}(\sum_{l=0}^{x_{mn}-1}\Gamma + \sum_{l=1}^{x_{mn}}\Delta))$ where $\gamma_1 = (rl + 1)/x_{mn}$, $\gamma_2 = (rl + \rho)/x_{mn}$, $\delta_1 = (rl - \rho)/x_{mn}$ and $\delta_2 = (rl - 1)/x_{mn}$ and*

$$\Gamma = \begin{cases} 0, & \text{if } \rho + 1 \leq \gamma_1 \leq \gamma_2, \\ \rho + 1 - \lceil\gamma_1\rceil & \text{if } \lceil\gamma_1\rceil < \rho + 1 \leq \lfloor\gamma_2\rfloor, \\ \lfloor\gamma_2\rfloor - \lceil\gamma_1\rceil + 1 & \text{if } \lfloor\gamma_2\rfloor < \rho + 1, \end{cases}$$

$$\Delta = \begin{cases} 0, & \text{if } \rho + 1 \leq \delta_1 \leq \delta_2, \\ \rho + 1 - \lceil\delta_1\rceil & \text{if } \lceil\delta_1\rceil < \rho + 1 \leq \lfloor\delta_2\rfloor, \\ \lfloor\delta_2\rfloor - \lceil\delta_1\rceil + 1 & \text{if } \lfloor\delta_2\rfloor < \rho + 1, \end{cases}$$

Proof. Let the keys lost when s nodes are compromised be (x_{10}, y_{10}), (x_{11}, y_{11}), $\cdots$, (x_{1k-1}, y_{1k-1}), (x_{20}, y_{20}), (x_{21}, y_{21}), $\cdots$, (x_{2k-1}, y_{2k-1}), $\cdots$, (x_{s0}, y_{s0}), (x_{s1}, y_{s1}), $\cdots$, (x_{sk-1}, y_{sk-1}). We assume that the nodes compromised are interior nodes and the keys are all distinct. If a node $m = U_{i,j}$ is compromised, then all the keys (x_{m0}, y_{m0}), (x_{m1}, y_{m1}), $\cdots$, (x_{mk-1}, y_{mk-1}), contained therein become ineffective. We can calculate the number of nodes within RF radius which share a particular key (x_{mn}, y_{mn}). The key (x_{mn}, y_{mn}) occurs in r nodes. Since any two nodes share a maximum of one key, the number of nodes which share common keys with node m will be

$$r\sum_{n=0}^{k-1}(\text{number of nodes that share key } (x_{mn}, y_{mn})).$$

The total number of links broken will be

$$\tfrac{1}{2}r\sum_{m=1}^{s}(\sum_{n=0}^{k-1}(\text{number of nodes that share key } (x_{mn}, y_{mn}))).$$

We find the number of nodes which share the key (x_{mn}, y_{mn}) within the RF radius. This is the same as finding the number of solutions to (5.2.6a) and (5.2.6b). $t = \lceil 1/x_{mn}\rceil$, $\lceil 1/x_{mn}\rceil + 1$, $\cdots$,$\lfloor\rho/x_{mn}\rfloor$ satisfy (5.2.6a).

There are $\lfloor\rho/x_{mn}\rfloor$ such values of t.

r	k	ρ	s	$E(s)(Experimental)$	$Bound\ for\ E(s)$
23	15	7	5	0.2006	0.2170
23	15	5	5	0.2008	0.2171
31	20	7	5	0.1516	0.1609
31	25	7	5	0.1513	0.1609
37	30	7	5	0.1283	0.1350
53	49	7	5	0.0915	0.0942
53	49	7	10	0.1736	0.1885

Table 5.2: Experimental value of $E(s)$ for 100 runs and bound for $E(s)$, when number of nodes in the grid is r^2, keys per node is k and the RF radius is ρ.

$t = \lceil \frac{r+1}{x_{mn}} \rceil, \lceil \frac{r+2}{x_{mn}} \rceil + 1, \cdots, \lfloor \frac{r+\rho}{x_{mn}} \rfloor$ satisfy (5.2.6a).
There are $\lfloor \frac{r+\rho}{x_{mn}} \rfloor - \lceil \frac{r+1}{x_{mn}} \rceil + 1$ such values of t, if $\lfloor \frac{r+\rho}{x_{mn}} \rfloor < \rho + 1$
And there are $k - \lceil \frac{r+1}{x_{mn}} \rceil$ solutions, if $\lceil \frac{r+1}{x_{mn}} \rceil < \rho + 1 \leq \lfloor \frac{r+\rho}{x_{mn}} \rfloor$.

Continuing similarly, $t = \lceil \frac{(x_{mn}-1)r+1}{x_{mn}} \rceil, \lceil \frac{(x_{mn}-1)r+1}{x_{mn}} \rceil + 1, \cdots, \lfloor \frac{(x_{mn}-1)r+\rho}{x_{mn}} \rfloor$ satisfy (5.2.6a).

There are $\lfloor \frac{(x_{mn}-1)r+\rho}{x_{mn}} \rfloor - \lceil \frac{(x_{mn}-1)r+1}{x_{mn}} \rceil + 1$ such values of t, if $\lfloor \frac{(x_{mn}-1)r+\rho}{x_{mn}} \rfloor < \rho + 1$.
And there are $k - \lceil \frac{(x_{mn}-1)r+1}{x_{mn}} \rceil$ solution, if $\lceil \frac{(x_{mn}-1)r+1}{x_{mn}} \rceil < \rho + 1 \leq \lfloor \frac{(x_{mn}-1)r+\rho}{x_{mn}} \rfloor$.

These are the only solutions satisfying (5.2.6a). So there are $2\sum_{l=0}^{x_{mn}-1} \Gamma$ solutions satisfying (5.2.6a), where A is as given above.

By the same argument and proofs of Lemma 5.2.1 and Theorem 5.2.2, we can show that there are $2\sum_{l=1}^{x_{mn}} \Delta$ solutions satisfying (5.2.6b), where Δ is as given above.

Hence the proof. ∎

$E(s) = \frac{B_s}{T}$, where $T = \frac{1}{2}\sum_{i=0}^{r-1}\sum_{j=0}^{r-1} |A_\rho^{(i,j)}|$ and the bound for B_s is given by Theorem 5.2.3

The position of the compromised nodes affect the value of $E(s)$. The experimental results for the calculation of $E(s)$ under node compromise is given in Table 5.2. We observe that the experimental results are very close to the upper bound, as obtained above. The reasons that they are exactly not same are (i) when calculating the upper bound we have considered that the keys exposed from different blocks are disjoint and (ii) we consider the interior nodes are compromised in the theoretical result (Theorem 5.2.3).

5.2.5 Experimental results for $V(s)$

$V(s)$ can be defined as the probability that a node is disconnected, given that s nodes are compromised.

For $s < k$, no node is disconnected. For any node to be disconnected each of its k keys must be present in some compromised node. However no two or more

r	k	s	$V(s)$
11	5	9	0.0180
13	5	9	0.0250
37	5	30	0.0350
47	5	30	0.0150
47	6	30	0.0096
47	7	30	0.0027
47	9	30	0.0004
53	13	30	0.0000

Table 5.3: Experimental value of $V(s)$ when number of nodes in the grid is r^2, keys per node is k and s nodes are compromised

keys that are present in the disconnected node can be present in any compromised node, since any pair of nodes share at most one key. Hence there is no node which will have all the k keys in the compromised s sensor nodes.

The number of disconnected nodes depends on the position of the compromised nodes. For example, if all nodes in a particular column are compromised, then all the nodes are disconnected.

The experimental results for the calculation of $V(s)$ is given in Table 5.3 for 100 runs. It can be seen that very few nodes are compromised even for sufficiently small values of k. For larger values of k no nodes are disconnected at all.

5.3 When the RF regions is a Lee sphere

5.3.1 Key exchange

We now present the key exchange protocol between two sensor nodes. We consider a $r \times r$ grid, where r is a prime power, such that each node contains k keys. The Lee Distance ρ_l is small for practical purposes and can be assumed to be much less than $\frac{r+1}{2}$. We see that node $U_{i,j}$ shares the common key $(0, j)$ with nodes $U_{0,j}$, $U_{1,j}$,$\cdots$, $U_{i-1,j}$, $U_{i+1,j}$, $\cdots$, $U_{r-1,j}$ and node $U_{i,j}$ do not share a common key with any of the nodes $U_{i,0}$, $U_{i,1}$, $\cdots$, $U_{i,j-1}$, $U_{i,j+1}$, $\cdots$, $U_{i,r-1}$. One may refer Figure 5.1(b) as an example. Two nodes $U_{i,j}$ and $U_{i',j'}$ share a common key [23, 24, 86] if for some x, $0 \leq x < k$, $xi + j \equiv xi' + j' \bmod r$ (by Equation (8.2.1)). It follows that, for $0 \leq x < k$, $x(i - i') \equiv j' - j \bmod r$ holds. So if $x \equiv (j' - j)(i - i')^{-1} \bmod r$, where $0 \leq x < k$, and $|i - i'| + |j - j'| \leq \rho_l$, then the nodes $U_{i,j}$ and $U_{i',j'}$ will share a common key. If $i = i'$, then $U_{i,j}$ and $U_{i',j'}$ do not share a common key. If $i \neq i'$, $x = (j' - j)(i - i')^{-1} \bmod r$, where $0 \leq x < k$ is a common key. Note that since p is prime and $i \neq i'$, $(i - i')^{-1}$ exists. The common key can thus be efficiently calculated, since the inverse can be calculated efficiently by Extended Euclidean Algorithm in $O(\log_2^2 r)$ time as shown in [155, Chapter 5]. If two i and j nodes

do not share a common key, then there exists an intermediary t node such that i and t share some common key k_{it} and j and t share a common key k_{jt}. Node i chooses some random key K encrypts it with k_{it} and sends it to t. t decrypts it using k_{it} and encrypts it using k_{jt} and sends it to j. j decrypts K using k_{jt}. All communications between i and j takes place using the key K.

5.3.2 Connectivity Analysis

In this section we calculate the number of nodes within Lee distance which share a common key with a given node.

Fix a node α located at (i, j) and Lee distance ρ_l. Consider the set $A_{\rho_l}^{(i,j)}$ of key sharing neighbors of α within the Lee Distance ρ_l and the set $B_{\rho_l}^{(i,j)}$ of physical neighbors of α within a Lee Distance ρ_l. We will calculate $|A_{\rho}^{(i,j)}|$ in Theorem 5.3.1 later.

We call a node $U_{i,j}$ an *interior* node (not around the boundary), if $i \geq \rho_l$, $(r-1-i) \geq \rho_l$, $j \geq \rho_l$, $(r-1-j) \geq \rho_l$. For all the interior nodes $U_{i,j}$,

$$|B_{\rho_l}^{(i,j)}| = 2\rho_l(\rho_l + 1) \tag{5.3.1}$$

Recall the definition of Connectivity Ratio from Section 5.1.

We calculate the value of connectivity ratio for an interior node.

5.3.3 Calculation of connectivity ratio $p_{c_{\rho_l}}^{(i,j)}$ of interior node

We give the value of $A_{\rho_l}^{(i,j)}$ for an interior node $U_{i,j}$, when $\rho_l \leq \frac{r-1}{2}$. Consider an interior node $U_{i,j}$ that contains the keys indexed by $(0, j)$, $(1, (i + j) \bmod r), \cdots$, $(k-1, (i(k-1) + j) \bmod r)$. According to our transversal design, any two nodes can share a maximum of one key. Therefore, to find the key sharing neighbors of the node $U_{i,j}$, it is sufficient to find the number of nodes in which each of the $(x, (xi + j) \bmod r)$ keys occur, where $0 \leq x < k$. Suppose the node $U_{i,j}$ contains the key (x, y). We find the number of nodes within the Lee distance ρ_l which also contain the key (x, y). Given a key (x, y), we find the nodes $U_{i,j}$ such that $y = xi + j \bmod r$. Hence the nodes $U_{i,j}$ must satisfy the equation

$$j = (y - xi) \bmod r \tag{5.3.2}$$

Note that the key (x, y) occurs in nodes $U_{0,y}$, $U_{1,y-x}$, $\cdots$, $U_{r-1,y-x(r-1) \bmod r}$.

The node $U_{i,j}$ contains the keys $(x, y = xi + j \bmod r)$ where $0 \leq x < k$. By equation (5.3.2), if $U_{i+t,j'}$ is a key sharing node of $U_{i,j}$, then $j' = (y - x(i+t)) \bmod r$. So $j' = (j - xt) \bmod r$. The key sharing neighbors of $U_{i,j}$ which share the key (x, y) are the following: $U_{i+1,j-x}$, $U_{i+2,j-2x}$, $\cdots$, $U_{i+t,j-xt}$ and $U_{i-1,j+x}$, $U_{i-2,j+2x}$, $\cdots$, $U_{i-t,j+xt}$.

To find $U_{i',j'}$ the key sharing neighbors of $U_{i,j}$, we refer to the Figure 5.2. We find the key sharing neighbors in the four quadrants. The following cases arise.

Case a: When $i' < i$, we consider the nodes $U_{i-1,j+x}$, $U_{i-2,j+2x}$, $\cdots$, $U_{i-t,j+tx}$, where $0 \leq x < k$.

If $j \leq j'$ (when the neighboring nodes are the quadrant I), then the following conditions must be satisfied.

$$0 < t \leq \rho_l, 0 \leq x < k \; and \; tx \bmod r \leq \rho_l - t \tag{5.3.3a}$$

If $j > j'$ (when the neighboring nodes are in quadrant II), then the following conditions must be satisfied.

$$0 < t \leq \rho_l, 0 \leq x < k \; and \; r - (tx \bmod r) \leq \rho_l - t \tag{5.3.3b}$$

Case b: When $i' > i$, we consider the nodes $U_{i+1,j-x}$, $U_{i+2,j-2x}$, $\cdots$, $U_{i+t,j-tx}$, where $0 \leq x < k$.

If $j \geq j'$ (when the neighboring nodes are in quadrant IV), then the following conditions must be satisfied.

$$0 < t \leq \rho_l, 0 \leq x \leq k-1 \; and \; tx \bmod r \leq \rho_l - t, \tag{5.3.4a}$$

If $j < j'$ (when the neighboring nodes are in quadrant III), then the following conditions must be satisfied.

$$0 < t \leq \rho_l, 0 \leq x \leq k-1 \; and \; r - (tx \bmod r) \leq \rho_l - t, \tag{5.3.4b}$$

So, the number of solutions (t, x) satisfying the above equations (5.3.3a), (5.3.3b), (5.3.4a) and (5.3.4b) give the number of the interior node $n_{i,j}$ within the Lee distance ρ which share key (x, y).

We recall that $|A_{\rho_l}^{(i,j)}|$ is the number of key sharing neighbors of $U_{i,j}$ within the Lee distance ρ_l. Using Lemma 5.2.1 and conditions (5.3.3a), (5.3.3b), (5.3.4a) and (5.3.4b) we arrive at the following theorem.

Theorem 5.3.1. $|A_{\rho_l}^{(i,j)}| = 2\rho_l + 2\sum_{t=1}^{\rho_l - 1}(\sum_{l=0}^{t-1}\Gamma + \sum_{l=1}^{t}\Delta)$ *where,* $\gamma_1 = (rl + 1)/t, \gamma_2 = (rl + \rho_l - t)/t$, $\delta_1 = (rl - \rho_l + t)/t, \delta_2 = (rl - 1)/t$, *and*

$$\Gamma = \begin{cases} 0, & if \; k \leq \lceil \gamma_1 \rceil \leq \lfloor \gamma_2 \rfloor, \\ k - \lceil \gamma_1 \rceil & if \; \lceil \gamma_1 \rceil < k \leq \lfloor \gamma_2 \rfloor, \\ \lfloor \gamma_2 \rfloor - \lceil \gamma_1 \rceil + 1 & if \; \lfloor \gamma_2 \rfloor < k, \end{cases}$$

and

$$\Delta = \begin{cases} 0, & if \; k \leq \lceil \delta_1 \rceil \leq \lfloor \delta_2 \rfloor, \\ k - \lceil \delta_1 \rceil & if \; \lceil \delta_1 \rceil < k \leq \lfloor \delta_2 \rfloor, \\ \lfloor \delta_2 \rfloor - \lceil \delta_1 \rceil + 1 & if \; \lfloor \delta_2 \rfloor < k, \end{cases}$$

Proof. When $x = 0$, $(i-1, 0)$, $(i-2, 0)$, $\cdots$, $(i-\rho_l, 0)$ satisfy (5.3.3a).

For $x \neq 0$, we can map (5.3.3a) to Lemma 5.2.1. Here, $w = r$, $S' = 1$, $S = \rho_l - t$, $T = \rho_l - 1$, $X = k - 1$. So number of solutions satisfying (5.3.3a) is given by $\rho_l + \sum_{t=1}^{\rho_l - 1} \sum_{l=0}^{t-1} \Gamma$. The number of solutions satisfying (5.3.3b) is given by $\sum_{t=1}^{\rho_l - 1} \sum_{l=1}^{t} \Delta$.

Since we are considering the interior node, number of key sharing neighbors in quadrant IV is the same as quadrant I and number of key sharing neighbors in quadrant III is the same as quadrant II. Hence total number of key sharing neighbors of $U_{i,j}$ will be $2\rho_l + 2\sum_{t=1}^{\rho_l - 1}(\sum_{l=0}^{t-1} \Gamma + \sum_{l=1}^{t} \Delta)$. Hence the theorem. ■

Note that the value of $|A_{\rho_l}^{(i,j)}|$ from the above theorem and the value of $|B_{\rho_l}^{(i,j)}|$ from Equation (5.3.1) directly provides the connectivity ratio $p_{c_{\rho_l}}^{(i,j)} = \frac{|A_{\rho_l}^{(i,j)}|}{|B_{\rho_l}^{(i,j)}|}$ for all interior node $U_{i,j}$.

The Table 5.4 compares the connectivity ratio with respect to the Lee sphere and square RF regions for an interior node. Though the connectivity ratio for square RF region is better we can see from the figures that Lee sphere is a better approximation of RF region. The Figure 5.3 presents the connectivity ratio with varying k (the number of keys in each sensor). We see that as the number of keys increases, the connectivity ratio increases.

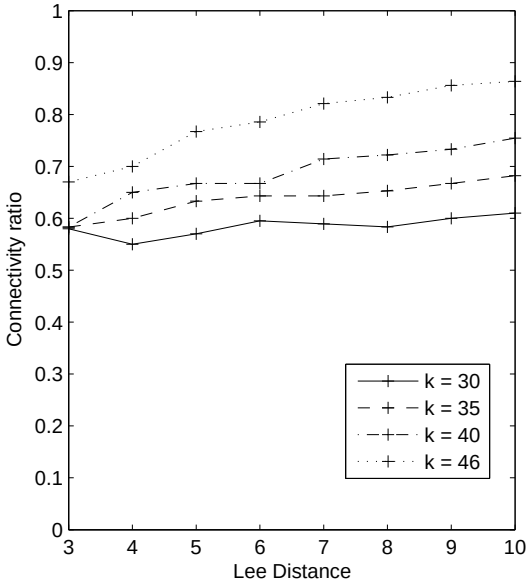


Figure 5.3: Comparison of Connectivity ratio $p_{c_{\rho_l}}$ with changing Lee distance ρ_l and number of keys k on 47×47 grid.

ρ or ρ_l	1	2	3	4	5	6	7	8
$p_{c_{\rho_l}}$	0.5	0.5	0.58	0.55	0.57	0.5952	0.5892	0.5833
p_{c_ρ}	0.5	0.58	0.5833	0.57500	0.60000	0.60714	0.61607	0.61111

Table 5.4: Connectivity Ratio $p_{c_{\rho_l}}$ for interior nodes with change in Lee distance or and square RF region for a 47×47 grid with 30 keys per node.

r	k	ρ_l	s	$E(s)$ (Lee Distance)	$E(s)$ (Square region)
23	15	7	5	0.1990	0.2006
23	15	5	5	0.1981	0.2008
31	20	7	5	0.1526	0.1516
31	25	7	5	0.1528	0.1513
37	30	7	5	0.1289	0.1283
53	49	7	5	0.0913	0.0915
53	49	7	10	0.1756	0.1736

Table 5.5: Experimental value of $E(s)$ and bound for $E(s)$, when number of nodes in the grid is r^2, keys per node is k and the Lee Distance (RF radius) is ρ_l.

5.3.4 Resiliency

We give the experimental values of $E(s)$ in Table 5.5.

5.4 Comparison of our scheme with previous schemes

Key predistribution using deployment knowledge has been studied in [27,49,70,92, 93,149,179]. Huang et. al. presented a grid-group deployment scheme in [70]. The key predistribution was done in a manner similar to Blom's scheme [9]. Huang et. al. assumed that nodes are selective captured whereas in our scheme nodes may be captured randomly. Zu et. al proposed a group based deployment scheme for mobile networks in [179]. According to their scheme sensor nodes were deployed in clusters with mobile agents for inter group communications. The schemes discussed by Du et al [49] and Younis et al [174] use grid-based deployment of sensor nodes

Deployment knowledge is also employed in [92, 93]. In [92, 93], Liu and Ning propose two location based schemes: the closest pairwise keys schemes and one based on symmetric polynomial. In the closest pairwise key scheme a node must share a pairwise key with all its c neighbors. However since the exact position of nodes is not known, the c neighbors found by the setup server may not be the exact neighbors. If c is chosen to be large, then a storage cost will be large. In the symmetric polynomial scheme the setup server generates RC (where $R \times C$ is

the dimension of the field) symmetric polynomials over a finite field K of degree t. For each server the setup server finds the home cell and four adjacent neighbors and distributes to the node its home cell co-ordinate and the polynomial shares of the home cell and that of the four neighbors. The symmetric polynomial scheme has the disadvantage that if the number of compromised nodes is greater that t (where t is the degree of the polynomial) nodes will lead to the collapse of the entire network.

In [149] Simonova, Ling and Wang proposed two grid-group deployment schemes. Though transversal designs have been used, a grid group deployment scheme has been used unlike a grid based scheme as presented by us. Also the resiliency analysis have not been presented in details in their paper.

Grid based deployment has also been studied by Blackburn et al [7, 8]. They use combinatorial designs like Costas arrays and Distinct Difference Configuration for key predistribution. However the detailed resiliency analysis for the different configurations have not been studied so far.

Our design is simple and results in high resiliency in terms of $V(s)$ and $E(s)$ as already mentioned in the previous section. Though the number of groups is chosen to be r^2, where r is a prime power, our design works in all those cases where the dimension n of the grid is not a prime. This can be done by simply choosing a prime power $r > n$ and neglect the regions which fall out of the $n \times n$ grid.

5.5 Concluding remarks

From the geometry of the diagrams we see that Lee Sphere approximation is a better approximation of RF region that a square. In the next chapter we discuss another deployment knowledge based key predistribution scheme. In the present chapter we assumed that all nodes have the same batter power and storage. In the next chapter we assume that there are two distinct types of nodes which differ in battery power and storage.

Chapter 6

Key Predistribution for Grid-group Deployment Scheme

In this chapter a new Grid-group deployment scheme in Wireless Sensor networks. To enhance the resiliency against node compromise, sensor nodes may be deployed in groups in a predetermined way. Several studies have been made where deployment knowledge has been used instead of deploying nodes randomly. Schemes in which deployment knowledge has been used are [47, 49, 69, 70, 92, 93, 149, 174–176, 179].

In a homogeneous key predistribution scheme all sensor nodes have the same storage and computation power. In this paper we propose a heterogeneous key predistribution scheme for a known deployment scheme. Other heterogeneous schemes can be found in [32, 40, 52, 56, 71, 74, 116–118, 181]. In some applications where sensors are scattered over an adversarial area we require that the complete disconnection of one region do not affect another region. For example consider the situation where sensors are deployed in a battlefield. Suppose the adversary captures one region and captures all the sensors. We must ensure that other regions are not affected by such a compromise. For this the whole target region where the sensor nodes are to be deployed is partitioned into equal sized squares or grids as in [92, 93]. There are two types of sensor nodes having different power and storage capacities- the nodes that have lower storage and battery power and agents which are more powerful. It is also assumed that it is more difficult to compromise an agent than a sensor node. The sensors belonging to one region contain a set of keys that are completely disjoint from the sensors in some other region. This ensures that even if one region is totally disconnected, the other regions are not affected. For each sensor node, keys are preloaded in such a way that all the nodes belonging to a particular square region can communicate with each other directly.

Key predistribution for these nodes are done using combinatorial designs called *projective planes*. The key predistribution scheme is given in Section 6.2. It is to be noted that any other combinatorial design could be used like PBIBD as discussed in Chapter 3. Each of the square region also has some specialized nodes called

agents with higher battery power and communication range. Since the regions can be thought of as a grid, we consider the transmission range of each agent (in a region of the grid) to be a *Lee Sphere* of appropriate radius [7]. We call this radius as the *Lee distance*. Any two agents cannot communicate with each other outside the Lee sphere. So all resiliency calculations have been done taking the Lee distance into account. Only three agents are required (whatever be the size of the network) in each region to ensure that a particular region can communicate with all other regions within communication range. Agents have two types of keys. One type of keys is used for communicating within the square region and the other type of keys is used for communicating with agents in different regions. The key predistribution scheme used for inter-region communication makes use of transversal designs. The number of keys preloaded in each sensor node is much less than all existing schemes and nodes are either directly connected or connected via two hop paths. The deterministic key predistribution schemes result in constant time computation overhead for shared key discovery and path key establishment.

We show that our scheme is resilient to selective node capture attack. We measure the resiliency in terms of the fraction of links compromised and also the fraction of nodes and regions disconnected. We obtain very high resiliency (in terms of fraction of links compromised) compared to the schemes in [27, 47, 49, 69, 70, 92, 93, 149, 174–176, 179]. We also get a very good resiliency in terms of the fraction of nodes disconnected and regions disconnected. For example, for a 31×31 grid, if nodes in each region contains 18 keys ($p = 17$), then on an average 17298 nodes must be compromised to disconnect one node of each region. We note that for a 31×31 grid if 200 agents are compromised then about $2 - 3$ regions will be disconnected, where each agent contains 20 keys. If agents contain 25 keys then on compromising 200 agents only about one region will be disconnected.

We use combinatorial structures like projective planes and transversal designs. The deterministic designs help in the estimation of resiliency. Direct communication is assured between any two nodes in the same region.

The rest of this chapter is organized in the following way. In Section 6.1, we present some notations. In Section 6.2, we present grid-group deployment scheme and predistribution scheme. In Section 6.3, we study the resiliency of the network. We give two parameters for security. In Section 6.4, we make a comparison of our scheme with other schemes. This chapter is the outcome of the paper [138].

6.1 Notations

Consider a square grid (as shown in figure 6.1). We recall from the previous chapter that a *Lee Sphere* [7] of radius ρ_l centered at a given square consists of the set of the squares that lie at a distance of at most ρ_l from the square. ρ_l is called the Lee distance.

Table 6.1 represent the notations we have used throughout the chapter.

r	Dimension of the grid, r is a prime power
N	Number of agents in the network
$S_{i,j}$	(i,j)th region
$P_{i,j}$	Set of all keys assigned to nodes in the $S_{i,j}$th region
$p+1$	Number of keys in each node
p^2+p+1	Maximum number of sensor nodes in each region
k	Number of Type I, Type II or Type III keys present in each agent
$B^1_{i,j}$	Set of keys assigned to agent of Type I in the region $S_{i,j}$
$B^2_{i,j}$	Set of keys assigned to agent of Type II in the region $S_{i,j}$
$B^3_{i,j}$	Set of keys assigned to agent of Type III in the region $S_{i,j}$
ρ_l	Lee distance
$L(x,y)$	Interlinks connected by key (x,y), $x,y<r$
$E'(s)$	Number of intralinks broken when s nodes are compromised
$E''(s)$	Number of interlinks broken when s agents are compromised
$V'(s)$	Number of nodes disconnected when s nodes are compromised
$V''(s)$	Number of agents disconnected when s agents are compromised

Table 6.1: Notations

6.2 Key predistribution scheme

We first discuss the deployment architecture and then present our key predistribution scheme. When nodes are deployed in a region, all nodes need not communicate with all other nodes in the network. Due to limited power, all nodes cannot communicate with all other nodes. So we divide the entire region into equal sized squares or grids as done in [69, 92, 93]. Let us consider an $r \times r$ deployment area, consisting of r^2 regions (r is a prime power). The r^2 regions are numbered as $S_{i,j}$, $0 \leq i,j < r$ as shown in Figure 6.1. Though U_i and U_j are not within Manhattan distance 2, but for simplicity U_j is considered to be within Lee distance 2 from U_i. All nodes within a particular square region communicate with each other. However to communicate with nodes in other regions, there are specialized nodes called *agents*. These are more powerful than the nodes and have higher storage capacities. Let each agent have a transmission range of ρ_l. Then each agent can communicate directly with agents which lie within the circle of radius ρ_l and center as the agent itself and shares keys with the agent. When we consider communication between regions we consider the communication region as the Lee sphere (defined in Section 5.1) of appropriate radius [7]. Though the Lee sphere does not exactly depict the agents within Manhattan distance from a particular agent, for simplicity we consider the Lee spheres around an agent in all our calculations. Each region contains a set of three agents irrespective of the number of regions. In general the number of agents must be proportional to the number of regions. However three and only three agents are enough to ensure that a re-

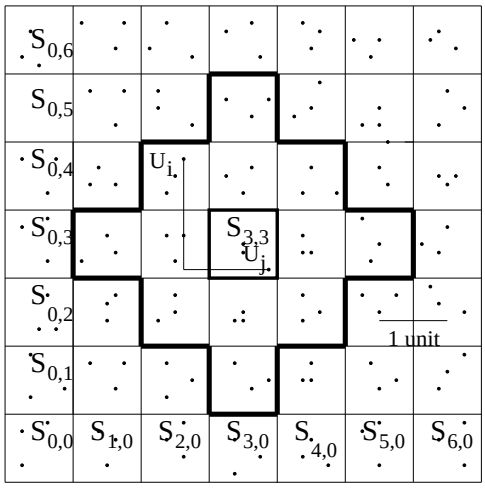


Figure 6.1: A deployment region having 49 regions. Lee distance is 2. Each region has three agents. Region $S_{3,3}$ is connected to all the regions within the marked Lee sphere.

gion can communicate with all regions within Lee distance. This is formalized in Theorem 6.2.1. Throughout the paper we refer to any small node or an agent as a *sensor node*.

6.2.1 Key predistribution in nodes in a region

For each region keys are predistributed in the nodes independently of the other region using some existing predistribution scheme. We use deterministic design because we would like to ensure that all nodes within a region can communicate with each other directly. Probabilistic designs cannot guarantee this fact. Since deterministic designs have a pattern, shared-key discovery and path key establishment is efficient [89]. We choose the symmetric design as given in [18, 20]. Any other combinatorial design which ensures direct communication can be employed like that given in Section 3. Each of the smaller regions consists of p^2+p-2 nodes each containing $p+1$ keys, where p is a prime power. We do not use the other designs using generalized quadrangles given in [18]. Though these designs result in large network size (of the order of 3 in the number of keys), the connectivity is very low. In case the number of sensors is not of the form p^2+p+1 for some prime power p, then we choose p such that $n \leq p^2+p+1$ and distribute keys to only n sensors. So first n is decided upon. Based on the value of n the parameter p is chosen. A maximum of $r^2(p^2+p+1)$ sensor nodes (nodes and agents) can be supported. If $r = 23$, $p = 17$, then 162403 sensor nodes can be supported. We distribute the keys according to the Algorithm 1 given in [18]. Let us denote the set of keys assigned to nodes in the region $S_{i,j}$ by $P_{i,j}$. Each of the regions have a distinct set of p^2+p+1 keys. So the entire size of the key pool is $r^2(p^2+p+1)$. Since $P_{i,j} \bigcap P_{i',j'} = 0$, for $(i,j) \neq (i',j')$, it can be ensured that even if a few nodes

(or all nodes) within a region are compromised, none of the nodes (or link between nodes) in the other regions are affected. If two nodes share a common key then an *intra link* is said to exist between the nodes.

We will see in the next chapter that the shared key discovery algorithm runs in constant time.

6.2.2 Key predistribution in agents over the entire network

For any square region $S_{i,j}$ a set of three agents $a^1_{i,j}$, $a^2_{i,j}$ and $a^3_{i,j}$ are deployed. The set of $k+p+1$ keys assigned to the three agents are denoted by $B^1_{i,j}$, $B^2_{i,j}$ and $B^3_{i,j}$. Apart from the $p+1$ keys assigned from the set $P_{i,j}$, $B^1_{i,j}$ contains $\{(x,(xi+j) \bmod r) : 0 \leq x < k\}$, $B^2_{i,j}$ contains $\{(x, r+((j-xi) \bmod r)) : 0 \leq x < k\}$ and $B^3_{i,j}$ contains $\{(x, 2r+((xj+i) \bmod r)) : 0 \leq x < k\}$.

Consider the keys of the form (x,y), where $0 \leq x < k$. If $y < r$, then the keys are called Type I keys, if $r \leq y < 2r$, then the keys are called Type II keys, and if $y \geq 2r$, then the keys are called Type III keys. Refer to Table 6.3. We note that any agent contains keys of only one type. Depending on the type of keys an agent contains, agents may be of Type I, Type II or Type III.

The agents can communicate with each other if they are within a fixed communication range. Generally this region around a given sensor node is the circular region with radius ρ called the RF radius and center as the sensor node itself. For simplicity we assume that agents within a particular region can communicate with agents which lie inside Lee sphere (with Lee distance ρ_l) of that region (Lee sphere is defined earlier in Section 5.1). For this reason it is equivalent if we consider the three agents to be placed at the center of the region.

A natural question arises; why not place one agent instead of three. The answer is that when nodes are compromised randomly, the probability that only one node is compromised is more than when all the nodes are compromised. Had there been one agent in a region, two agents would have shared more than one key. In such a case a common key would have to be selected for communication. Now if this key is compromised, the shared key algorithm would have to be executed again to find a new common key. Since there are three agents and any two agent share at most one common key, if some key is compromised, the link is broken, there is no need of executing the shared key discovery algorithm again.

We could also have predistributed keys in such a way that nodes in two regions share some common keys. However the compromise of one region adversely affects the other regions. Hence though economical this method is not proposed.

If two regions within Lee distance communicate via some common key belonging to some agent, then a *interlink* is said to exist. Each agent $B^l_{i,j}$ has an identifier (i,j,l) denoting the region $S_{i,j}$ where it belongs and l denoting the type of agent. To carry out secure communication, a common key needs to be established. Since we use a deterministic design having some definite structure, shared key discovery becomes very simple. Two agents of different type do not have any common key.

However if two agents of the same type (say Type I) want to find out a shared key (if there exists one), then they broadcast their identifier. The common key will be obtained very easily using only an inverse calculation.

6.2.3 Shared-key discovery

Suppose two Type I agents belonging to regions (i, j) and (i', j') want to find the shared key (if it exists). Then the shared key (x, y) will be such that $xi + j = xi' + j' \bmod r$. So if $x = (j' - j)(i - i')^{-1} < k$, then a common key exists. Suppose two Type II agents belonging to regions (i, j) and (i', j') want to find the shared key (if it exists). Then the shared key (x, y) will be such that $j - xi = j' - xi' \bmod r$. So if $x = (j' - j)(i' - i)^{-1} < k$, then a common key exists. Similarly to find if two Type II agents belonging to regions (i, j) and (i', j') want to find the shared key (if it exists), then the shared key (x, y) will be such that $xj + i = xj' + i' \bmod r$. So if $x = (i' - i)(j - j')^{-1} < k$, then a common key exists. The algorithm for shared key discovery runs in $O(1)$ time and only the identifier of the agent has to be sent. This results in a communication overhead of $O(\log r)$ bits. Since randomized key predistribution result in high communication and computation complexity, as discussed in [89, Section 2.2.6], our approach is better than probabilistic key predistribution like [69].

This can be found in constant time. If no common key exists, then a path key can be found using the technique given below.

Suppose node u belonging to region S_1 wants to communicate with node v belonging to region S_2, such that S_1 and S_2 are within the Lee distance each other. u generates a random key K and finds a common key say k_1 it shares with one of the agents a_1 in S_1. It then sends the key K encrypted by k_1 to a_1. a_1 decrypts K using k_1. If a_1 shares a common key k_2 with agent a_2 of region S_2, then a_1 encrypts the key K with k_2 and sends it to a_2. a_2 then decrypts K using k_2 and sends it to v encrypted with the shared key k_3 between a_2 and v. v then decrypts K using k_3. So k is now a shared key (path key) between u and v. This is depicted in Figure 6.2. We next show that if $k \geq (r+1)/2$, then any two regions within Lee distance are connected.

Theorem 6.2.1. *If $k \geq (r + 1)/2$, then any two regions within Lee distance is connected via one or more common keys belonging to one or more agents.*

Proof. Let us consider two regions $S_{i,j}$ and $S_{i',j'}$. If $B^1_{i,j} \bigcap B^1_{i',j'} \neq 0$, then $S_{i,j}$ is connected to $S_{i',j'}$. Suppose $B^1_{i,j} \bigcap B^1_{i',j'} = 0$, then for all $x = (j' - j)(i - i')^{-1} \bmod r \geq k$. Since $k \geq (r + 1)/2$, $(j' - j)(i' - i)^{-1} < k - 1$. For some x' if $j - x'i = j' - x'i' \bmod r$, then $x' < k - 1$. So $B^2_{i,j}$ is connected to $B^2_{i',j'}$. If $i = i'$ then regions are not connected via Type I or Type II keys. However all nodes where $i = i'$, are connected via Type III keys, because they share the key $(0, 2r + i)$. Hence all regions within the Lee distance of (i, j) are connected to it. ■

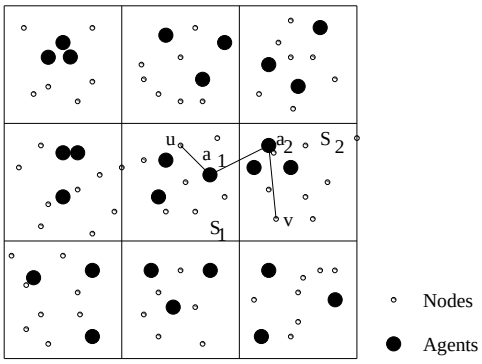


Figure 6.2: A deployment region with nodes and agents. Communication between two nodes via agents is shown.

The Theorem 6.2.1 provides the justification of choosing only three agents for each region whatever be the size of the network.

Throughout the chapter we consider r to be prime power and $k \geq (r+1)/2$.

6.3 Analysis of resiliency

We consider selective node capture and random node capture models. We show that selective node capture model cannot be mounted on our scheme.

6.3.1 Resiliency against selective node capture

During selective node capture the attacker compromises those nodes whose keys have not already been compromised. We note that any two nodes broadcast their node identifiers during the shared-key discovery phase. The key identifiers are not broadcasted. Thus at no stage the attacker can know what key identifier is present in which node. Thus there is no way of knowing which nodes are left to be compromised. Thus unless the attacker compromises the node, she cannot choose a node for compromise to maximize the number of keys compromised. Hence our scheme is secure against selective node capture.

6.3.2 Random node compromise

We give measures of resiliency $E'(s)$, $E''(s)$, $V'(s)$ and $V''(s)$ as the proportion of links and nodes being broken respectively. We consider two types of resiliency - Local resiliency which denotes the fraction of intra links or nodes affected within a region and global resiliency which denotes the fraction of interlinks and agents affected. Mathematically,

$$E'(s) = \frac{\text{Number of intralinks broken after } s \text{ nodes are compromised}}{\text{Number of intralinks present before compromised}}$$

$$E''(s) = \frac{\text{Number of interlinks broken after } s \text{ agents are compromised}}{\text{Number of interlinks present before compromised}}$$

$$V'(s) = \frac{\text{Number of nodes disconnected after } s \text{ nodes are compromised}}{\text{Number of nodes present before compromised}}$$

and

$$V''(s) = \frac{\text{Number of regions disconnected after } s \text{ agents are compromised}}{\text{Number of regions present before compromised}}$$

We consider the local and global resiliency. By local resiliency we mean the resiliency within a particular region. This is measured in terms of $E'(s)$ (defined as the fraction of intralinks affected when s nodes are compromised) and $V'(s)$ (defined as the fraction of nodes disconnected when s nodes are compromised). By global resiliency we mean the resiliency of the entire region. This is measured in terms of $E''(s)$ (defined as the fraction of interlinks affected when s nodes are compromised) and $V''(s)$ (defined as the fraction of regions disconnected when s nodes are compromised). We give experimental and theoretical results for $E'(s)$, $E''(s)$, $V'(s)$ and $V''(s)$.

6.3.3 Estimation of $E'(s)$ and $E''(s)$

Let s sensor nodes be randomly compromised. Let s' nodes be compromised and s'' agents be compromised. We first find local resiliency $E'(s')$ (fraction of intra links broken when s' nodes are compromised) and then calculate the global resiliency $E''(s'')$ (fraction of inter links broken when s'' agents are compromised).

6.3.3.1 Estimation of local resiliency $E'(s')$ for intra links

According to the predistribution scheme any key within a particular region is present in exactly $p+1$ sensor nodes, including the agents. Also, any two nodes have only one common key. So if a key K is compromised, then $p(p+1)/2$ links are broken. Let us assume that k_i distinct keys are compromised from the S_i th region. Then $pk_i(p+1)/2$ links are broken. Let us assume s_i nodes belonging to S_ith region are broken. We assume that all the keys exposed are distinct. This is an overestimate, because two compromised nodes may have a common key. So a maximum of $(p+1)s_i$ keys are exposed. So a maximum of $s_i p(p+1)^2/2$ intra links are broken. Since there are only three agents, there are a maximum of six links broken between between the agents. We can ignore this. So the fraction of links broken when s_i nodes are compromised within region S_i is less than $\frac{s_i p(p+1)^2/2}{(p^2+p+1)(p^2+p)/2} = \frac{(p+1)s_i}{p^2+p+1}$. Suppose a total of s' nodes are compromised. These belong to the r^2 regions. Assuming that the compromised nodes are evenly

r	p	$n = p^2 + p + 1$	s'	$E'(s')$ (Experimental)	$E'(s')$ (Theoretical Upper Bound)
23	17	307	700	0.0336	0.0775
31	17	307	900	0.0534	0.0549
37	19	381	1200	0.0405	0.0460
37	23	553	2000	0.0616	0.0634
47	19	381	2000	0.0465	0.0475
47	23	553	2500	0.0680	0.0885

Table 6.2: Experimental value Vs Theoretical value of $E'(s')$ when number of nodes in a region is $n = p^2 + p + 1$, keys per node is $p + 1$ and s' nodes are compromised.

distributed in the region, s'/r^2 nodes are nodes are broken per region. Hence the resiliency $E'(s') = \frac{1}{r^2}\sum_{i=0}^{r^2-1}\frac{(p+1)s_i}{p^2+p+1} = \frac{s'(p+1)}{r^2(p^2+p+1)}$. The Table 6.2 gives the experimental and theoretical estimates of $E'(s')$. Experimental results are obtained for s' nodes chosen randomly over 100 runs.

6.3.3.2 Estimation of global resiliency $E''(s'')$ for interlinks

We give an outline of how to compute the number of links broken when agents are compromised. Number of interlinks connected to each interior region is $2\rho_l(\rho_l+1)$. The initial number of interlinks is less than $2r^2\rho_l(\rho_l + 1)$. (since the regions near the periphery will be connected to less number of regions). Let s'' agents be compromised. Any two regions in the same column are connected by Type III keys. Also any two regions not in the same row are connected by Type I and Type II keys. Any two regions share either one, two or three keys of different types. Refer to Table 6.3. Since regions may be connected by more than one keys, only if all the shared keys are compromised, the interlink is disrupted. We say that the triple $\{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$ is a *good triple*, with (x_1, y_1) a Type I key, (x_2, y_2) a Type II key, (x_3, y_3) a Type III key, if whenever any one of the keys (x_i, y_i) occurs in an agent in a region, then (x_j, y_j), $(j \in \{1, 2, 3\}$ and $j \neq i)$ also occurs in some other agents in the same region. For example $\{(4, 4), (3, 11), (2, 15)\}$ is a good triple. Similarly, we say that the pair of keys $\{(x_1, y_1), (x_2, y_2)\}$ is a *good pair* of Type I - Type II, with (x_1, y_1) a Type I key, (x_2, y_2) a Type II key, if whenever any one of the keys (x_i, y_i) occurs in a region, then (x_j, y_j), $(j \in \{1, 2\}$ and $j \neq i)$ also occurs in some agents in the same region but there is no key (x_3, y_3) of Type III such that $\{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$ is not a good triple. Similarly we can define good pairs for Type II - Type III and Type I -Type III keys. For example, $\{(2, 3), (4, 19)\}$ is a good pair. However $\{(4, 4), (3, 11)\}$ is not a good pair because, $\{(4, 4), (3, 11), (2, 15)\}$ is a good triple. If there is a Type I or Type II or Type III key such that it does not form a good pair or good triple with some other type of

key, then it is called an isolate. There are three types of isolates - Type I isolate, Type II isolate and Type III isolate. For example $(1, 7)$ is a Type II isolate in the example given in Table 6.3. There are no Type I isolates. We next find the conditions for existence of good triple or good pair. Let us consider two regions (i, j) and (i', j'). A good triple exists if all the three conditions below are satisfied.

$$x_1 i + j = x_1 i' + j' (\mathrm{mod} r) \tag{6.3.1a}$$

$$-x_2 i + j = -x_2 i' + j' (\mathrm{mod} r) \tag{6.3.1b}$$

$$x_3 j + i = x_3 j' + i' (\mathrm{mod} r) \tag{6.3.1c}$$

From equations 6.3.1a and 6.3.1b we find that $x_1 = -x_2 (\mathrm{mod} r)$. Similarly, from equations 6.3.1a and 6.3.1c we find that $x_1 = x_3^{-1} (\mathrm{mod} r)$ and from equations 6.3.1b and 6.3.1c we find that $x_2 = -x_3^{-1} (\bmod r)$. Hence a good triple exists for all $x_1, x_2, x_3 < k$, such that $x_1 = -x_2 = x_3^{-1}$. The good triple that arises when this condition holds is $\{(x_1, y_1), (-x_1, r + y_1), (x_1^{-1}, 2r + (x_1^{-1} y_1) \mod r))\}$. Similarly for good pairs consisting of Type I and Type II keys we have $x_1 = -x_2$, $x_1, x_2 < k$ and $x_1^{-1} \geq k$ and the good pair is given by $\{(x_1, y_1), (-x_1, r + y_1)\}$. For good pairs consisting of Type I and Type III keys we have $x_1 = x_3^{-1}$, $x_1, x_3 < k$ and $-x_1 \geq k$ and good pair of Type I-Type III is given by $\{(x_1, y_1), (x_1^{-1}, 2r + (x_1^{-1} y_1) \mod r))\}$. Similarly for good pairs consisting of Type II and Type III keys we have $x_2 = -x_3^{-1}$, $x_2, x_3 < k$ and $-x_2 \geq k$ and good pair of Type II-Type III is given by $\{(x_2, y_2), (-x_2^{-1}, 2r + (-x_2^{-1} y_2) \mod r))\}$.

Suppose s'' agents are compromised. We calculate the number of interlinks broken. We find all the distinct keys that are exposed. Generally when s'' agents are compromised, not all $3ks''$ keys are distinct, because two compromised agents may contain some common keys. Once we know the keys, we find the good triples and good pairs and isolates which are compromised. Then the number of interlinks broken within Lee distance ρ_l will be the number of interlinks connected by the compromised good triples, good pairs and isolates. We explain this in the Section 6.3.4.

6.3.4 Estimation of the number of links disrupted when s'' agents are compromised

Suppose s'' agents are compromised. The compromised agents may be only of one type, Type I, Type II or Type III or may be a combination of these types. We enumerate the different conditions that can arise and how the links may be affected due to these conditions. It can be noted that not all exposed keys are distinct. We also note that only agents of the same type can have keys in common. For example, agent of Type I can share at most one key with another agent of Type I, but none of Type II or Type III. Given an existing interlink we can say if it will be disconnected if all the shared keys in all the agents are exposed. There may be one, two or three shared keys between an interlink. Only if all the shared

Region	Type I					Type II					Type III				
$S_{0,0}$:	(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(0,7)	(1,7)	(2,7)	(3,7)	(4,7)	(0,14)	(1,14)	(2,14)	(3,14)	(4,14)
$S_{0,1}$:	(0,1)	(1,1)	(2,1)	(3,1)	(4,1)	(0,8)	(1,8)	(2,8)	(3,8)	(4,8)	(0,14)	(1,15)	(2,16)	(3,17)	(4,18)
$S_{0,2}$:	(0,2)	(1,2)	(2,2)	(3,2)	(4,2)	(0,9)	(1,9)	(2,9)	(3,9)	(4,9)	(0,14)	(1,16)	(2,18)	(3,20)	(4,15)
$S_{0,3}$:	(0,3)	(1,3)	(2,3)	(3,3)	(4,3)	(0,10)	(1,10)	(2,10)	(3,10)	(4,10)	(0,14)	(1,17)	(2,20)	(3,16)	(4,19)
$S_{0,4}$:	(0,4)	(1,4)	(2,4)	(3,4)	(4,4)	(0,11)	(1,11)	(2,11)	(3,11)	(4,11)	(0,14)	(1,18)	(2,15)	(3,19)	(4,16)
$S_{0,5}$:	(0,5)	(1,5)	(2,5)	(3,5)	(4,5)	(0,12)	(1,12)	(2,12)	(3,12)	(4,12)	(0,14)	(1,19)	(2,17)	(3,15)	(4,20)
$S_{0,6}$:	(0,6)	(1,6)	(2,6)	(3,6)	(4,6)	(0,13)	(1,13)	(2,13)	(3,13)	(4,13)	(0,14)	(1,20)	(2,19)	(3,18)	(4,17)
$S_{1,0}$:	(0,0)	(1,1)	(2,2)	(3,3)	(4,4)	(0,7)	(1,13)	(2,12)	(3,11)	(4,10)	(0,15)	(1,15)	(2,15)	(3,15)	(4,15)
$S_{1,1}$:	(0,1)	(1,2)	(2,3)	(3,4)	(4,5)	(0,8)	(1,7)	(2,13)	(3,12)	(4,11)	(0,15)	(1,16)	(2,17)	(3,18)	(4,19)
$S_{1,2}$:	(0,2)	(1,3)	(2,4)	(3,5)	(4,6)	(0,9)	(1,8)	(2,7)	(3,13)	(4,12)	(0,15)	(1,17)	(2,19)	(3,14)	(4,16)
$S_{1,3}$:	(0,3)	(1,4)	(2,5)	(3,6)	(4,0)	(0,10)	(1,9)	(2,8)	(3,7)	(4,13)	(0,15)	(1,18)	(2,14)	(3,17)	(4,20)
$S_{1,4}$:	(0,4)	(1,5)	(2,6)	(3,0)	(4,1)	(0,11)	(1,10)	(2,9)	(3,8)	(4,7)	(0,15)	(1,19)	(2,16)	(3,16)	(4,17)
$S_{1,5}$:	(0,5)	(1,6)	(2,0)	(3,1)	(4,2)	(0,12)	(1,11)	(2,10)	(3,9)	(4,8)	(0,15)	(1,20)	(2,18)	(3,18)	(4,14)
$S_{1,6}$:	(0,6)	(1,7)	(2,1)	(3,2)	(4,3)	(0,13)	(1,12)	(2,11)	(3,10)	(4,9)	(0,15)	(1,14)	(2,20)	(3,20)	(4,18)
$S_{2,0}$:	(0,0)	(1,2)	(2,4)	(3,6)	(4,1)	(0,7)	(1,12)	(2,10)	(3,8)	(4,13)	(0,16)	(1,16)	(2,16)	(3,16)	(4,16)
$S_{2,1}$:	(0,1)	(1,3)	(2,5)	(3,0)	(4,2)	(0,8)	(1,13)	(2,11)	(3,9)	(4,7)	(0,16)	(1,17)	(2,18)	(3,19)	(4,20)
$S_{2,2}$:	(0,2)	(1,4)	(2,6)	(3,1)	(4,3)	(0,9)	(1,7)	(2,12)	(3,10)	(4,8)	(0,16)	(1,18)	(2,20)	(3,15)	(4,17)
$S_{2,3}$:	(0,3)	(1,5)	(2,0)	(3,2)	(4,4)	(0,10)	(1,8)	(2,13)	(3,11)	(4,9)	(0,16)	(1,19)	(2,15)	(3,18)	(4,14)
$S_{2,4}$:	(0,4)	(1,6)	(2,1)	(3,3)	(4,5)	(0,11)	(1,9)	(2,7)	(3,12)	(4,10)	(0,16)	(1,20)	(2,17)	(3,14)	(4,18)
$S_{2,5}$:	(0,5)	(1,0)	(2,2)	(3,4)	(4,6)	(0,12)	(1,10)	(2,8)	(3,13)	(4,11)	(0,16)	(1,14)	(2,19)	(3,17)	(4,15)
$S_{2,6}$:	(0,6)	(1,1)	(2,3)	(3,5)	(4,0)	(0,13)	(1,11)	(2,9)	(3,7)	(4,12)	(0,16)	(1,15)	(2,14)	(3,19)	(4,19)
$S_{3,0}$:	(0,0)	(1,3)	(2,6)	(3,2)	(4,5)	(0,7)	(1,11)	(2,8)	(3,12)	(4,9)	(0,17)	(1,17)	(2,17)	(3,17)	(4,17)
$S_{3,1}$:	(0,1)	(1,4)	(2,0)	(3,3)	(4,6)	(0,8)	(1,12)	(2,9)	(3,13)	(4,10)	(0,17)	(1,18)	(2,19)	(3,20)	(4,14)
$S_{3,2}$:	(0,2)	(1,5)	(2,1)	(3,4)	(4,0)	(0,9)	(1,13)	(2,10)	(3,7)	(4,11)	(0,17)	(1,19)	(2,14)	(3,16)	(4,18)
$S_{3,3}$:	(0,3)	(1,6)	(2,2)	(3,5)	(4,1)	(0,10)	(1,7)	(2,11)	(3,8)	(4,12)	(0,17)	(1,20)	(2,16)	(3,19)	(4,15)
$S_{3,4}$:	(0,4)	(1,0)	(2,3)	(3,6)	(4,2)	(0,11)	(1,8)	(2,12)	(3,9)	(4,13)	(0,17)	(1,14)	(2,18)	(3,15)	(4,19)
$S_{3,5}$:	(0,5)	(1,1)	(2,4)	(3,0)	(4,3)	(0,12)	(1,9)	(2,13)	(3,10)	(4,7)	(0,17)	(1,15)	(2,20)	(3,18)	(4,16)
$S_{3,6}$:	(0,6)	(1,2)	(2,5)	(3,1)	(4,4)	(0,13)	(1,10)	(2,7)	(3,11)	(4,8)	(0,17)	(1,26)	(2,15)	(3,14)	(4,20)
$S_{4,0}$:	(0,0)	(1,4)	(2,1)	(3,5)	(4,2)	(0,7)	(1,10)	(2,13)	(3,9)	(4,12)	(0,18)	(1,18)	(2,18)	(3,18)	(4,18)
$S_{4,1}$:	(0,1)	(1,5)	(2,2)	(3,6)	(4,3)	(0,8)	(1,11)	(2,7)	(3,10)	(4,13)	(0,18)	(1,19)	(2,20)	(3,14)	(4,15)
$S_{4,2}$:	(0,2)	(1,6)	(2,3)	(3,0)	(4,4)	(0,9)	(1,12)	(2,8)	(3,11)	(4,7)	(0,18)	(1,20)	(2,15)	(3,17)	(4,19)
$S_{4,3}$:	(0,3)	(1,0)	(2,4)	(3,1)	(4,5)	(0,10)	(1,13)	(2,9)	(3,12)	(4,8)	(0,18)	(1,14)	(2,17)	(3,20)	(4,16)
$S_{4,4}$:	(0,4)	(1,1)	(2,5)	(3,2)	(4,6)	(0,11)	(1,7)	(2,10)	(3,13)	(4,9)	(0,18)	(1,15)	(2,19)	(3,16)	(4,20)
$S_{4,5}$:	(0,5)	(1,2)	(2,6)	(3,3)	(4,0)	(0,12)	(1,8)	(2,11)	(3,7)	(4,10)	(0,18)	(1,16)	(2,14)	(3,19)	(4,17)
$S_{4,6}$:	(0,6)	(1,3)	(2,0)	(3,4)	(4,1)	(0,13)	(1,9)	(2,12)	(3,8)	(4,11)	(0,18)	(1,17)	(2,16)	(3,15)	(4,14)
$S_{5,0}$:	(0,0)	(1,5)	(2,3)	(3,1)	(4,6)	(0,7)	(1,9)	(2,11)	(3,13)	(4,8)	(0,19)	(1,19)	(2,19)	(3,19)	(4,19)
$S_{5,1}$:	(0,1)	(1,6)	(2,4)	(3,2)	(4,0)	(0,8)	(1,10)	(2,12)	(3,7)	(4,9)	(0,19)	(1,20)	(2,14)	(3,15)	(4,16)
$S_{5,2}$:	(0,2)	(1,0)	(2,5)	(3,3)	(4,1)	(0,9)	(1,11)	(2,13)	(3,8)	(4,10)	(0,19)	(1,14)	(2,16)	(3,18)	(4,20)
$S_{5,3}$:	(0,3)	(1,1)	(2,6)	(3,4)	(4,2)	(0,10)	(1,12)	(2,7)	(3,9)	(4,11)	(0,19)	(1,15)	(2,18)	(3,14)	(4,17)
$S_{5,4}$:	(0,4)	(1,2)	(2,0)	(3,5)	(4,3)	(0,11)	(1,13)	(2,8)	(3,10)	(4,12)	(0,19)	(1,16)	(2,20)	(3,17)	(4,14)
$S_{5,5}$:	(0,5)	(1,3)	(2,1)	(3,6)	(4,4)	(0,12)	(1,7)	(2,9)	(3,11)	(4,13)	(0,19)	(1,17)	(2,15)	(3,20)	(4,18)
$S_{5,6}$:	(0,6)	(1,4)	(2,2)	(3,0)	(4,5)	(0,13)	(1,8)	(2,10)	(3,12)	(4,7)	(0,19)	(1,18)	(2,17)	(3,16)	(4,15)
$S_{6,0}$:	(0,0)	(1,6)	(2,5)	(3,4)	(4,3)	(0,7)	(1,8)	(2,9)	(3,10)	(4,11)	(0,20)	(1,20)	(2,20)	(3,20)	(4,20)
$S_{6,1}$:	(0,1)	(1,0)	(2,6)	(3,5)	(4,4)	(0,8)	(1,9)	(2,10)	(3,11)	(4,12)	(0,20)	(1,14)	(2,15)	(3,16)	(4,17)
$S_{6,2}$:	(0,2)	(1,1)	(2,0)	(3,6)	(4,5)	(0,9)	(1,10)	(2,11)	(3,12)	(4,13)	(0,20)	(1,15)	(2,17)	(3,19)	(4,14)
$S_{6,3}$:	(0,3)	(1,2)	(2,1)	(3,0)	(4,6)	(0,10)	(1,11)	(2,12)	(3,13)	(4,7)	(0,20)	(1,16)	(2,19)	(3,15)	(4,18)
$S_{6,4}$:	(0,4)	(1,3)	(2,2)	(3,1)	(4,0)	(0,11)	(1,12)	(2,13)	(3,7)	(4,8)	(0,20)	(1,17)	(2,14)	(3,18)	(4,15)
$S_{6,5}$:	(0,5)	(1,4)	(2,3)	(3,2)	(4,1)	(0,12)	(1,13)	(2,7)	(3,8)	(4,9)	(0,20)	(1,18)	(2,16)	(3,14)	(4,19)
$S_{6,6}$:	(0,6)	(1,5)	(2,4)	(3,3)	(4,2)	(0,13)	(1,7)	(2,8)	(3,9)	(4,10)	(0,20)	(1,19)	(2,18)	(3,17)	(4,16)

Table 6.3: A 7×7 region with 5 keys in each agent.

keys are exposed, the link is broken. For example regions $S_{1,3}$ and $S_{3,2}$ have three keys $(4,0)$, $(3,7)$ and $(2,14)$ in common. So if all three keys are compromised, then the link $S_{1,3}S_{3,2}$ will be broken, provided they are within communication range. Again we note that $\{(4,0), (3,7), (2,14)\}$ is a good triple. So whenever agents are compromised, we check if good triples and good pairs are compromised. Then the interlinks connected by these triples and pairs will be broken. Also if there are isolates which are exposed, then interlinks which are connected by these isolates are broken. We illustrate these different cases with the example given in Table 6.3. $L(x,y)$ denotes the number of interlinks within communication range that are connected by key (x,y).

1. If all the compromised agents are of Type I, then number of links broken is given by

   $$\sum_{(x,y)} L(x,y) \text{ , where } (x,y) \text{ is a compromised isolate of Type I.}$$

   Since Type II and Type III keys are not compromised, no good triple or good pair is compromised. Only the interlinks which are connected via compromised Type I isolates will be broken. In the example given, since there are no isolates of Type I, no links will be compromised if only Type I agents are compromised. All interlinks will be connected by Type II or Type III keys.

2. If all the compromised agents are of Type II, then number of links broken is given by

   $$\sum_{(x,y)} L(x,y-r) \text{ , where } (x,y) \text{ is a compromised isolate of Type II.}$$

   The reason is same as Case 1. For example, if Type II agents in $S_{2,1}$ and $S_{4,5}$ are compromised, then the keys $(0,8)$, $(1,13)$, $(2,11)$, $(3,9)$, $(4,7)$, $(0,12)$, $(1,8)$, $(3,7)$ and $(4,10)$ are exposed. Only $(1,13)$ and $(1,8)$ are isolates of Type II and all the interlinks within communication range which are connected by these two keys will be broken. No interlinks connected by $(0,8)$ will not be broken because $\{(0,8),(0,1)\}$ is a good pair and $(0,1)$ is not compromised. So all interlinks which were connected via $(0,8)$ will still be connected by $(0,1)$. Similar is the case with other exposed keys.

3. If all the compromised agents are of Type III, then number of inter links broken is given by

   $$\sum_{(x,y)} L(x,y-2r) \text{ , where } (x,y) \text{ is a compromised isolate Type of III.}$$

   The reason is same as Case 1. For example, if Type III agents in $S_{1,6}$ and $S_{4,5}$ are compromised, then the keys $(0,15)$, $(1,14)$, $(2,20)$, $(3,19)$, $(4,18)$, $(0,18)$, $(1,16)$, $(2,14)$ and $(4,17)$ are exposed. Only $(0,15)$ and $(0,18)$ are

isolates of Type III and all the interlinks within communication range which are connected by these two keys will be broken. No other interlinks will be broken since good pairs and good triples exists which are not fully exposed.

4. If all the compromised agents are of Type I or Type II, then number of interlinks broken is given by

$$\begin{gathered}\sum_{(x,y)} L(x,y), \text{ where } \{(x,y),(-x,r+y)\} \text{ is a compromised good pair } + \\ \sum_{(x,y)} L(x,y), \text{ where } (x,y) \text{ is a Type I compromised isolate } + \\ \sum_{(x,y)} L(x,y-r), \text{ where } (x,y) \text{ is a Type II compromised isolate.}\end{gathered}$$

   For example if Type I agent of region $S_{2,0}$ and Type II agent of region $S_{3,3}$ are compromised, then the keys $(0,0)$, $(1,2)$, $(2,4)$, $(3,6)$, $(4,1)$, $(0,10)$, $(1,7)$, $(2,11)$, $(3,8)$ and $(4,12)$ are exposed. $\{(4,1),(3,8)\}$ is not a compromised good pair, since $\{(4,1),(3,8),(2,16)\}$ is a good triple and $(2,16)$ is not exposed. So the interlink connected by $(4,1)$ will not be compromised. $(1,7)$ is a isolate and so the interlinks connected by $(1,7)$ will be broken. None of the other links will be broken because all other keys belong to some good pair or triple which have not been fully exposed. If the Type I agent in region $S_{2,0}$ and Type II agent of region $S_{3,4}$, then the interlinks connected by $(3,6)$ (and $(4,13)$)and $(1,8)$ will be broken. Number of interlinks broken is $L(3,6)+L(1,1)$.

5. If all the compromised agents are of Type I or Type III, then number of interlinks broken is given by

$$\begin{gathered}\sum_{(x,y)} L(x,y), \text{ where } \{(x^{-1}, 2r + x^{-1}y \mod r) \text{ is a compromised good pair} \\ + \sum_{(x,y)} L(x,y), \text{ where } (x,y) \text{ is a Type I compromised isolate} \\ + \sum_{(x,y)} L(x,y-2r), \text{ where } (x,y) \text{ is a Type III compromised isolate.}\end{gathered}$$

6. If all the compromised agents are of Type II or Type III, then number of interlinks broken is given by

$$\begin{gathered}\sum_{(x,y)} L(x,y-r), \text{ where } (x,y),(r+(-x^{-1}) \bmod r, 2r+(-x^{-1}y) \mod r)\} \\ \text{is a compromised good pair } + \\ \sum_{(x,y)} L(x,y-r), \text{ where } (x,y) \text{ is a Type II compromised isolate } + \\ \sum_{(x,y)} L(x,y-2r), \text{ where } (x,y) \text{ is a Type III compromised isolate.}\end{gathered}$$

7. If the compromised agents are of Type I, Type II or Type III, then number of interlinks broken is given by

$$\begin{gathered}\sum_{(x,y)} L(x,y), \text{ where } \{(x,y),(-x,r+y),(x^{-1},2r+(x^{-1}y) \mod r)\} \text{ is a} \\ \text{compromised good triple } + \\ \sum_{(x,y)} L(x,y), \text{ where } \{(x,y),(-x,r+y)\} \text{ is a compromised good pair } +\end{gathered}$$

$\sum_{(x,y)} L(x,y)$, where $\{(x,y),(x^{-1}, 2r+(x^{-1}y) \mod r)\}$ is a compromised good pair +
$\sum_{(x,y)} L(x,y-r)$, where $\{(x,y),(-x^{-1}, 2r+(-x^{-1}y) \mod r)\}$ is a compromised good pair +
$\sum_{(x,y)} L(x,y)$, where (x,y) is a compromised isolate of Type I +
$\sum_{(x,y)} L(x,y-r)$, where (x,y) is a compromised isolate of Type II +
$\sum_{(x,y)} L(x,y-2r)$, where (x,y) is a compromised isolate of Type III.

For example let Type I agent of $S_{2,3}$, Type II agents of $S_{0,4}$ and $S_{5,3}$ and Type III agent of $S_{4,2}$ are compromised. Then the keys $(0,3),(1,5),(2,0),(3,2),(4,4),(0,11),(1,11),(2,11),(3,11),(4,11),(0,10),(1,12),(2,7),(3,9),(0,18),(1,20),(2,15),(3,17),(4,19)$ are exposed. Since $\{(4,4),(3,11),(2,15)\}$ is a good triple and is fully compromised, the interlinks connected by $(4,4)$ will be broken. $(0,3),(0,10)$ is a good pair and fully compromised, so the interlink connected by $(0,3)$ will be broken. Links connected by Type II isolates $(1,11)$ and $(1,12)$ and Type III isolates of $(0,18)$ will also be broken. None of the other links will be broken because they belong to some pair or triple which are not fully compromised. The number of interlinks broken will be given by $L(4,4)+L(0,3)+L(1,4)+L(1,5)+L(0,4)$.

Since we know how the interlinks will be affected we calculate the number of interlinks connected by key (x,y). This number is denoted by $L(x,y)$. A region S is called an *interior region* if the Lee sphere of radius ρ_l surrounding S contains an agent of the network. For simplicity we calculate all the links that are connected to interior region S via (x,y). Let (x,y) be a Type I key. Suppose a Type I agent belonging to the region $S_{i,j}$ which has been compromised. Let it contain key (x,y). We find the regions $S_{i',j'}$ within Lee distance ρ_l of $S_{i,j}$ which share the key (x,y). If two regions $S_{i,j}$ and $S_{i-t,j'}$ share the key (x,y), then $xi+j = x(i-t)+j' (\mathrm{mod} r)$. So $j' = j+tx(\mathrm{\ mod\ } r)$. Thus we look at the regions $S_{i-1,j+x}$, $S_{i-2,j+2x}$, $\cdots$, $S_{i-t,j+tx}$ and $S_{i+1,j-x}$, $S_{i+2,j-2x}$, $\cdots$, $S_{i+t,j-tx}$ share key (x,y) with the region $S_{i,j}$. We want to consider only those regions which lie within the Lee Sphere of radius ρ_l. So $|t| \leq \rho_l$ and either $tx \mod r \leq \rho_l - t$ or $r - |tx \mod r| \leq \rho_l - t$. Since x is known, the number of regions sharing key (x,y) within the communication range is the same as finding the number of values of t which satisfy the equations

$$|t| \leq \rho_l \textit{ and } |tx \bmod r| \leq \rho_l - t \tag{6.3.2a}$$

and

$$|t| \leq \rho_l \textit{ and } r - |tx \bmod r| \leq \rho_l - t \tag{6.3.2b}$$

For simplicity, we consider only interior regions and the number of regions connected to a particular region is the solutions of t of the following equation.

$$t \leq \rho_l \textit{ and } tx \bmod r \leq \rho_l - t \tag{6.3.3}$$

r	k	ρ_l	s''	$E''(s'')$
23	15	7	10	0.04589
23	15	5	10	0.05317
31	20	7	10	0.04082
37	30	7	40	0.11696
47	40	9	50	0.09948
53	50	10	50	0.07038

Table 6.4: Experimental value of $E''(s'')$ when number of regions is r^2, number of keys in each agent is k and the Lee distance is ρ_l and s'' agents are compromised.

We now consider agents of Type II and Type III. If two regions $S_{i,j}$ and $S_{i-t,j'}$ share the Type II key $(x, r+y)$, then $-xi+j = -x(i-t)+j'(\mathrm{mod} r)$. So $j' = j - tx(\mod r)$. Thus we see that the regions $S_{i-1,j-x}$, $S_{i-2,j-2x}$, $\cdots$, $S_{i-t,j-tx}$ and $S_{i+1,j+x}$, $S_{i+2,j+2x}$, $\cdots$, $S_{i+t,j+tx}$ are the share key $(x, r+y)$ with the region $S_{i,j}$. We want to consider only those regions which lie within the Lee Sphere of radius ρ_l. Since we consider only the interior regions, the number of interlinks connected by key $(x, r+y)$ of Type II we need to find solutions to Equation 6.3.3. For Type III agents if two regions $S_{i,j}$ and $S_{i',j-t}$ share the Type III key $(x, 2r+y)$, then $xj+i = x(j-t)+i'(\mathrm{mod}\, r)$. So $i' = i+tx(\mod r)$. Thus we see that the regions $S_{i+x,j-1}$, $S_{i+2x,j-2}$, $\cdots$, $S_{i+tx,j-t}$ and $S_{i-x,j+1}$, $S_{i-2x,j+2}$, $\cdots$, $S_{i-tx,j+t}$ are the share key $(x, 2r+y)$ with the region $S_{i,j}$. We want to consider only those regions which lie within the Lee Sphere of radius ρ_l. Since we consider only the interior regions the number of interlinks connected by key $(x, 2r+y)$ of Type III we need to find solutions to Equation 6.3.3.

To find the number of regions connected by Type II key $(x, r+y)$ we find $L(x, y)$ and the number of regions connected by Type III key $(x, 2r+y)$ we find $L(x, y)$. The same formula holds for Type II and Type III keys.

When s'' agents are compromised randomly, number of interlinks disrupted cannot be calculated deterministically. This is because of the following reasons.

1. The exact number of distinct keys cannot be deterministically calculated. This is because it depends on the position of the agents over the entire region.

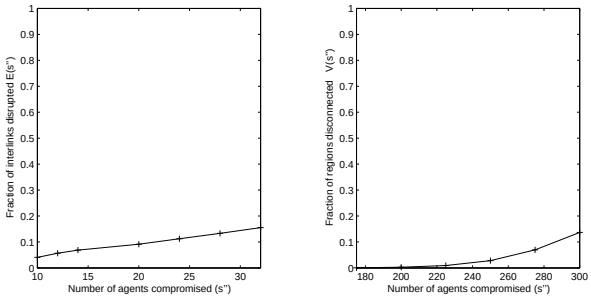
2. The fraction of links compromised depends on the number of good triples, good pairs and isolates which are compromised. Existence of these triples, pairs and isolates depends on the existence of inverse which are less than k. Since inverses are randomly scattered over r, it is difficult to find the number of triples, pairs or isolates affected.

The experimental results for $E''(s'')$ for interlinks is given in Table 6.4. Experimental results are obtained for s'' agents chosen randomly over 100 runs.

r	k	s''	$V''(s'')$
23	15	100	0.00028
31	20	200	0.0027
37	30	300	0.0067
47	40	400	0.0022
53	50	500	0.0043

Table 6.5: Experimental value of $V''(s'')$ when number of regions is r^2, number of keys per agent is k and s'' agents are compromised.

6.3.5 Estimation of $V'(s)$ and $V''(s)$



(a) Fraction of interlinks affected $E(s'')$ when s'' agents are compromised. $r = 31, k = 20$

(b) Fraction of regions disconnected $V(s'')$ when s'' agents are compromised. $r = 31, k = 20$

Figure 6.3: Study of resiliency

When sensor nodes are compromised, keys are exposed. There is chance that all the keys in a non compromised nodes are exposed. So this node is totally disconnected from the network. When links are broken communication can occur through alternate paths. However when sensor nodes are disconnected no communication with the disconnected nodes are possible. Hence node loss must be prevented. None of the other papers on key predistribution using deployment knowledge have discussed this very important issue so far. We look at the local resiliency $V'(s)$ (defined as the fraction of nodes disconnected when s nodes are compromised) and global resiliency $V''(s)$ (defined as the fraction of regions disconnected when s agents are compromised). Our scheme reduces the chance of sensor nodes being disconnected altogether. To disconnect a node all the keys contained therein are exposed. Since a node U_i contains $p + 1$ keys, a minimum of $p + 1$ nodes having those keys must be compromised to disconnect a node in that region. Suppose s' nodes are compromised. An average of s'/r^2 nodes are compromised in each region. So $s'/r^2 > p + 1$. This means on an average $s' > r^2(p + 1)$ nodes have to

Schemes	Deployment	Types	Communication cost	Storage	Scalability
DDHV [47, 49]	Grid-group	Homogeneous	$O(\tau)$	$\tau(\lambda+1)$	Scalable
LN [92, 93]	Grid	Homogeneous	$O(\log C \log R)$	$(t+1)q$	Not scalable
YG [175, 176]	Grid-group	Homogeneous	$\lambda(\ \log g)$	$(\lambda+1)\omega$	Not scalable
ZNR [179]	Group	Heterogeneous	$O(\log N)$	$O(\gamma)^1$ $O(n_s)^2$	Not scalable
HMMH [70]	Grid-group	Homogeneous	$O(\tau)$	$\tau(\lambda+1)$	Scalable
HM [69]	Grid-group	Homogeneous	$O(\tau)$	$\tau(\lambda+1)$	Scalable
SLW [149]-1	Grid-group	Homogeneous	$O(\log p')$	$O(\sqrt{N/g})$	Scalable
SLW [149]-2	Grid-group	Heterogeneous	$O(\log p')$	$O(\sqrt{N/g})$	Scalable
Ours	Grid-group	Heterogeneous	$O(\log p)$	$O(\log n)^1$ $O(\log N)^2$	Not Scalable

Table 6.6: Comparison of the different schemes with respect to the Type of deployment, type of nodes, communication overhead, storage and scalability.

be compromised to disconnect one node. For example, for a 31×31 grid, if nodes in each region contains 18 keys ($p = 17$), then on an average 17298 nodes must be compromised to disconnect one node of each region. This value of s' is quite large. So we can say that our scheme is very resilient. Only very few nodes will be disconnected even when a large number of nodes are compromised.

We next find the number of regions disconnected when s'' agents are compromised. We give experimental results in Table 6.5. Experimental results are obtained for s'' agents chosen randomly over 100 runs. From the table we note that for a 31×31 grid if 200 agent are compromised then about $2-3$ regions will be disconnected, where each agent contains 20 keys. When agents contain 25 keys then on compromising 200 agents about one region will be disconnected. The theoretical bound for $V''(s'')$ is difficult to estimate because it depends on the position and type of agents compromised. The position and type of agents determine which keys are being exposed. If all distinct keys are exposed, then $V''(s'')$ is expected to be more than the case where there is an overlap between the exposed keys. Also if the keys exposed form a good triple or good pair, then $V''(s'')$ is expected to be more than if one of the keys belonging to a good triple or good pair is exposed. Further since finding good triple or good pair depends on the existence of inverse within a certain range ($< k$), the problem becomes all the more difficult, as we know that inverse are scattered randomly. The experimental results for $E''(s'')$ and $V''(s'')$ are given in the Figures 6.3(a) and 6.3(b) respectively.

6.4 Comparison with other schemes

We recall from Section 2.8 the schemes presented in [47, 49, 69, 70, 92, 93, 149, 174–176, 179].

We present a comparative study of type (homogeneous or heterogeneous), communication overhead, storage and scalability of several schemes in Table 6.6. Here τ denotes the number of key spaces selected out of ω spaces in DDHV scheme, c denotes the security parameter for Blom scheme, ω denotes the number of key spaces t denotes the degree of polynomial whose coefficients are in F_q. $C \times R$ is

Schemes	Number of keys	Connectivity
DDHV [47, 49]	200	0.92
LN [92, 93]	200	0.99
ZNR [179]	100	1.00
SLW [149]-1	16	0.5856
SLW [149]-2	40	0.80
Ours	12	1.00

Table 6.7: Comparative study of intra connectivity with the number of keys. The size of the network in DDHV,LN,ZNR is 10000, for SLW it is 12100 and 16093 for our scheme.

the area of the region for LN scheme. g is the number of groups, n is the number of nodes in each group and $\gamma = n_s/g$, where n_s is the total number of sensors. N is the total number of sensors. p and p' are parameters in our scheme and that of SLW schemes respectively. [1] is the storage for small sensor nodes and [2] is the storage for agents.

We compare the connectivity of the various schemes in Table 6.7.

The scheme given by Huang, Mehta, Medhi and Harn [70] (HMMH) and Huang and Medhi [69] (HM) differ from our scheme in that the sensor nodes are randomly distributed in a two dimensional field which is divided into rectangular regions. All nodes have equal power and storage capacities (homogeneous) compared to our scheme where there are two different types of nodes. The nodes in a region can communicate directly with each other with probability > 0.5. Our scheme is better in this respect that all nodes in a region can directly communicate with each other thus reducing delays in communication. Since shared key discovery is by matching identifiers of keys in the sensor nodes, this will involve huge computation and communication costs. We consider a network with a total of $10,000$ nodes. There are 100 regions each having 100 nodes. Suppose 300 nodes are compromised, then fraction of links compromised amongst uncompromised nodes is negligible in [69, 70]. However if all the nodes are considered then the fraction of links compromised will be higher. With $16,093$ sensor nodes, distributed in 121 regions of 133 nodes each, fraction of intra links broken is about 0.07. However this includes not just the links between the uncompromised sensor nodes but all the compromised and uncompromised sensor nodes.

We next compare our scheme with that given by Zhou, Ni and Ravishankar [179] (ZNH). The sensor nodes are deployed similarly as in our scheme however their scheme employs mobile agents instead of static agents in case of ours. The nodes in a region have 99 keys, which is only 12 in case of ours, for a region containing 100 nodes. In their scheme resiliency is defined to be the fraction of links broken, where links are said to exist between two nodes provided they have a common key or a path key which is establish between nodes where there is no

direct communication. As such the resiliency will be lower than ours.

Deployment knowledge is also employed by Liu and Ning [92, 93] (LN) and Blackburn, Etzion, Martin and Paterson [7, 8] (BEMP). There the whole region is divided into squares as in our scheme but instead of a group of nodes being deployed in a square as in our scheme, only one node is placed in a square in these schemes.

Though deployment knowledge has been used by Younis, Ghumman and Eltoweissy [174] (YGE) and Du, Deng, Han and Varshney [49] (DDHV), the deployment scheme is different in that there are no specialized agents to communicate between regions. In these schemes direct communication between nodes in not guaranteed.

In Simonova, Ling and Wang's [149] scheme, the number of specialized nodes depends upon the size of the network unlike ours which is constant (= 3). The resiliency as given in the graph is much lower compared to ours scheme. Also resiliency in terms of nodes or regions disconnected has not been presented.

Though the number of groups is chosen to be r^2, where r is a prime power, our design works in all those cases where the dimension n of the grid is not a prime. This can be done by simply choosing a prime power $r > n$ and neglect the regions which fall out of the $n \times n$ grid.

Our scheme has several advantages. Firstly, the number of keys per node is very low, of the order of $\sqrt{n}$, where n is the number of nodes in a region. Secondly, the agents also have $\sqrt{n'}$ keys, where n' is the number of regions. Thirdly, all nodes can communicate with each other in a group, which results in higher resiliency. Only three agents are required to ensure that all regions which are within Lee distance are connected. The resiliency of our scheme is much better than many state of art schemes. In our scheme we also use another measure of resiliency $V'(s)$ and $V''(s)$ (the fraction of nodes or regions disconnected when s nodes are compromised) which has not been considered in any deployment knowledge based network.

We present a comparison of resiliency of the several schemes with our scheme in Figure 6.4. We consider the schemes with the following parameters.

1. DDHV scheme has parameters $k = 200$, $\omega = 11$ and $\tau = 2$,
2. LN scheme has parameters $k = 200$, $m = 60$ and $L = 1$, m is the number of nodes within RF range of a given node,
3. YG scheme has parameters $k = 100$,
4. ZNR scheme has parameters $k = 100$,
5. HMMH scheme has parameters $k = 200$, $\omega = 27$ and $\tau = 3$
6. SLW scheme has parameters $k = 16$, $p = 11$ and $m = 4$
7. Our scheme has parameters $k = 12$.

The size of the network in DDHV, LN, YG, ZNR, HMMH is 10000, for SLW it is 12100 and 16093 for our scheme. We notice that our resiliency is better than most schemes. Also the number of keys in our scheme is surprisingly low compared to the other schemes.

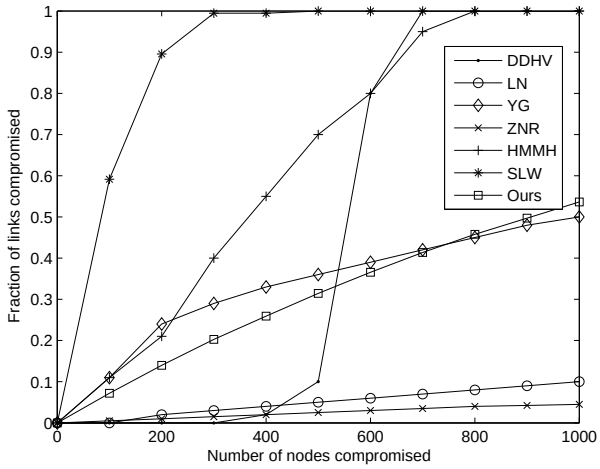


Figure 6.4: Comparison of DDHV, LN, YG, ZNR, HMMH, SLW, and our scheme.

6.5 Concluding remarks

In the next chapter we discuss some known predistribution schemes. We modify the Çamtepe and Yener scheme [18, 20] and present key establishment algorithms for the same. We also present key establishment algorithms for Dong, Pei, Wang 's scheme [44].

Chapter 7

Revisiting some key predistribution schemes

Key establishment is a major problem in sensor networks because of resource constraints. Several key predistribution schemes have been discussed in literature. Though the key predistribution algorithms have been described very well in these papers, no key establishment algorithm has been presented in some of them. Without efficient key establishment algorithm the key predistribution schemes are incomplete. We present efficient shared-key discovery algorithms for some known deterministic key predistribution schemes which employ combinatorial designs. Our algorithms run in $O(1)$ time and the communication overhead is at most $O(log\sqrt{N})$ bits, where N is the size of the network. The efficient key establishment schemes make deterministic key predistribution an attractive option over randomized schemes.

As pointed out by [89], the efficiency of key establishment algorithms depends on two factors.

1. The communication overhead - the amount of information that needs to be broadcasted to enable other nodes to find the common keys.

2. Efficient shared key discovery algorithms - algorithms which are efficient in terms of computation and storage.

In randomized technique of key predistribution by Eschenauer and Gligor [55] and Chan Perrig and Song [28], keys are drawn randomly from a key pool and placed in each sensor node. Suppose each sensor contains k keys. In some schemes such as [28, 55], sensor nodes broadcast the entire list of key identifiers. On receiving a list of identifiers a sensor node compares it with its own identifier list to find one or more common identifiers. Now the key(s) corresponding to this(these) identifier(s) is(are) used to establish a pairwise common key. All encryption and decryption is done using this common key. If there are v keys in the key pool and each key is given a unique id, then for sensor nodes consisting of k keys, $O(k\ log\ v)$

bits needs to be sent. If the list of identifiers are sent in sorted order, then a common identifier can be found in $O(k)$ steps. This is quite high considering the fact that sensors are resource constrained. Also the sorting takes $O(k \log k)$ time, so a huge computation cost is involved. Another way is to use Merkle puzzles [108] as done by Eschenauer and Gligor in [55] and Chan, Perrig and Song in [28]. Then to find one or more common shared keys between two nodes, each node has to broadcast a list $\{\alpha, E_{k_i}(\alpha), i = 1, 2, \cdots, k\}$, where α is a challenge. The decryption of E_{k_i} with proper key by the other node would reveal the challenge α and establish a shared key with the broadcasting node. The communication overhead for the schemes [28,55] will be $O(k\ log\ v)$, where v is the number of keys in the key pool. The calculation of $E_{k_i}(\alpha), i = 1, 2, \cdots, k$ encryption will require $O(k)$ time. However this is not a very efficient way, since communication overhead increases. The computation overhead for encryption and decryption is also quite high.

Deterministic key predistribution using combinatorial designs have been studied in [7, 18, 20, 24, 44, 86, 87, 89, 134]. Though key predistribution algorithms have been discussed in details [18,44], key establishment algorithms have not been given in any of them. Without efficient key establishment algorithms the predistribution schemes are incomplete. In this chapter we present key establishment algorithms for the known key predistribution techniques of Çamtepe and Yener's [18] and Dong, Pei and Wang [44].

We revisit the key predistribution schemes of Çamtepe and Yener's [18] and Dong, Pei and Wang [44]. In *Esorics'04* Çamtape and Yener proposed three key predistribution algorithms, which were modified by them in [20]. Two of these algorithms are deterministic in nature and use combinatorial designs like *projective planes* and *generalized quadrangles*. The third scheme was a hybrid scheme using the above two designs. We modify their predistribution scheme using projective plane slightly. The new key predistribution algorithm has the same time complexity as their scheme, but gives rise to a very simple and efficient shared - key discovery algorithm. In their paper, Çamtepe and Yener assumed that some common pairwise key exists or they can be found by the techniques proposed by [28, 55]. However we saw that these algorithms are costly in terms of computation and communication. We propose a shared key discovery algorithm which runs in $O(1)$ time. The communication overhead is $O(\log \sqrt{N})$, where N is the size of the network. We also propose a shared key discovery algorithm for their scheme using *Generalized quadrangles*. This algorithm also runs in $O(1)$ and has a communication overhead of $O(\log \sqrt{N})$, where N is the size of the network.

We also present efficient shared-key discovery algorithms for the key predistribution schemes given by Dong, Pei, Wang [44]. The shared key discovery algorithms given by [44] run in $O(\sqrt[3]{N})$ time. The communication overhead is $O(\log \sqrt{N})$, where N is the size of the network.

The rest of the chapter is organized as follows. In Section 7.1 we define a few terms and concepts. In Sections 7.2 and 7.3 we present the symmetric key predistribution and key predistribution scheme using Generalized Quadrangles re-

spectively for Çamtepe and Yener's scheme. In Section 7.4 we present key establishment strategies for the key predistribution scheme given by Dong, Pei and Wang in [44]. Path key establishment has been represented in 7.5.

7.1 Preliminaries

Recall the definition of Generalized quadrangles. (Definition 1.5.16 in Section 1.5). Consider the example given in 7.3.1. The line j is denoted by B_j. Each point is incident with four lines. For example the point 0 is incident with lines B_1, B_2, B_3 and B_4. Any two distinct points occur in at most one line. For example the points 0 and 4 are incident with only one line i.e., B_1. Each line is incident with four points. Two distinct lines say B_1 and B_5 are incident with one point (i.e., 4). B_1 and B_{10} do not have any points in common. 0 is a point not incident on B_{10}. There is a unique pair $(13, B_2)$, such that 0 is incident on B_2 and 13 is incident on B_{10}.

7.2 Revisiting Çamtepe-Yener's scheme [18, 20] of key predistribution using symmetric design

Three key predistribution schemes have been proposed in [18]. The first two schemes are deterministic in nature while the third takes a hybrid approach. We discuss the shared discovery for key predistribution using symmetric design in this section and using generalized quadrangles in the next section. The efficiency of these algorithms give one stronger reason of using deterministic designs over probabilistic and hybrid designs.

7.2.1 Key predistribution using symmetric design

According to this design, q^2+q+1 nodes are each preloaded with $q+1$ keys according to a $PG(2, q)$, where q is a prime power. The construction of the symmetric design (which is the same as $PG(2, q)$) given by Çamtepe and Yener uses mutually orthogonal latin squares. They did not however provide an algorithm for shared key discovery and assumed that a shared key exists which can be found by the methods given in [28,55]. Here we use a simpler construction (Algorithm 3) using the technique given in [162, Section 8.4]. This makes the shared key discovery algorithm much simpler. The complexity of our shared-key discovery algorithm is $O(1)$ and the communication overhead is $O(\log\ q)$. We index the nodes (or blocks) by (a, b, c) where $a, b, c \in GF(q)$. The nodes are given by the identifiers $(1, b, c)$, $(0, 1, c)$ and $(0, 0, 1)$, where $b, c \in GF(q)$. So there are a total of $q^2 + q + 1$ nodes. Similarly the keys are indexed by (x, y, z) where $x, y, z, \in GF(q)$. The identifiers

of the keys are given by $(x, y, 1)$, $(x, 1, 0)$ and $(1, 0, 0)$, where $x, y \in GF(q)$. So there are a total of $q^2 + q + 1$ keys (or elements). A key (x, y, z) is assigned to node (a, b, c) if $ax + by + cz = 0$. This design results in a $PG(2, q)$. For details one may refer to [162, Section 8.4]. We note that this predistribution is the same as that given by Çamtepe and Yener. However this method is much simpler than calculating MOLS, then constructing affine planes and then constructing projective planes as given in their construction ([18, Section 3.1]). Steps 1-9 assigns

Algorithm 3 Key predistribution using $PG(2, q)$

1: **for** Each element b in $GF(q)$ **do**
2: **for** Each element c in $GF(q)$ **do**
3: **for** Each element y in $GF(q)$ **do**
4: $x = -(c + by)$
5: Assign key $(x, y, 1)$ to node $(1, b, c)$
6: **end for**
7: Assign key $(-b, 1, 0)$ to node $(1, b, c)$
8: **end for**
9: **end for**
10: **for** Each element c in $GF(q)$ **do**
11: **for** Each element x in $GF(q)$ **do**
12: Assign key $(x, -c, 1)$ to node $(0, 1, c)$
13: **end for**
14: Assign key $(1, 0, 0)$ to node $(0, 1, c)$
15: **end for**
16: **for** Each element x in $GF(q)$ **do**
17: Assign key $(x, 1, 0)$ to node $(0, 0, 1)$
18: **end for**
19: Assign key $(1, 0, 0)$ to node $(0, 0, 1)$

keys to nodes $(1, b, c)$, where $b, c \in GF(q)$. $(1, b, c)$ are assigned keys $(x, y, 1)$ such that $x + by + c = 0$. The key $(-b, 1, 0)$ is also assigned to $(1, b, c)$. Thus a total of $q + 1$ keys are assigned to $(1, b, c)$. Similarly $q + 1$ keys are assigned to the nodes $(0, 1, c)$ where $c \in GF(q)$. and $q + 1$ keys are assigned to the nodes $(0, 0, 1)$. Steps 1 - 9 will take $O(q^3) = O(N^{1.5})$ time, where N is the maximum number of nodes that the network can support. Steps 10 - 15 will take $O(q^2) = O(N)$. Steps 16-19 take $O(q) = O(\sqrt{N})$ time. Thus the algorithm takes $O(N^{1.5})$. This is the same as the algorithm given by Çamtepe Yener in [20, Section III].

Example 7.2.1. *We consider* $q = 5$. *There are* 31 *nodes each node containing six keys. The total number of keys is* 31. *The nodes are represented by* $(1, b, c)$ *for* $b, c \in \{0, 1, 2, 3, 4\}$, $(0, 1, c)$ *for* $c \in 0, 1, 2, 3, 4$ *and* $(0, 0, 1)$. *The keys have identifiers* $(x, y, 1)$ *for* $x, y \in \{0, 1, 2, 3, 4\}$, $(x, 1, 0)$ *for* $x \in 0, 1, 2, 3, 4$ *and* $(1, 0, 0)$. *A key* (x, y, z) *is present in node* (a, b, c) *if* $ax + by + cz = 0$. *Thus the nodes contain the following keys as shown in Table 7.1.*

Node	Keys
$(1,0,0)$	$(0,0,1),(0,1,1),(0,2,1),(0,3,1),(0,4,1),(0,1,0)$
$(1,0,1)$	$(4,0,1),(4,1,1),(4,2,1),(4,3,1),(4,4,1),(0,1,0)$
$(1,0,2)$	$(3,0,1),(3,1,1),(3,2,1),(3,3,1),(3,4,1),(0,1,0)$
$(1,0,3)$	$(2,0,1),(2,1,1),(2,2,1),(2,3,1),(2,4,1),(0,1,0)$
$(1,0,4)$	$(1,0,1),(1,1,1),(1,2,1),(1,3,1),(1,4,1),(0,1,0)$
$(1,1,0)$	$(0,0,1),(1,4,1),(2,3,1),(3,2,1),(4,1,1),(4,1,0)$
$(1,1,1)$	$(0,4,1),(1,3,1),(2,2,1),(3,1,1),(4,0,1),(4,1,0)$
$(1,1,2)$	$(0,3,1),(1,2,1),(2,1,1),(3,0,1),(4,4,1),(4,1,0)$
$(1,1,3)$	$(0,2,1),(1,1,1),(2,0,1),(3,4,1),(4,3,1),(4,1,0)$
$(1,1,4)$	$(0,1,1),(1,0,1),(2,4,1),(3,3,1),(4,2,1),(4,1,0)$
$(1,2,0)$	$(0,0,1),(1,2,1),(2,4,1),(3,1,1),(4,3,1),(3,1,0)$
$(1,2,1)$	$(0,2,1),(1,4,1),(2,1,1),(3,3,1),(4,0,1),(3,1,0)$
$(1,2,2)$	$(0,4,1),(1,1,1),(2,3,1),(3,0,1),(4,2,1),(3,1,0)$
$(1,2,3)$	$(0,1,1),(1,3,1),(2,0,1),(3,2,1),(4,4,1),(3,1,0)$
$(1,2,4)$	$(0,3,1),(1,0,1),(2,2,1),(3,4,1),(4,1,1),(3,1,0)$
$(1,3,0)$	$(0,0,1),(1,3,1),(2,1,1),(3,4,1),(4,2,1),(2,1,0)$
$(1,3,1)$	$(0,3,1),(1,1,1),(2,4,1),(3,2,1),(4,0,1),(2,1,0)$
$(1,3,2)$	$(0,1,1),(1,4,1),(2,2,1),(3,0,1),(4,3,1),(2,1,0)$
$(1,3,3)$	$(0,4,1),(1,2,1),(2,0,1),(3,3,1),(4,1,1),(2,1,0)$
$(1,3,4)$	$(0,2,1),(1,0,1),(2,3,1),(3,1,1),(4,4,1),(2,1,0)$
$(1,4,0)$	$(0,0,1),(1,1,1),(2,2,1),(3,3,1),(4,4,1),(1,1,0)$
$(1,4,1)$	$(0,1,1),(1,2,1),(2,3,1),(3,4,1),(4,0,1),(1,1,0)$
$(1,4,2)$	$(0,2,1),(1,3,1),(2,4,1),(3,0,1),(4,1,1),(1,1,0)$
$(1,4,3)$	$(0,3,1),(1,4,1),(2,0,1),(3,1,1),(4,2,1),(1,1,0)$
$(1,4,4)$	$(0,4,1),(1,0,1),(2,1,1),(3,2,1),(4,3,1),(1,1,0)$
$(0,1,0)$	$(0,0,1),(1,0,1),(2,0,1),(3,0,1),(4,0,1),(1,0,0)$
$(0,1,1)$	$(0,4,1),(1,4,1),(2,4,1),(3,4,1),(4,4,1),(1,0,0)$
$(0,1,2)$	$(0,3,1),(1,3,1),(2,3,1),(3,3,1),(4,3,1),(1,0,0)$
$(0,1,3)$	$(0,2,1),(1,2,1),(2,2,1),(3,2,1),(4,2,1),(1,0,0)$
$(0,1,4)$	$(0,1,1),(1,1,1),(2,1,1),(3,1,1),(4,1,1),(1,0,0)$
$(0,0,1)$	$(0,1,0),(4,1,0),(3,1,0),(2,1,0),(1,1,0),(1,0,0)$

Table 7.1: Distribution of keys in nodes

We now present the algorithm for shared key discovery.

7.2.2 Algorithm for shared-key discovery

The Algorithm 4 takes as input two nodes U_i and U_j having ids (a_i, b_i, c_i) and (a_j, b_j, c_j). It finds the identifier of the common key as (x, y, z). All calculations are done in $GF(q)$.

Example 7.2.1: Contd. We illustrate the algorithm with the example in Section 7.2.1 Suppose node 31 wants to communicate 29. The nodes broadcast the identifiers $(0, 0, 1)$ and $(0, 1, 3)$. So step 3 of Algorithm 4 will be executed. And the identifier of the common key will be given by $(1, 0, 0)$. Suppose node 31 wants to communicate 1. The nodes broadcast the identifiers $(0, 0, 1)$ and $(1, 0, 0)$. So step 5 of algorithm will be executed. And the identifier of the common key will be given by $(0, 1, 0)$. Similarly if node 29 wants to communicate with node 28, then from the broadcasted information $(0, 1, 3)$ of node 28 and $(0, 1, 2)$ of node 28, we calculate the id of the common key as $(1, 0, 0)$ (Step 9). However if node 29 wants to communicate with node 25 then step 11 will be executed and the common key will be calculated as $91, 2, 1)$. If node 18 wants to communicate with node 25, then step 14 will be executed and the identifier of the common key will be calculated as $(4, 3, 1)$.

Algorithm 4 Shared key discovery using symmetric design

Require: : (a_i, b_i, c_i) and (a_j, b_j, c_j), the identifiers of nodes i and j respectively.

1: **if** $a_i = 0$ and $b_i = 0$ and $c_i = 1$ **then**
2: **if** $a_j = 0$ and $b_j = 1$ **then**
3: Identifier of the common key $= (1, 0, 0)$
4: **else**
5: Identifier of the common key $= (-b_j, 1, 0)$
6: **end if**
7: **else if** $a_i = 0$ and $b_i = 1$ **then**
8: **if** $a_j = 0$ and $b_j = 1$ **then**
9: Identifier of the common key $= (1, 0, 0)$
10: **else**
11: Identifier of the common key $= (b_j c_i - c_j, -c_i, 1)$
12: **end if**
13: **else** { When $(a_i, b_i, c_i) = (1, b_1, c_1)$ and $(a_j, b_j, c_j) = (1, b_2, c_2)$}
14: Identifier of the common key $= (-c_1 + b_1 \frac{c_1 - c_2}{b_1 - b_2}, \frac{c_2 - c_1}{b_1 - b_2}, 1)$
15: **end if**

7.2.3 Correctness of the Algorithm 4

Let two nodes with ids $i = (a_i, b_i, c_i)$ and $j = (a_j, b_j, c_j)$ communicate with each other. Steps 1-3 are straight forward. The common key between nodes $(0, 0, 1)$

and $(1, b, c)$ will be $(x, 1, 0)$. From this we get $x = -b$. So step 5 is justified. Step 7-9 follows from Step 14 of Algorithm 4. To find the common keys between the nodes $(0, 1, c_i)$ and $(1, b_j, c_j)$, the common key is of the form $(x, y, 1)$. We solve the following equations

$$x + b_j y + c_j = 0 \tag{7.2.1}$$
$$y + c_i = 0 \tag{7.2.2}$$

So identifier of the common key $= (b_j c_i - c_j, -c_i, 1)$. Steps 1-12 are straight forward. For step 13 the common key will be of the form $(x, y, 1)$. We solve the equations

$$x + b_1 y + c_1 = 0 \tag{7.2.3}$$
$$x + b_2 y + c_2 = 0 \tag{7.2.4}$$

7.2.4 Time complexity of Algorithm 4

All the steps take $O(1)$ time to be calculated. The only information that needs to be broadcasted is the identifiers represented by the tuple (a, b, c), where a, b, c are in $GF(q)$. Thus the communication overhead is $O(\log\ q) = O(\log \sqrt{N})$, where N is the number of nodes in the network. This is very less compared to the time complexity given in [28, 55].

7.3 Shared key discovery for key predistribution using generalized quadrangles

Recall the definition of Generalized Quadrangles from Definition 1.5.16 given in Section 1.5 and example given in Section 7.1. Three known designs for generalized quadrangles have been used : $GQ(q, q)$, $GQ(q, q^2)$ and $GQ(q^2, q^3)$. The construction of $GQ(q, q)$, $GQ(q, q^2)$ and $GQ(q^2, q^3)$ have been done from $PG(4, q)$, $PG(5, q)$ and $H(4, q^2)$. Details can be found in [20, Section B] We discuss the shared key discovery for the scheme obtained by $GQ(q, q)$. For the other two designs we use a similar approach. For completeness we describe the algorithm (Algorithm 5) briefly. Details can be found in [18].

Algorithm 5 Key predistribution using $GQ(q, q)$

1: Find minimum prime q s.t. $(q + 1)(q^2 + 1) \geq N$;
2: Find the points $x =< x_0, x_1, x_2, x_3, x_4 >$ in $GF(q)$ which satisfy the equation $x_0^2 + x_1 x_2 + x_3 x_4 = 0$.
3: For each point x, generate the set of points collinear to x. There are $v = (q + 1)(q^2 + 1)$ non-collinear points.
4: Construct $b = (q + 1)(q^2 + 1)$ lines with $q + 1$ points.

It can be easily seen that given two points $X^{(0)} =< x_0^{(0)}, x_1^{(0)}, x_2^{(0)}, x_3^{(0)}, x_4^{(0)} >$ and $X^{(1)} =< x_0^{(1)}, x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, x_4^{(1)} >$, any other point lying on the line joining $X^{(0)}$ and $X^{(1)}$ is given by $X^{(t)} =< tx_0^{(0)} + x_0^{(1)}, tx_1^{(0)} + x_1^{(1)}, tx_2^{(0)} + x_2^{(1)}, tx_3^{(0)} + x_3^{(1)}, tx_4^{(0)} + x_4^{(1)} >$., where $t = 1, 2, 3, \cdots, q-1$. This important observation helps us in finding a shared key between two nodes efficiently. The points represent the key ids. The vectors can be used to calculate the keys. The keys are predistributed in the node such that each node i receives the points (keys) that lie on line i. Let $k_0^{(i)}$, $k_1^{(i)}$, $\cdots$, $k_q^{(i)}$ represent the ids of the keys belonging to node i, and $k_0^{(j)}$, $k_1^{(j)}$, $\cdots$, $k_q^{(j)}$ represent the ids of the keys belonging to node j. Any key id $k_s^{(t)}$ is given by $< x_0^{(ts)}, x_1^{(ts)}, \cdots, x_4^{(ts)} >$.

7.3.1 Finding common key between nodes

In this section we show how we can find a common key between two nodes U_i and U_j efficiently, if one exists or report failure if none exists. When two nodes U_i and U_j want to communicate with each other, they broadcast the vectors corresponding to two keys which have the minimum id. Algorithm 6 depicts how a node U_i will calculate id of the common key which it shares with another node U_j. Node U_j will run the same algorithm and find t_2 to enable it to calculate the id of the common key as $< t_2x_0^{(j0)} + x_0^{(j1)}, t_2x_1^{(j0)} + x_1^{(j1)}, t_2x_2^{(j0)} + x_2^{(j1)}, t_2x_3^{(j0)} + x_3^{(j1)}, t_2x_4^{(j0)} + x_4^{(j1)} >$

Example 7.3.1. *We present an example to illustrate the Algorithm 6. Let* $q = 3$. *There are forty blocks each containing four keys. The blocks are given below. Let the vectors from which the keys can be obtained will be given by*
$0 =< 0, 1, 0, 0, 0 >$, $1 =< 0, 0, 1, 0, 0 >$, $2 =< 1, 2, 1, 0, 0 >$, $3 =< 2, 2, 1, 0, 0 >$, $4 =< 0, 0, 0, 1, 0 >$, $5 =< 0, 1, 0, 1, 0 >$, $6 =< 0, 2, 0, 1, 0 >$, $7 =< 0, 0, 1, 1, 0 >$, $8 =< 1, 2, 1, 1, 0 >$, $9 =< 2, 2, 1, 1, 0 >$, $10 =< 0, 0, 2, 1, 0 >$, $11 =< 1, 1, 2, 1, 0 >$, $12 =< 2, 1, 2, 1, 0 >$, $13 =< 0, 0, 0, 0, 1 >$, $14 =< 0, 1, 0, 0, 1 >$, $15 =< 0, 2, 0, 0, 1 >$, $16 =< 0, 0, 1, 0, 1 >$, $17 =< 1, 2, 1, 0, 1 >$, $18 =< 2, 2, 1, 0, 1 >$, $19 =< 0, 0, 2, 0, 1 >$, $20 =< 1, 1, 2, 0, 1 >$, $21 =< 2, 1, 2, 0, 1 >$, $22 =< 1, 1, 1, 1, 1 >$, $23 =< 2, 1, 1, 1, 1 >$, $24 =< 0, 2, 1, 1, 1 >$, $25 =< 0, 1, 2, 1, 1 >$, $26 =< 1, 2, 2, 1, 1 >$, $27 =< 2, 2, 2, 1, 1 >$, $28 =< 1, 0, 0, 1, 2 >$, $29 =< 2, 0, 0, 2, 1 >$, $30 =< 1, 1, 0, 2, 1 >$, $31 =< 2, 1, 0, 2, 1 >$, $32 =< 1, 2, 0, 2, 1 >$, $33 =< 2, 2, 0, 2, 1 >$, $34 =< 1, 0, 1, 2, 1 >$, $35 =< 2, 0, 1, 2, 1 >$, $36 =< 0, 1, 1, 2, 1 >$, $37 =< 1, 0, 2, 2, 1 >$, $38 =< 2, 0, 2, 2, 1 >$, $39 =< 0, 2, 2, 2, 1 >$,
The nodes will contain the following key identifiers.
$U_1 = \{ 0, 4, 5, 6\}$, $U_2 = \{ 0, 13, 14, 15\}$, $U_3 = \{ 0, 28, 30, 32\}$,
$U_4 = \{ 0, 29, 31, 33\}$, $U_5 = \{ 1, 4, 7, 10\}$, $U_6 = \{ 1, 13, 16, 19\}$,
$U_7 = \{ 1, 28, 34, 37\}$, $U_8 = \{ 1, 29, 35, 38\}$, $U_9 = \{ 2, 4, 8, 12\}$,
$U_{10} = \{ 2, 13, 17, 21\}$, $U_{11} = \{ 2, 30, 35, 39\}$, $U_{12} = \{ 2, 33, 36, 37\}$,
$U_{13} = \{ 3, 4, 9, 11\}$, $U_{14} = \{ 3, 13, 18, 20\}$, $U_{15} = \{ 3, 31, 34, 39\}$,
$U_{16} = \{ 3, 32, 36, 38\}$, $U_{17} = \{ 5, 19, 25, 39\}$, $U_{18} = \{ 5, 20, 26, 37\}$,
$U_{19} = \{ 5, 21, 27, 38\}$, $U_{20} = \{ 6, 16, 24, 36\}$, $U_{21} = \{ 6, 17, 22, 34\}$,

Algorithm 6 Shared key discovery for GQ(q,q) scheme

Require: Two vectors corresponding to the ids with minimum value belonging to each node i and j.

if $k_0^{(i)} = k_0^{(j)}$ **then**
  Id of the common key is $k_0^{(i)}$
else if $k_0^{(i)} = k_1^{(j)}$ **then**
  Id of the common key is $k_0^{(i)}$
else if $k_1^{(i)} = k_0^{(j)}$ **then**
  Id of the common key is $k_1^{(i)}$
else if $k_1^{(i)} = k_1^{(j)}$ **then**
  Id of the common key is $k_1^{(i)}$
else
  Check if for some t_1, $t_1 x_s^{(i0)} + x_s^{(i1)} = x_s^{(j1)}$, $s = 0, 1, \cdots, 4$
  Common key is $< x_0^{(j1)}, x_1^{(j1)}, x_2^{(j1)}, x_3^{(j1)}, x_4^{(j1)} >$, Break;
  Check if for some t_1 , $t_1 x_s^{(i0)} + x_s^{(i1)} = x_s^{(j0)}$, $s = 0, 1, \cdots, 4$
  Common key is $< x_0^{(j0)}, x_1^{(j0)}, x_2^{(j0)}, x_3^{(j0)}, x_4^{(j0)} >$, Break;
  Check if for some t_2, $t_2 x_s^{(j0)} + x_s^{(j1)} = x_s^{(i1)}$, $s = 0, 1, \cdots, 4$
  Common key is $< x_0^{(i1)}, x_1^{(i1)}, x_2^{(i1)}, x_3^{(i1)}, x_4^{(i1)} >$, Break;
  Check if for some t_2, $t_2 x_s^{(j0)} + x_s^{(j1)} = x_s^{(i0)}$, $s = 0, 1, \cdots, 4$
  Common key is $< x_0^{(i0)}, x_1^{(i0)}, x_2^{(i0)}, x_3^{(i0)}, x_4^{(i0)} >$, Break;
  $tag = 0$
  for $s = 0$ to 3 **do**
    for $s' = s + 1$ to 4 **do**
      If t_1, t_2 exists, such that
      $t_1 x_s^{(i0)} + x_s^{(i1)} = t_2 x_s^{(j0)} + x_s^{(j1)}$ and $t_1 x_{s'}^{(i0)} + x_{s'}^{(i1)} = t_2 x_{s'}^{(j0)} + x_{s'}^{(j1)}$,
      $tag = 1$
    end for
  end for
  if $tag = 1$ **then**
    Common key will be $< t_1 x_0^{(i0)} + x_0^{(i1)}, t_1 x_1^{(i0)} + x_1^{(i1)}, t_1 x_2^{(i0)} + x_2^{(i1)}, t_1 x_3^{(i0)} + x_3^{(i1)}, t_1 x_4^{(i0)} + x_4^{(i1)} >$
  else
    Print : No common key exists
  end if
end if

$U_{22} = \{\ 6, 18, 23, 35\}$, $U_{23} = \{\ 7, 15, 24, 39\}$, $U_{24} = \{\ 7, 17, 26, 32\}$,
$U_{25} = \{\ 7, 18, 27, 33\}$, $U_{26} = \{\ 8, 15, 22, 38\}$, $U_{27} = \{\ 8, 16, 23, 39\}$,
$U_{28} = \{\ 8, 18, 25, 28\}$, $U_{29} = \{\ 9, 15, 23, 37\}$, $U_{30} = \{\ 9, 16, 27, 30\}$,
$U_{31} = \{\ 9, 17, 25, 29\}$, $U_{32} = \{10, 14, 25, 36\}$, $U_{33} = \{10, 20, 22, 30\}$,
$U_{34} = \{10, 21, 23, 31\}$, $U_{35} = \{11, 14, 26, 35\}$, $U_{36} = \{11, 19, 22, 33\}$,
$U_{37} = \{11, 21, 24, 28\}$, $U_{38} = \{12, 14, 27, 34\}$, $U_{39} = \{12, 19, 23, 32\}$,
$U_{40} = \{12, 20, 24, 29\}$.

7.3.2 Proof of correctness of Algorithm 6

Steps 1-8 are straight forward. If the ids of the keys sent by j matches that of i, then the matching id is the id of the common key. To check for matching ids, the vectors must be checked tuple by tuple. This is done in constant time. If this does not happen then we check to see if there exists a value of t_1, such that $t_1 x_s^{(i0)} + x_s^{(i1)} = x_s^{(j0)}$. This can be calculated as $< x_0^{(j0)}, x_1^{(j0)}, x_2^{(j0)}, x_3^{(j0)}, x_4^{(j0)} >$. $t_1 = (x_s^{(j0)} - x_s^{(i1)})(x_s^{(i0)})^{-1}$, for $s = 0, 1, 2, 3, 4$. This can be easily done in constant time, if we find all the values of t_1 corresponding to each value of s and check if all five values of t are the same. Then the key can be calculated as $< x_0^{(j0)}, x_1^{(j0)}, x_2^{(j0)}, x_3^{(j0)}, x_4^{(j0)} >$. Similarly, we check to see if $t_1 x_s^{(i0)} + x_s^{(i1)} = x_s^{(j1)}$. We also check if there exists a value of t_2 such that $t_2 x_s^{(j0)} + x_s^{(j1)} = x_s^{(i0)}$ or $t_2 x_s^{(j0)} + x_s^{(j1)} = x_s^{(i1)}$. Suppose such a thing does not happen, then we try to find t_1 and t_2 such that the following equation will be satisfied.

$$t_1 x_s^{(i0)} + x_s^{(i1)} = t_2 x_s^{(j0)} + x_s^{(j1)} \tag{7.3.1}$$

We form at most $\binom{5}{2}$ equations. We solve for t_1 and t_2 if a solution exists. Let $t_1 m + t_2 n = p$ and $t_1 m' + t_2 n' = p'$ be two equations, then no solutions exists, if $m/m' = n/n'$. The condition if $m/m' = n/n' = p/p'$ does not arise. Suppose this was true. Then it would imply that two lines will meet at more than one point which violates the definition of Generalized quadrangles. If the solutions t_1 and t_2 are none equal to zero, then the value of t_1 so obtained can be used to calculate the key that i shares with j and is given in step 27.

Example 7.3.1: Contd. Suppose sensors U_1 and U_4 want to communicate. Then U_1 broadcasts two vectors which constitute the keys $< 0, 1, 0, 0, 0 >$ and $< 0, 0, 0, 1, 0 >$ and U_2 broadcasts $< 0, 1, 0, 0, 0 >$ and $< 1, 0, 0, 2, 1 >$. Since 0 is the common identifier, it is the identifier of the common key. Similarly, U_{36} sends the vectors, $< 1, 1, 2, 1, 0 >$ and $< 0, 0, 2, 0, 1 >$ to U_{17}. Then 19 is the common identifier (by step 7) of the algorithm. If node U_{12} wants to communicate with U_{36}, then from the vectors $< 1, 1, 2, 1, 0 >$ and $< 0, 0, 2, 0, 1 >$ broadcasted by U_{36}, U_{12} finds that $t_2 = 2$ (Step 15) and the common identifier is 33. If node U_{12} wants to communicate with U_{32}, then from the vectors $< 0, 0, 2, 1, 0 >$ and $< 0, 1, 0, 0, 1 >$ broadcasted by U_{32}, U_{12} finds that $t_1 = 2$ and $t_2 = 1$, (Steps 21-22) and the common identifier is 36.

7.3.3 Time complexity of Algorithm 6

Since the expressions within the loops ($s = 0$ to 3 and $s' = s + 1$ to 4) are performed a constant number of times, the entire algorithm has a run time of $O(1)$. For finding the id of the common key, node U_i must have only two vectors from node j. Each key is a vector of length 5 chosen from the field q. Hence the communication overhead is $O(\log\ q)$ bits. Since number of nodes is $O(q^3)$, $O(\log\ \sqrt[3]{N})$ bits are required for communications.

7.4 Shared-key discovery for Dong-Pei-Wang scheme [44]

The key predistribution scheme proposed by Dong et al. in [44] makes use of $3-designs$. In particular they use inversive planes to assign keys in the sensor nodes. However shared key discovery algorithm for such a scheme has not been discussed by the authors in [44]. We present an algorithm to find a common key between two given nodes or report failure if no common key is present. For completeness we present the key predistribution algorithm using 3-designs.

Let q be a prime. We use an irreducible polynomial $f(x)$ of order 2 to construct a field $F_{q^2} = Z_q/(f(x))$. Let $f(x) = x^2 + f_1'x + f_0'$.

Let the field elements be $f_0 = 0$, $f_1 = 1$, $f_2, \cdots, f_{q^2-1}$. We choose $a, b, c, d \in F_{q^2}$, such that $ad - bc \neq 0$. Let $\infty \notin F_q$. We define a function

$$\pi\begin{pmatrix} a & b \\ c & d \end{pmatrix}(x) = \begin{cases} \frac{ax+b}{cx+d} & \text{if } x \in F_q \text{ and } cx + d \neq 0 \\ \infty & \text{if } x \in F_q,\ cx + d = 0 \text{ and } ax + b \neq 0 \\ \frac{a}{c} & \text{if } x = \infty \text{ and } c \neq 0 \\ \infty & \text{if } x = \infty,\ c = 0 \text{ and } a \neq 0 \end{cases}$$

Let $PGL(2, q^2)$ to consist of all distinct permutations $\pi\begin{pmatrix} a & b \\ c & d \end{pmatrix}$, where $a, b, c, d \in F_{q^2}$, such that $ad - bc \neq 0$. It can be proved as in [154, Lemma 9.25] that there are $q^6 - q^2$ such permutations.

We create blocks in the following way. For each permutation π_i, $(i = 0, 1, 2, \cdots, q^6 - q^2 - 1)$ block B_{π_i} consists elements $\pi_i(j)$, $j = 0, 1, \cdots, q - 1, \infty$. So each block consists of $q + 1$ elements. The resulting design gives rise to a $3 - (q^2 + 1, q + 1, 1)$ design as stated in Theorem 1.5.13.

We consider the distinct blocks and map the blocks to the nodes and preload each node with the keys contained in that particular block. Since the number of distinct blocks is $q^3 + q$, the number of nodes supported by the network is $q^3 + q$. Let the key chain belonging to node U_i be denoted by $\{k_t^{(i)} : 0 \leq t \leq q\}$. Any two nodes can share at most two keys.

Next we describe an algorithm to find the common keys between any two nodes if one exists, or report failure if one doesn't exists. Consider the following example.

Block number	a	b	c	d	Blocks (Users)
1	0	1	1	0	$\{\infty, 1+x, 0\}$
2	0	1	1	1	$\{1, x, 0\}$
3	0	1	x	0	$\{\infty, x, 0\}$
4	0	1	x	x	$\{1, 1+x, 0\}$
5	0	1	$1+x$	0	$\{\infty, 1, 0\}$
6	0	1	$1+x$	x	$\{1+x, x, 0\}$
7	1	1	0	1	$\{1, 1+x, \infty\}$
8	1	1	0	x	$\{1+x, x, \infty\}$
9	1	1	0	$1+x$	$\{x, 1, \infty\}$
10	1	1	1	$1+x$	$\{x, 1+x, 1\}$

Table 7.2: Each block corresponds to the keys given to each user

Example 7.4.1. *Consider $q = 2$. There are five elements ∞, 0, 1, x, $1+x$. We can construct $q^3 + q = 10$ blocks each containing $q + 1 = 3$ elements. The the following blocks can be constructed. Each user is given a set of keys corresponding to a block. Each user is given three keys. The table 7.1 gives the keys present in each block. Each user broadcasts the values of a, b, c, d.*

7.4.1 Algorithm to find common key

Consider the Algorithm 7. Let node U_i want to communicate with node U_j. The node U_j broadcasts corresponding values of a, b, c, d. Denote these values by a_j, b_j, c_j, d_j. We give below the algorithm to find the common key that U_i shares with U_j. When U_j wants to calculate the common key that it shares with U_i, it runs the same algorithm and finds x_j and calculates the common key as $ck = \frac{a_j x_j + b_j}{c_j x_j + d_j}$. (See step 25). All calculations are done modulo q.

Consider the example given in Table 7.2. If nodes U_1 and U_4 want to communicate. U_4 sends $a = 0$, $b = 1$, $c = x$, $d = x$. U_1 follows the algorithm. We see that $tag = 1$ in step 9, since $k_0^{(i)} = \infty$. However $c_j s + d_j \neq \infty$ for any value of s. So we solve the equation in step 19. We see that $x_i = 1$. So the common key will given by $ck = 1 + x$.

7.4.2 Correctness of Algorithm 7

Suppose $c_i = 0$ and $c_j = 0$, then $\pi_i(\infty) = \pi_j(\infty) = \infty$, hence $ck = \infty$ and Step 1-2 holds. If $\pi_i(\infty) = a_i/c_i$, if $a_i/c_i \neq 0$. So if $a_i/c_i = a_j/c_j \neq 0$, then $\pi_i(\infty) = \pi_j(\infty) = a_i/c_i$ and Step 3-4 holds. If one of the keys in node i is ∞, and $c_j x_j + d_j = 0$, for some $x_j \in F_q \bigcup \{\infty\}$, then ∞ is a common key between the nodes U_i and U_j. The only condition that remains if that when $x_i, x_j \neq \infty$ and

Algorithm 7 Shared key discovery for the scheme of Dong et al. [44]

Require: a_i, b_i, c_i, d_i, a_j, b_j, c_j, d_j.

if $c_i = 0$ and $c_j = 0$ **then**
  $ck = \infty$
else if $a_i/c_i = a_j/c_j \neq 0$ **then**
  $ck = a_i/c_i$
else
  $tag = 0$
  for $s = 0$ to q **do**
    if $k_s^{(i)} = \infty$ **then**
      $tag = 1$
    end if
  end for
  if $tag = 1$ **then**
    for $s = 0$ to q **do**
      if $c_j s + d_j = 0$ **then**
        $ck = \infty$
      else
        Print : No solution exists
      end if
    end for
  else
    Solve the equation $\frac{a_i x_i + b_i}{c_i x_i + d_i} = \frac{a_j x_j + b_j}{c_j x_j + d_j}$ for x_i.
    if No solution exists **then**
      Print : No solution exists
    else
      $ck = \frac{a_i x_i + b_i}{c_i x_i + d_i}$
    end if
  end if
end if

$c_ix_i + d_i \neq \infty$ and $c_jx_j + d_j \neq \infty$. In such a case we try to find if there exists x_i and x_j, such that $\frac{a_ix_i+b_i}{c_ix_i+d_i} = \frac{a_jx_j+b_j}{c_jx_j+d_j}$. Hence if a solution to this equation exists, then the common key will be $\frac{a_ix_i+b_i}{c_ix_i+d_i}$. By the design we know that any two nodes will share maximum of two keys. We now show that we can find all the values of x_i if they exist, or report failure if no keys are common.

We know that a, b, c d are all one degree polynomial with coefficients in F_q. Let

$$\begin{aligned}
a_i &= a_{i1}x + a_{i0},\\
b_i &= b_{i1}x + b_{i0},\\
c_i &= c_{i1}x + c_{i0},\\
d_i &= d_{i1}x + d_{i0},\\
a_j &= a_{j1}x + a_{j0},\\
b_j &= b_{j1}x + b_{j0},\\
c_j &= c_{j1}x + c_{j0},\\
d_j &= d_{j1}x + d_{j0},
\end{aligned}$$

We solve for x_i in the following equation. Note that all calculations are done modulo q.

$\frac{a_ix_i+b_i}{c_ix_i+d_i} = \frac{a_jx_j+b_j}{c_jx_j+d_j}$

$\Rightarrow (a_ix_i + b_i)(c_jx_j + d_j) = (c_ix_i + d_i)(a_jx_j + b_j)$

$\Rightarrow \{(a_{i1}x + a_{i0})x_i + (b_{i1}x + b_{i0})\}\{(c_{j1}x + c_{j0})x_j + (d_{j1}x + d_{j0})\}$
$= \{(c_{i1}x + c_{i0})x_i + (d_{i1}x + d_{i0})\}\{(a_{j1}x + a_{j0})x_j + (b_{j1}x + b_{j0})\}$

$\Rightarrow \{(a_{i1}x_i + b_{i1})x + (a_{i0}x_i + b_{i0})\}\{(c_{j1}x_j + d_{j1})x + (c_{j0}x_j + d_{j0})\}$
$= \{(c_{i1}x_i + d_{i1})x + (c_{i0}x_i + d_{i0})\}\{(a_{j1}x_j + b_{j1})x + (a_{j0}x_j + b_{j0})\}$

$\Rightarrow x\{(a_{i1}x_i + b_{i1})(c_{j0}x_j + d_{j0}) + (a_{i0}x_i + b_{i0})(c_{j1}x_j + d_{j1}) +$
$(q - f'_1)(a_{i1}x_i + b_{i1})(c_{j1}x_j + d_{j1})\} +$
$(a_{i0}x_i + b_{i0})(c_{j0}x_j + d_{j0}) + (q - f'_0)(a_{i1}x_i + b_{i1})(c_{j1}x_j + d_{j1})\}$
$= x\{(a_{j1}x_j + b_{j1})(c_{i0}x_i + d_{i0}) + (a_{j0}x_j + b_{j0})(c_{i1}x_i + d_{i1}) +$
$(q - f'_1)(a_{j1}x_j + b_{j1})(c_{i1}x_i + d_{i1})\} +$
$(a_{j0}x_j + b_{j0})(c_{i0}x_i + d_{i0}) + (q - f'_0)(a_{j1}x_j + b_{j1})(c_{i1}x_i + d_{i1})\}$

Equating the coefficients of x and the constant term we get two equations

$$P_1x_ix_j + Q_1x_i + R_1x_j + S_1 = 0 \tag{7.4.1a}$$

and

$$P_2x_ix_j + Q_2x_i + R_2x_j + S_2 = 0 \tag{7.4.1b}$$

where,

$P_1 = a_{i1}c_{j0} + a_{i0}c_{j1} + (q - f'_1)a_{i1}c_{j1} - (a_{j1}c_{i0} + a_{j0}c_{i1} + (q - f'_1)a_{j1}c_{i1})$,
$P_2 = a_{i0}c_{j0} + (q - f'_0)a_{i1}c_{j1} - (a_{j0}c_{i0} + (q - f'_0)a_{j1}c_{i1})$,
$Q_1 = a_{i1}d_{j0} + a_{i0}d_{j1} + (q - f_1)a_{i1}d_{j1} - (b_{j1}c_{i0} + b_{j0}c_{i1} + (q - f'_1)b_{j1}c_{i1})$,
$Q_2 = a_{i0}d_{j0} + (q - f'_0)a_{i1}d_{j1} - (b_{j0}c_{i0} + (q - f'_0)b_{j1}c_{i1})$,
$R_1 = b_{i1}c_{j0} + b_{i0}c_{j1} + (q - f'_1)b_{i1}c_{j1} - (a_{j1}d_{i0} + a_{j0}d_{i1} + (q - f'_1)a_{j1}d_{i1})$,

$R_2 = b_{i0}c_{j0} + (q - f'_0)b_{i1}c_{j1} - (a_{j0}d_{i0} + (q - f'_0)a_{j1}d_{i1})$,
$S_1 = b_{i1}d_{j0} + b_{i0}d_{j1} + (q - f'_1)b_{i1}d_{j1} - (b_{j1}d_{i0} + b_{j0}d_{i1} + (q - f'_1)b_{j1}d_{i1})$,
$S_2 = b_{i0}d_{j0} + (q - f'_0)b_{i1}d_{j1} - (b_{j0}d_{i0} + (q - f'_0)b_{j1}d_{i1})$.
Eliminating the term $x_i x_j$, we get

$$(\frac{Q_1}{P_1} - \frac{Q_2}{P_2})x_i + (\frac{R_1}{P_1} - \frac{R_2}{P_2})x_j = \frac{S_2}{P_2} - \frac{S_1}{P_1}$$

$$\Rightarrow x_j = \{(\frac{S_2}{P_2} - \frac{S_1}{P_1}) - (\frac{Q_1}{P_1} - \frac{Q_2}{P_2})x_i\}(\frac{R_1}{P_1} - \frac{R_2}{P_2})^{-1}$$

$$x_j = U + Vx_i \tag{7.4.2}$$

where, $U = (\frac{S_2}{P_2} - \frac{S_1}{P_1})(\frac{R_1}{P_1} - \frac{R_2}{P_2})^{-1}$ and $V = q - (\frac{Q_1}{P_1} - \frac{Q_2}{P_2})(\frac{R_1}{P_1} - \frac{R_2}{P_2})^{-1}$

Substituting the value of x_j in (7.4.1a) we get

$$\begin{aligned} P_1(Vx_i + U)x_i + Q_1(Vx_i + U) + R_1(Vx_i + U) + S_1 &= 0 \\ \Rightarrow P_1Vx_i^2 + x_i(UP_1 + VQ_1 + VR_1) + UP_1 + UQ_1 + UR_1 + S_1 &= 0 \end{aligned}$$

where

The above equation can have either one or two or no solutions which can be calculated easily. Hence, we obtain a maximum of two values for x_i. Then the common key will be $\frac{a_i x_i + b_i}{c_i x_i + d_i}$.

Thus the algorithm gives all the common keys or reports none is present.

7.4.3 Time complexity of Algorithm 7

Steps 7 to 10 are executed at most q times. All the other steps require constant time. Since the number of nodes N is $O(\sqrt[3]{N})$, the time complexity is $O(\sqrt[3]{N})$. Only the four values of a, b, c and d need to be broadcasted. Hence the communication overhead is $O(\log q) = O(\log \sqrt{N})$ bits, which is quite efficient compared to algorithms proposed in [28, 55].

7.5 Path key establishment

Where shared key exists between nodes, a secure channel is created and all communications between the nodes are performed using the common key. However there may exist situations where nodes may not share common keys (as in the scheme of [44] which uses $t - designs$) or when common shared keys are exposed because of node compromise. In such cases a path needs to be established between

the nodes. Suppose U_u and U_v having no common key need to communicate with each other. U_u establishes communication with some node U_1 through some common key which further establishes communication with U_2 and so onwards. Let $U_u, U_1, U_2, \cdots, U_l, U_v$ be the path between U_u and U_v. Let U_u share a common key k_1 with U_1. Similarly, let U_1 share a common key k_2 with U_2, and U_{l-1} share a common key k_l with U_l and U_l share a common key k_{l+1} with U_v. U_u generates a random key K, encrypts with k_1 and sends it to U_1. U_1 decrypts K using k_1 and encrypts it using k_2 and sends it to U_2 and the process continues. Ultimately K reaches U_v using k_{l+1}. So U_v can decrypt using k_{l+1} and obtain K. K is the path key and communication between U_u and U_v is done using K. This approach has been taken in [49]. The path is found in a breadth first manner.

Also a shared key can be found by Multipath key enforcement in which multiple paths are found between the nodes U_u and U_v. Let there be p such paths. Let the path keys be k'_1, k'_2, ..., k'_p. Then the path key is calculated as $k'_1 \oplus k'_2 \oplus \cdots \oplus k'_p$. This method was first proposed by Chan, Perrig and Song [28].

7.6 Concluding remarks

In this Chapter we revisited some key predistribution schemes and presented the key establishment algorithms for the same. In the next chapter we deviate from Sensor Networks to the other interesting area of traitor tracing. We discuss several key distribution algorithms for traitor tracing. All the schemes use combinatorial designs. We will also explore a new type of product construction called *expanded product* and study its properties.

Chapter 8

Key Distribution schemes to identify all Traitors

In this Chapter we provide several explicit constructions of key distribution for traceability schemes. These constructions are based on combinatorial designs. Our constructions have the advantage that all colluders can be identified after the pirate decoder is confiscated. We present a new type of construction of merging two designs. We study the properties of the resulting design arising from such a construction and use it for traitor tracing. We also devise schemes to add more users without redistributing the keys.

Suppose there are a total of b users. The data supplier generates a base set of v keys and assigns k keys to each authorized user. These k keys form the user's set of *personal keys*. The set of personal keys of user U_i is denoted by $P(U_i)$. The data supplier gives a *message* consisting of two blocks to each user. The *enabling block* which is encrypted using some of the personal keys of the user. The *enabling block* can be decrypted using the personal keys to obtain the session key S. The *message* also consists of a *cipher block* which can be decrypted using the session key to obtain the plain text message.

Some of the authorized users (called *traitors*) can form a pirate decoder F which consists of k keys from the set of personal keys of the traitors. An unauthorized user may be able to decrypt the session key using the keys in the pirate decoder. The goal of the data supplier is that once the pirate decoder is captured and the keys contained therein are examined, at least one of the traitors will be found. If the traceability scheme can correctly identify at least one of the colluders in a coalition of c or less traitors, then the scheme is termed as a c - *traceability* scheme.

We present conditions for *c-traceability* schemes which will identify all the traitors. We call such schemes *complete traceability schemes* (CTS). Fiat and Tassa introduced *dynamic traitor tracing* in [58] that can detect all traitors if they try to broadcast the contents after it is decrypted. According to their scheme the content is broken into segments and marked so that a group of users can be traced from a re-broadcast segment. However this requires huge real time com-

putation. To reduce the real time computation Safavi-Naini and Wang proposed *sequential traitor tracing* in [141,143] where mark allocation is predetermined and independent of the re-broadcasted segment. In [158] Stinson and Wei used a traitor tracing scheme in which the exposed user is the person who contributes the maximum number of keys to the pirate decoder. Since it is a natural tendency to reduce the chance of being traced, each traitor would like to contribute as many keys to the pirate decoder as any other traitor in the group. If we properly preassign the keys, then if all colluders contribute equally, all the traitors can be found and no innocent person will be convicted. We also show that if the colluders do not contribute the same number of keys, then we get new *c-traceability* schemes which can support more number of users compared to Stinson and Wei's schemes. In this Chapter we give several explicit constructions for key distribution for traitor tracing using combinatorial designs. We also present a method which helps the data supplier to incorporate more users, without redistributing the keys.

The rest of the Chapter is organized as follows. The problem is defined in Section 8.1. We give explicit constructions for traceability schemes in Section 8.2. We present a product construction in Section 8.3. We call this an *expanded design*. We study the properties of this new design and apply it to traitor tracing schemes. In Section 8.4 we present methods to incorporate more users without redistributing the keys in the already existing users. We compare our scheme with Stinson and Wei's scheme [158] in Section 8.5.

The chapter is based on the paper [135].

8.1 Problem definition

Let the data supplier supply data to a set of $\mathcal{U} = \{U_1, U_2, \cdots, U_N\}$ of $b = N$ users. Let the base set consists of v keys. Each user U_i is given a set of k keys which form the personal keys. We denote this set by $P(U_i)$. A set $\mathcal{R} = \{U_1, U_2, \cdots, U_c\}$ of c colluders collude together and construct a pirate decoder F such that $F \subseteq \cup_{U_i \in \mathcal{C}} P(U_i)$ and $|F| = k$. The goal of the data supplier is to assign the keys to the users in such a way that once the pirate decoder is found and the keys are examined, at least one traitor will be identified.

As discussed by Chor, Fiat and Naor in [33], the traitors can by found by computing $|F \cap P(U_i)|$ for all users U_i. If $|F \cap P(U_i)| \geq |F \cap P(U_V)|$ for all users $U_V \neq U_i$, then U_i is defined to be an *exposed user*.

A traitor tracing scheme is denoted by $c\text{-}TS(v, N, k)$. We define a *complete traceability scheme* to be a traceability scheme such that if c or less colluders collude, all the traitors will be found. We denote this by *c-CTS(v, N, k)*. Complete traceability schemes were studied by Fiat and Tassa in the context of *dynamic traitor tracing* [58, 59] and by Safavi-Naini and Wang in context of *sequential traitor tracing* [141, 143].

In this chapter we will assume that all the traitors contribute the same number

Designs	key distribution
Set of elements X	Set of all key identifiers
Blocks $\mathcal{A}$	Users
Element $x \in X$	Key identifier
Elements in a Block	Set of personal keys
k	Number of personal keys given to each user
Replication number r	Number of users to whom a given key is assigned
λ	Number of users to whom a given pair of keys is assigned

Table 8.1: Mapping between set systems and key distribution system

of keys to the pirate decoder like that discussed by Wang et al in [169]. This condition is more likely to happen in practice since revealing more information will result in higher probability of being found guilty. So any traitor would like to contribute as many keys as any other traitor in the coalition. So for all traitors, $|F \cap U_1| = |F \cap U_2| = \cdots = |F \cap U_c|$ holds. In Theorems 8.2.1 and 8.2.7 we will see that when all the traitors contribute equally, then proper key assignment finds all the traitors.

8.2 Proposed Traitor Tracing schemes

We can map a set system $(X, \mathcal{A})$ (as defined in Section 1.5) to a key distribution problem as shown in Table 8.1. There are $N = b$ users, each user containing k keys as the personal keys. We recall the definition of intersection numbers from Section 1.5.

It can be seen that for a symmetric design any pair of blocks will contain λ elements in common, since $\mu = \lambda$.

Suppose a set of $\mathcal{C}$ traitors contribute to form the pirate decoder. Each traitor contributes the same number of keys to form the pirate decoder. Consider the set system $(\mathcal{X}, \mathcal{A})$ as discussed in the previous section. Let each colluder be assigned keys according to the elements in each block. This traitor tracing scheme is similar to that discussed by Stinson and Wei in [158]. We state an important theorem which helps the data supplier to correctly trace all the traitors. A similar theorem was presented for simple traitor tracing schemes (which identify at least one traitor) by Stinson and Wei in [159] and by Safavi-Naini and Wang in context of asymmetric traitor tracing schemes in [140].

Theorem 8.2.1. *If c or less colluders collude to form the pirate decoder F, such that $c|k$ and $k > \mu_{max}c^2$, then all traitors will be correctly identified and no innocent user will be convicted.*

Proof. Since $c|k$, let $w = k/c$. Let each colluder contribute w keys to form the pirate decoder F. We show that for all colluders U_i, $|F \cap P(U_i)| > |F \cap P(U_V)|$, for

all users $U_V \neq U_i$. We note that any two users can have a maximum of μ_{max} keys in common. Consider a user U_i contributing w keys to form F. Then $P(U_{V'}) \cap P(U_i)$ has at the maximum μ_{max} keys in common with any user $U_{V'} \neq U_i$. So F has at the maximum $\mu_{max}c$ keys in common with any user $U_{V'}$ who is not a traitor. Now since $w > \mu_{max}c$, $|F \cap P(U_{V'})| < w$. Now since $k > \mu_{max}c^2$, for all colluders U_i, and innocent users U_V, $|F \cap P(U_i)| > |F \cap P(V)|$. Thus all the traitors can be found and no innocent person will be convicted. ■

Note : A good traceability scheme has the following requirements.

1. Large number of users ie, large $N = b$.
2. Small size of the set of personal keys, ie, small k.
3. High resiliency, ie large c. Since c depends on the value of k and μ we look for designs with small μ_{max}.

In the rest of this Section we present several explicit construction for schemes which will identify all traitors. For a scheme having N users, the size of the personal set is $O(\sqrt{N})$ and a coalition of $O(\sqrt[4]{N})$ can be traced. In the next section we present a new design construction which we call the *Expanded design* and use it to give more tracing schemes. The new design has some interesting properties. We study these properties. We also observe that we can get a new class of PBIBD which to the best of our knowledge had not been studied earlier.

Construction TT1 We consider the following *transversal design* $TD(k, r)$ and map it to the set system $(X, \mathcal{A})$. Let r be a prime power.

1. $X = \{(x, y) : 0 \leq x < k, 0 \leq y < r\}$,
2. For all i, $G_i = \{(i, y) : 0 \leq y < r\}$,
3. $\mathcal{A} = \{A_{i,j} : 0 \leq i < r \ \& \ 0 \leq j < r\}$.

We define a block $A_{i,j}$ by

$$A_{i,j} = \{(x, xi + j \bmod r) : 0 \leq x < k\} \tag{8.2.1}$$

Let the r^2 blocks denote the set of personal keys given to each user U_i. Hence each of the r^2 users receive k keys. The i-th user receives the keys $\{(x, x\lfloor \frac{i}{r} \rfloor + (i \bmod r)) : 0 \leq x < k\}$. We note that any two blocks share a maximum of one keys. Thus $\mu_{max} = 1$. Now, by Theorem 8.2.1, if less than $\sqrt{k}$ traitors collude to form the pirate decoder, then all traitors will be identified. If k is of the order of r, then the scheme can trace a coalition of $O(\sqrt[4]{N})$ traitors, where N is the number of users. The number of keys used is $O(\sqrt{N})$. We can thus state the following theorem.

Theorem 8.2.2. *There exists a* $\sqrt{k} - CTS(rk, r^2, k)$*, where* r *is a prime.*

We will see in Section 8.4 how this can be extended to incorporate more users without redistributing keys in the existing users.

Construction TT2 Consider the symmetric design arising from the projective planes $PG(2, s)$ (for prime s) as given in [162]. We obtain the $BIBD(s^2 + s + 1, s^2 + s + 1, s + 1, s + 1; 1)$. Here $\mu_{max} = 1$. If we distribute $s + 1$ keys to each of the $s^2 + s + 1$ users, then by Theorem 8.2.1, all traitors can be found if the number of colluders is less than $\sqrt{s+1}$. Thus the scheme can trace a collusion of $O(\sqrt[4]{N})$ traitors, where N is the number of users. The number of keys used is $O(\sqrt{N})$. We can then state the following theorem.

Theorem 8.2.3. *There exists a* $\sqrt{s+1} - CTS(s^2 + s + 1, s^2 + s + 1, s + 1)$*, when* s *is a prime.*

Construction TT3 If we consider the affine plane arising from the residue of the projective plane then we obtain a design $BIBD(s^2, s^2 + s, s + 1, s; 1)$. Intersection number between two blocks is either 0 or 1. Thus $\mu_{max} = 1$. If we distribute s keys to each of the $s^2 + s + 1$ users, then by Theorem 8.2.1, all traitors can be found if the number of colluders is less than $\sqrt{s}$. Thus the scheme can trace a collusion of $O(\sqrt[4]{N})$ traitors, where N is the number of users. The number of keys used is $O(\sqrt{N})$. Thus the theorem follows.

Theorem 8.2.4. *There exists a* $\sqrt{s} - CTS(s^2, s^2 + s, s)$*, when* s *is a prime.*

Construction TT4 Consider the PBIBD $PB[2, 0, 1; n(n-1)/2]$, for $n \geq 5$, as given in [36, Page 14]. If we take the dual of the design we arrive at a scheme such that each of the $n(n-1)/2$ users is assigned $2(n-2)$ keys. $\mu_{max} = 1$. So all the traitors in a coalition of less than $\sqrt{2(n-2)}$, all the traitors can be correctly identified. We note that as the previous schemes this scheme can also trace a collusion of $O(\sqrt[4]{N})$ traitors, where N is the number of users. The number of keys used is $O(\sqrt{N})$. Thus the theorem follows.

Theorem 8.2.5. *There exists a* $\sqrt{2(n-2)} - CTS(n(n-1)(n-2)/2,\ n(n-1)/2,\ 2(n-2))$*, when* $n \geq 5$.

In all the above schemes a coalition of up to $O(\sqrt[4]{N})$ users can be traced using $O(\sqrt{N})$ keys, where N is the number of users. In all the above four constructions $\mu_{max} = 1$. Next we give an example of our scheme when $\mu_{max} = 2$. We consider the $PB[3, 1, 2; 12]$ given in [36, Page 196]. We take the dual of the design. Each of the 12 users are given a personal set of key consisting of 10 keys. The users are given the following sets of key identifiers.

$P(U_1) = \{\ 1,\ 10,\ 12,\ 13,\ 19,\ 24,\ 25,\ 32,\ 35,\ 37\ \}$
$P(U_2) = \{\ 1,\ 2,\ \ 11,\ 13,\ 14,\ 20,\ 26,\ 33,\ 36,\ 38\ \}$
$P(U_3) = \{\ 2,\ 3,\ \ 12,\ 14,\ 15,\ 21,\ 25,\ 27,\ 34,\ 39\ \}$
$P(U_4) = \{\ 1,\ 3,\ \ 4,\ \ 15,\ 16,\ 22,\ 26,\ 28,\ 35,\ 40\ \}$

$P(U_5)$ = { 2, 4, 5, 16, 17, 23, 27, 29, 36, 37 }
$P(U_6)$ = { 3, 5, 6, 17, 18, 24, 25, 28, 30, 38 }
$P(U_7)$ = { 4, 6, 7, 13, 18, 19, 26, 29, 31, 39 }
$P(U_8)$ = { 5, 7, 8, 14, 19, 20, 27, 30, 32, 40 }
$P(U_9)$ = { 6, 8, 9, 15, 20, 21, 28, 31, 33, 37 }
$P(U_{10})$= { 7, 9, 10, 16, 21, 22, 29, 32, 34, 38 }
$P(U_{11})$= { 8, 10, 11, 17, 22, 23, 30, 33, 35, 39 }
$P(U_{12})$= { 9, 11, 12, 18, 23, 24, 31, 34, 36, 40 }.

Here $\mu_{max} = 2$. According to Theorem 8.2.1 any group of two colluders contributing 5 keys each will be easily found, since they will have 5 keys in common with any pirate decoder and anyone not in the collusion will have a maximum of 4 keys. Suppose User U_1 and User U_2 collude to form a pirate encoder $F = \{10, 12, 19, 32, 35, 2, 11, 20, 33, 36\}$. Then, $|F \cap P(U_1)| = |F \cap P(U_2)| = 5$ and $|F \cap P(U_i)| < 5$, when $U_i \neq U_1, U_2$. So all the traitors will be found correctly.

Sometimes users may contribute w keys such that $w > k/c$. This happens when the keys contributed by users overlap. Since each colluder contributes more than k/c keys the following result holds.

Corollary 8.2.6. *If c or less traitors collude to form the pirate decoder F, such that each contributes w keys and $w > \mu_{max}c$, then all traitors will be correctly identified.*

We next give bound on the number of colluders c, when $c \nmid k$.

Theorem 8.2.7. *Suppose each colluder contribute w keys to form the pirate decoder. If $c\mu_{max} < w$, then all the traitors can be correctly identified and no innocent person will be convicted.*

Proof. Let each colluder contribute F_1 keys, where $|F_1| = w$ keys. Since any block shares a maximum of μ_{max} keys, $|F_1 \cap P(U_i)| \leq \mu_{max}$, where U_i is an innocent user. Since $|F| < |F_1| + |F_2| + \cdots + |F_c|$, so $|F \cap U_i| \leq c\mu_{max}$. Since $c\mu_{max} < w$, $|F \cap U_i| < w$, hence all the traitors can be found and no innocent person will be convicted. ∎

We give an example to illustrate this point. We construct $PB[10, 1, 0; 82]$. There are 41 users where each user is given a set of 10 keys. The following are the key identifiers assigned to each user.

$P(U_1)$ = { 1 , 6 , 25, 27, 33, 46, 64, 76, 77, 80 },
$P(U_2)$ = { 2, 7, 26, 28, 34, 47, 65, 77, 78, 81 },
$P(U_3)$ = { 3, 8, 27, 29, 35, 48, 66, 78, 79, 82 },
$P(U_4)$ = { 4, 9, 28, 30, 36, 49, 67, 79, 80, 42 },
$P(U_5)$ = { 5, 10, 29, 31, 37, 50, 68, 80, 81, 43 },

$P(U_6) = \{\ 6,\ 11, 30, 32, 38, 51, 69, 81, 82, 44\ \},$
$P(U_7) = \{\ 7,\ 12, 31, 33, 39, 52, 70, 82, 42, 45\ \},$
$P(U_8) = \{\ 8,\ 13, 32, 34, 40, 53, 71, 42, 43, 46\ \},$
$P(U_9) = \{\ 9,\ 14, 33, 35, 41, 54, 72, 43, 44, 47\ \},$
$P(U_{10}) = \{\ 10, 15, 34, 36, 1,\ 55, 73, 44, 45, 48\ \},$
$P(U_{11}) = \{\ 11, 16, 35, 37, 2,\ 56, 74, 45, 46, 49\ \},$
$P(U_{12}) = \{\ 12, 17, 36, 38, 3,\ 57, 75, 46, 47, 50\ \},$
$P(U_{13}) = \{\ 13, 18, 37, 39, 4,\ 58, 76, 47, 48, 51\ \},$
$P(U_{14}) = \{\ 14, 19, 38, 40, 5,\ 59, 77, 48, 49, 52\ \},$
$P(U_{15}) = \{\ 15, 20, 39, 41, 6,\ 60, 78, 49, 50, 53\ \},$
$P(U_{16}) = \{\ 16, 21, 40, 1,\ 7,\ 61, 79, 50, 51, 54\ \},$
$P(U_{17}) = \{\ 17, 22, 41, 2,\ 8,\ 62, 80, 51, 52, 55\ \},$
$P(U_{18}) = \{\ 18, 23, 1,\ 3,\ 9,\ 63, 81, 52, 53, 56\ \},$
$P(U_{19}) = \{\ 19, 24, 2,\ 4,\ 10, 64, 82, 53, 54, 57\ \},$
$P(U_{20}) = \{\ 20, 25, 3,\ 5,\ 11, 65, 42, 54, 55, 58\ \},$
$P(U_{21}) = \{\ 21, 26, 4,\ 6,\ 12, 66, 43, 55, 56, 59\ \},$
$P(U_{22}) = \{\ 22, 27, 5,\ 7,\ 13, 67, 44, 56, 57, 60\ \},$
$P(U_{23}) = \{\ 23, 28, 6,\ 8,\ 14, 68, 45, 57, 58, 61\ \},$
$P(U_{24}) = \{\ 24, 29, 7,\ 9,\ 15, 69, 46, 58, 59, 62\ \},$
$P(U_{25}) = \{\ 25, 30, 8,\ 10, 16, 70, 47, 59, 60, 63\ \},$
$P(U_{26}) = \{\ 26, 31, 9,\ 11, 17, 71, 48, 60, 61, 64\ \},$
$P(U_{27}) = \{\ 27, 32, 10, 12, 18, 72, 49, 61, 62, 65\ \},$
$P(U_{28}) = \{\ 28, 33, 11, 13, 19, 73, 50, 62, 63, 66\ \},$
$P(U_{29}) = \{\ 29, 34, 12, 14, 20, 74, 51, 63, 64, 67\ \},$
$P(U_{30}) = \{\ 30, 35, 13, 15, 21, 75, 52, 64, 65, 68\ \},$
$P(U_{31}) = \{\ 31, 36, 14, 16, 22, 76, 53, 65, 66, 69\ \},$
$P(U_{32}) = \{\ 32, 37, 15, 17, 23, 77, 54, 66, 67, 70\ \},$
$P(U_{33}) = \{\ 33, 38, 16, 18, 24, 78, 55, 67, 68, 71\ \},$
$P(U_{34}) = \{\ 34, 39, 17, 19, 25, 79, 56, 68, 69, 72\ \},$
$P(U_{35}) = \{\ 35, 40, 18, 20, 26, 80, 57, 69, 70, 73\ \},$
$P(U_{36}) = \{\ 36, 41, 19, 21, 27, 81, 58, 70, 71, 74\ \},$
$P(u_{37}) = \{\ 37, 1,\ 20, 22, 28, 82, 59, 71, 72, 75\ \},$
$P(U_{38}) = \{\ 38, 2,\ 21, 23, 29, 42, 60, 72, 73, 76\ \},$
$P(U_{39}) = \{\ 39, 3,\ 22, 24, 30, 43, 61, 73, 74, 77\ \},$
$P(U_{40}) = \{\ 40, 4,\ 23, 25, 31, 44, 62, 74, 75, 78\ \},$
$P(U_{41}) = \{\ 41, 5,\ 24, 26, 32, 45, 63, 75, 76, 79\ \}.$

Suppose U_1 contributes the key identifiers $1, 6, 76, 77$, U_2 contributes $2, 7, 77, 28$ and U_{24} contributes $7, 9, 62, 69$. Then $F = \{1, 2, 6, 7, 9, 28, 62, 69, 76, 77\}$. Then we can see that all the traitors can be correctly identified, since for any user $U_i \neq U_1, U_2, U_{24}$, $|F \cap P(U_i)| < 4$.

We next give some schemes of key distribution using generalized quadrangles. We give three constructions using $GQ(q, q)$, $GQ(q, q^2)$ and $GQ(q^2, q^3)$. We present

the algorithm for key distribution using $GQ(q,q)$ in Algorithm 8. This construction uses $PG(4,q)$. From definition of generalized quadrangles, we note that any two

Algorithm 8 Key distribution using $GQ(q,q)$

1: Find minimum prime q s.t. $(q+1)(q^2+1) \geq N$;
2: Find the points $x =< x_0, x_1, x_2, x_3, x_4 >$ in $GF(q)$ which satisfy the equation $x_0^2 + x_1x_2 + x_3x_4 = 0$.
3: For each point x, generate the set of points collinear to x. There are $v = (q+1)(q^2+1)$ non-collinear points. These v points form the base set.
4: Construct $b = (q+1)(q^2+1)$ lines with $q+1$ points.
5: Assign $q+1$ personal keys to each of the b users as calculated in the previous step.

blocks share at most one key. From the above algorithm and Theorem 8.2.8 we get the following tracing scheme.

Theorem 8.2.8. *There exists a* $\sqrt{q+1}-CTS(q^3+q^2+q+1, q^3+q^2+q+1, q+1)$, *where* q *is a prime power.*

We use two other constructions of generalized quadrangles $GQ(q,q)$ using $PG(5,q)$ and $GQ(q^2,q^3)$ using $H(4,q^2)$. Considering the parameters of the designs $GQ(q,q^2)$ and $GQ(q^2,q^3)$, we get the following two tracing schemes.

Theorem 8.2.9. *There exists a* $\sqrt{q+1}-CTS(q^4+q^3+q+1, q^5+q^3+q^2+1, q+1)$, *where* q *is a prime power.*

Theorem 8.2.10. *There exists a* $\sqrt{q^2+1}-CTS(q^8+q^5+q^3+1, q^7+q^5+q^2+1, q^2+1)$, *where* q *is a prime power.*

8.3 A new design construction: Expanded Design

Motivated by Kronecker product of designs as discussed by Vartak in [166] and our requirement to support large number of users (ie, large values of the b) we define a new construction called *expanded design*. Let us consider two set systems $D_1 = (X_1, \mathcal{A}_1)$ and $D_2 = (X_2, \mathcal{A}_2)$ such that $X_1 = \{x_{11}, x_{12}, \cdots, x_{1v_1}\}$, $X_2 = \{x_{21}, x_{22}, \cdots, x_{2v_2}\}$ $\mathcal{A}_1 = \{A_{11}, A_{12}, \cdots, A_{1b_1}\}$, $\mathcal{A}_2 = \{A_{21}, A_{22}, \cdots, A_{2b_2}\}$. Let $M_1 = [m'_{ij}]$ and $M_2 = [m''_{ij}]$ be the respective incidence matrices. Therefore the dimension of M_1 and M_2 are $v_1 \times b_1$ and $v_2 \times b_2$ respectively. Assume that $k_1 = v_2$. We construct the matrix M in the following way.

1. For every column j of M_1 replace m'_{ij} by a row of M_2, if $m'_{ij} = 1$.
2. For every column j of M_1 replace m'_{ij} by a row vector of length b_2 containing all zeros, if $m'_{ij} = 0$.

The result of the following operations is a matrix M of dimension $v_1 \times b_1 b_2$. We denote this design by D and represent D by $D = D_1 \bowtie D_2$. We call this design an *Expanded Design*.

Example 8.3.1. *Consider the designs* $D_1 = (7, 7, 3, 3, 1)$ *and* $D_2 = (3, 3, 2, 2, 1)$ *[37]. Then we arrive at the matrices which are given below.*

$$D_1 = \begin{matrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{matrix} \quad \text{and } D_2 = \begin{matrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{matrix}$$

$$D_1 \bowtie D_2 =$$

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

We first study the properties of this design and in next subsection we apply it to traitor tracing.

Theorem 8.3.2. *If* D_1 *and* D_2 *are two* $BIBDs$ *with parameters* v_1, b_1, r_1, k_1, λ_1 *and* v_2, b_2, r_2, k_2, λ_2, *such that* $v_2 = k_1$ *then* $D_1 \bowtie D_2$ *is a* $BIBD$ *with parameters* $v = v_1$, $b = b_1 b_2$, $r = r_1 r_2$, $k = k_2$, $\lambda = \lambda_1 \lambda_2$.

Proof. According to the construction we see that $v = v_1$, $b = b_1 b_2$ and $k = k_2$. For each 1 in row of M_1 we replace it by a row of M_2 which contains r_2 1s. Since there are r_1 1s in a row of M_1, the number of 1s in a row of M is $r_1 r_2$. Thus $r = r_1 r_2$. Let us consider a pair of elements x and y. Consider the columns c_1, c_2, $\cdots$, c_{λ_1} such that $m'_{xc_i} = 1$ and $m'_{yc_i} = 1$, for $i = 1, 2, \cdots, \lambda_1$. Since all pairs of elements occur together in λ_2 columns in M_2, for each column c_i, x and y occur λ_2 times. So for all λ_1 columns c_1, c_2, $\cdots$, c_{λ_1} x and y elements occur together in $\lambda_1 \lambda_2$ columns in M. So $\lambda = \lambda_1 \lambda_2$. ∎

The next design results in a PBIBD, which to the best of our knowledge has not been studied earlier.

Theorem 8.3.3. *If* D_1 *is a* $PBIBD$ $PB[k_1; \lambda_{11}, \lambda_{12}, \cdots, \lambda_{1m}; v_1]$ *and* D_2 *is a* $BIBDs$ *with parameters* v_2, b_2, r_2, k_2, λ_2, *then* $D_1 \bowtie D_2$ *is a* $PBIBD$ $PB[k_2; \lambda_{11}\lambda_2, \lambda_{12}\lambda_2, \cdots, \lambda_{1m}\lambda_2; v_1]$.

The proof is similar to the proof of Theorem 8.3.2. The following two theorems follow similarly.

Theorem 8.3.4. *Consider the $TD(r', k')$ and $BIBD(n^2+n+1, n^2+n+1, n+1, n+1, \lambda)$, such that $k' = n^2+n+1$, then the expanded design is a group divisible PBIBD having parameters, $v = r'k'$, $b = r'^2k'$, $r = r'(n+1)$, $k = n+1$, $\lambda = 0, 1$. The parameters of the designs $n_1 = r'-1$, $n_2 = r'k'-r'$, $\lambda_1 = 0$ $\lambda_2 = 1$. The matrices are represented by*

$$P_1 = \begin{pmatrix} r'-2 & 0 \\ 0 & r'(k'-1) \end{pmatrix},\ P_2 = \begin{pmatrix} 0 & r'-1 \\ r'-1 & r'(k'-2) \end{pmatrix}.$$

Theorem 8.3.5. *Consider two TDs $TD(r_1, k_1)$ and $TD(r_2, k_2)$, such that $k_1 = r_2k_2$, then the expanded design is a group divisible PBIBD having parameters, $v = r_1k_1$, $b = r_1^2r_2^2$, $r = r_1r_2$, $k = k_2$, $\lambda = 0, 1$. The parameters of the designs $n_1 = r_1r_2 - 1$, $n_2 = r_1k_1 - r_1r_2$, $\lambda_1 = 0$ $\lambda_2 = 1$. The matrices are represented by*

$$P_1 = \begin{pmatrix} r_1r_2-2 & 0 \\ 0 & r_1r_2(k_2-1) \end{pmatrix},\ P_2 = \begin{pmatrix} 0 & r_1r_2-1 \\ r_1r_2-1 & r_1r_2(k_2-2) \end{pmatrix}.$$

Consider the following example.

Example 8.3.6. *Consider the designs $D_1 = PB[4; 0, 1; 10]$ and $D_2 = (4, 4, 3, 3, 2)$. So $v_1 = 10$, $b_1 = 5$, $r_1 = 2$, $k_1 = 4$, $\lambda_{11} = 0$, $\lambda_{12} = 1$, $v_1 = 4$, $b_1 = 4$, $r_1 = 3$, $k_1 = 3$, $\lambda_2 = 2$. Then we arrive at the matrices which are given below.*

$$D_1 = \begin{matrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{matrix} \quad \text{and } D_2 = \begin{matrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{matrix}$$

$$D_1 \bowtie D_2 =$$

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

We see that $v = 10$, $b = 20$, $r = 6$, $k = 3$, $\lambda_1 = 0$ and $\lambda_2 = 2$.

The Expanded Design can however result in two blocks having identical set of elements. This is shown in the following example.

Example 8.3.7. *Consider the designs* $D_1 = (8, 14, 7, 4, 3)$ *and* $D_2 = (4, 4, 3, 3, 2)$. *Then we arrive at the matrices which are given below.*

$$D_1 = \begin{matrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{matrix} \quad \text{and } D_2 = \begin{matrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{matrix}.$$

For columns say 1 *and* 2 *of* D_1, *we get eight columns in* $D_1 \bowtie D_2$ *as*

$$\begin{matrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix} \quad \text{and} \quad \begin{matrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}.$$

.

So we see that two blocks will be identical.

The following theorem imposes some restriction on the choice of the parameters of the design to meet this demand.

Theorem 8.3.8. *Let* $D_1 = (v_1, b_1, r_1, k_1, \lambda_1)$ *and* $D_2 = (v_2, b_2, r_2, k_2, \lambda_2)$ *be two BIBDs. Let* μ_1, μ_2, $\cdots$, μ_t *be the intersection numbers of any two blocks in* D_1. *Let* $\mu = max\{\mu_i : i = 1, 2, \cdots, t\}$. *If* $\mu < k_2$ *then, no two blocks will have the same set of keys in them.*

Proof. Each column gives rise to b_2 columns. The b_2 blocks corresponding to these columns will be different since the blocks in D_2 are different. So we consider blocks which arise out of two different columns of D_1. Let B_1 and B_2 be two blocks in D_1 which share μ elements in common. So the matrix M_1 will have μ rows where both the columns B_1 and B_2 will have ones in them. Let these rows be n_1, n_2, $\cdots$, n_μ. When we construct the matrix M, for each of these k_1 ones of column B_1 and B_2 we substitute a row of M_2. Let for the μ rows n_1, n_2, $\cdots$, n_μ we have ones in the same rows in two columns C_1 arising from B_1 and column C_2 arising from B_2

of the matrix M. Since $\mu < k_2$ there are some other rows in the block B_1 which will give a 1 in column C_1 but not in C_2. Similarly, there are some other rows in the block B_2 which will give a 1 in column C_2 but not in C_1. Hence the two nodes arising from C_1 and C_2 can never have the same keys. This happens for every pair of columns in the resulting matrix M. Hence none of the blocks in M have the same elements. ∎

Note: If μ_{max_1} and μ_{max_2} be the maximum intersection numbers of D_1 and D_2, then $\mu_{max} \leq max(\mu_{max_1}, \mu_{max_2})$.

8.3.1 Application of Expanded Design to Traitor Tracing

We can apply the Expanded Design to Traitor tracing in such a a way that v is the size of the base set, N is the maximum number of users that the system can support. The key distribution algorithm and traitor tracing schemes are similar to that discussed in in Section 8.2. From the Theorems 8.3.2, 8.3.3, 8.3.4, 8.3.5 we state the following theorems.

Theorem 8.3.9. *If there exists two BIBDs* $BIBD(v_1, b_1, r_1, k_1, \lambda_1)$ *and* $BIBD(v_2, b_2, r_2, k_2, \lambda_2)$*, such that* $v_2 = k_1$*, then there exists a* $\sqrt{\frac{k_2}{\mu_{max}}} - CTS(v_1, b_1 b_2, k_2)$*, where* μ_{max} *is the maximum intersection number of the expanded design obtained from the BIBDs.*

Theorem 8.3.10. *If there exists PBIBD with parameters* v_1, b_1, r_1, k_1, λ_{11}, λ_{12}, $\cdots$, λ_{1m} *and* $BIBD(v_2, b_2, r_2, k_2, \lambda_2)$*, such that,* $v_2 = k_1$*, then there exists a* $\sqrt{\frac{k_2}{\mu_{max}}} - CTS(v_1, b_1 b_2, k_2)$*, where* μ_{max} *is the maximum intersection number of the expanded design obtained from the PBIBD and BIBD.*

Theorem 8.3.11. *If there exists TD* $TD(r_1, k_1)$ *and* $BIBD(n^2 + n + 1, n^2 + n + 1, n + 1, n + 1, 1)$*, such that* $n^2 + n + 1 = k_1$*, then there exists a* $\sqrt{\frac{k_2}{\mu_{max}}} - CTS(r_1 k_1, r_1^2(n^2 + n + 1), k_2)$*, where* μ_{max} *is the maximum intersection number of the expanded design obtained from the TD and the BIBD.*

Theorem 8.3.12. *If there exists two TD* $TD(r_1, k_1)$ *and* $TD(r_2, k_2)$*, such that* $r_2^2 = k_1$*, then there exists a* $\sqrt{\frac{k_2}{\mu_{max}}} - CTS(r_1 k_1, r_1^2 r_2^2, k_2)$*, where* μ_{max} *is the maximum intersection number of the expanded design obtained from the TDs.*

We consider a few general constructions that can be used for key distribution. Suppose q^2+q+1 is a prime power. Let $v_1 = (q^2+q+1)(q^2+q+2)$, $b_1 = (q^2+q+1)^2+1$, $r_1 = q^2 + q$, $k_1 = q^2 + q + 1$, $\lambda_{11} = 0$, $\lambda_{12} = 1$ (using the design in [36, Page 12]) be the parameters of a PBIBD D_1, and let $D_2 = (q^2+q+1, q^2+q+1, q+1, q+1, 1)$ be a BIBD. Then we get a design $D = D_1 \bowtie D_2$ such that the number of elements is $(q^2 + q + 1)(q^2 + q + 2)$, number of blocks is $((q^2 + q + 1)^2 + 1)(q^2 + q + 1)$ and

number of elements in each block is $q+1$. When we use this design we see that we can support $((q^2+q+1)^2+1)(q^2+q+1)$ users. $\mu_{max}=1$. Each user has a set of $q+1$ personal keys. By Theorem all traitors can be found provided a coalition of $\sqrt{q+1}$ exists. We arrive at the following theorem.

Theorem 8.3.13. *There exists a* $\sqrt{q+1}-CTS((q^2+q+1)(q^2+q+2),((q^2+q+1)^2+1)(q^2+q+1),q+1)$*, where* q *and* q^2+q+1 *are prime powers.*

Consider the transversal design given in Lemma 8 [162, Section 6.3]. Let $D_1=(rq^2,q^4,r,q^2,1)$ and $D_2=(q^2,q(q+1),q+1,q,1)$. By Theorem we can construct a key distribution system which supports $q^5(q+1)$ users. Each user has a personal set consisting of q keys. The size of the base set is rq^2. By Theorem 8.3.2 all traitors can be found provided a coalition of $\sqrt{q}$ exists. We arrive at the following theorem.

Theorem 8.3.14. *There exists a* $\sqrt{q}-CTS(rq^2,q^5(q+1),q)$*, where* q *and* r *are prime powers.*

We note the following important facts about our *expanded design.*
Note 1: The design $D_1 \bowtie D_2$ depends upon the order in which the rows are selected from M_2 when D_2 is not a $BIBD$.
Note 2: In case D_2 is a $PBIBD$, $0 \leq \lambda \leq \lambda_1\lambda_2$ depending upon the order in which the rows are selected from the incidence matrix M_2 of D_2.

8.4 Introducing more users

We can introduce more users into the system without redistributing the keys. An embedding scheme appears in [158]. We give three new constructions to make our schemes scalable.

Construction S1: Consider the $PB[s+1,0,1;s^2+s]$. We first observe by Theorem 8.2.1 that if we distribute the keys according to the design $BIBD(s^2+s+1,s^2+s+1,s+1,s+1,1)$, then all the traitors can be found. If we remove any arbitrary element we get a $PB[s+1,0,1;s^2+s]$. It follows from Theorem 8.2.1, that distributing the keys according to this PBIBD will enable the data distributer to trace all the traitors.

So initially if the data distributor distributes keys to s^2 users according to the $PBIBD$, then there will be scope to add $s+1$ more users without redistributing the keys. The following example illustrates this scheme.

Example 8.4.1. *Initially let the data distributor distribute* 30 *keys to* 25 *users, such that each user receives* 6 *keys.*

$P(U_1) = \{0, 1, 6, 18, 22, 29\}$, $P(U_2) = \{0, 2, 3, 8, 20, 24\}$,
$P(U_3) = \{1, 3, 4, 9, 21, 25\}$, $P(U_4) = \{2, 4, 5, 10, 22, 26\}$,
$P(U_5) = \{3, 5, 6, 11, 23, 27\}$, $P(U_6) = \{4, 6, 7, 12, 24, 28\}$,
$P(U_7) = \{5, 7, 8, 13, 25, 29\}$, $P(U_8) = \{0, 7, 9, 10, 15, 27\}$,
$P(U_9) = \{1, 8, 10, 11, 16, 28\}$, $P(U_{10}) = \{2, 9, 11, 12, 17, 29\}$,
$p(U_{11}) = \{0, 4, 11, 13, 14, 19\}$, $P(U_{12}) = \{1, 5, 12, 14, 15, 20\}$,
$P(U_{13}) = \{2, 6, 13, 15, 16, 21\}$, $P(U_{14}) = \{3, 7, 14, 16, 17, 22\}$,
$P(U_{15}) = \{4, 8, 15, 17, 18, 23\}$, $P(U_{16}) = \{5, 9, 16, 18, 19, 24\}$,
$P(U_{17}) = \{6, 10, 17, 19, 20, 25\}$, $P(U_{18}) = \{7, 11, 18, 20, 21, 26\}$,
$P(U_{19}) = \{8, 12, 19, 21, 22, 27\}$, $P(U_{20}) = \{9, 13, 20, 22, 23, 28\}$,
$P(U_{21}) = \{10, 14, 21, 23, 24, 29\}$, $P(U_{22}) = \{0, 12, 16, 23, 25, 26\}$,
$P(U_{23}) = \{1, 13, 17, 24, 26, 27\}$, $P(U_{24}) = \{2, 14, 18, 25, 27, 28\}$,
$P(U_{25}) = \{3, 15, 19, 26, 28, 29\}$.

We can introduce the following 6 *users by assigning them the following keys*

$P(U_{26}) = \{1, 2, 7, 19, 23, 30\}$, $P(U_{27}) = \{6, 8, 9, 14, 26, 30\}$,
$P(U_{28}) = \{3, 10, 12, 13, 18, 30\}$, $P(U_{29}) = \{11, 15, 22, 24, 25, 30\}$,
$P(U_{30}) = \{4, 16, 20, 27, 29, 30\}$, $P(U_{31}) = \{0, 5, 17, 21, 28, 30\}$.

Construction S2: We refer to Construction TT1. We use a similar construction. Initially the data distributor has a key pool of size r^2. The data distributor distributes $k = (r-1)/2$ keys to each of the r^2 users in the following way. The i-th user receives the keys $\{(x + (i \bmod r), (x + (i \bmod r))\lfloor \frac{i}{r} \rfloor + i \bmod r) : 0 \leq x \leq (r-3)/2\}$. It can be shown that any two users will have either 0 or 1 key in common. Thus we get a $\sqrt{\frac{r-1}{2}} - CTS(r^2, r^2, k)$.

Now when more users come in, she can double the number of users by distributing the keys to the new users in a way such that the i'-th user receives the keys $\{(x+((i'-r^2) \bmod r), (x+((i'-r^2) \bmod r))\lfloor \frac{i'-r^2}{r} \rfloor + (i'-r^2) \bmod r) : (r-1)/2 \leq x < r-1\}$. So each of the $2r^2$ users receive k keys and no key redistribution is required for the initial users. We also note that since any two of the $2r^2$ users share at the maximum of one key, the final scheme is a $\sqrt{\frac{r-1}{2}} - CTS(r^2, 2r^2, \frac{r-1}{2})$.

Construction S3: According to the construction S2 we can increase the number of users to twice the original number. We now show a construction where we can increase the number of users to t times, where $t = \lfloor \frac{r-1}{k} \rfloor$. We use a similar construction. Initially the data distributor has a key pool of size r^2. The data distributor distributes $k = \lfloor \frac{r-1}{t} \rfloor$ keys to each of the r^2 users in the following way. The u-th user receives the keys $\{(x + (u \bmod r)), (x + (u \bmod r))\lfloor \frac{u}{r} \rfloor + u \bmod r) : 0 \leq x < \lfloor \frac{r-1}{k} \rfloor\}$. It can be shown that any two users will have either 0 or 1 key in common. Thus we get a $\sqrt{k} - CTS(r^2, r^2, k)$. Now when more users come in, she can double the number of users by distributing the keys such that the u'-th user receives the keys. $\{(x + (u' - r^2) \bmod r), (x + ((u' - r^2) \bmod r))\lfloor \frac{u'-r^2}{r} \rfloor + (u' - r^2) \bmod r) : \lceil (r-1)/k \rceil \leq x < \lfloor 2(r-1)/k \rfloor\}$. When more users come in, she

can triple the number of users by distributing the keys such that the u'-th user receives the keys $\{(x + (u' - 2r^2) \bmod r), (x + (u' - 2r^2) \bmod r)\lfloor \frac{u'-2r^2}{r} \rfloor + (u' - 2r^2) \bmod r) : \lceil 2(r-1)/k \rceil \leq x < \lfloor 3(r-1)/k \rfloor\}$. Similarly we can increase the number of user to t times, where the u'-th user receives the keys $\{(x + (u' - (t-1)r^2) \bmod r), (x + (u' - (t-1)r^2) \bmod r)\lfloor \frac{u'-(t-1)r^2}{r} \rfloor + (u' - (t-1)r^2) \bmod r) : \lceil (t-1)(r-1)/k \rceil \leq x < \lfloor t(r-1)/k \rfloor\}$. Here $t = \lfloor (r-1)/k \rfloor$. The final scheme results in a $\sqrt{\frac{r-1}{t}} - CTS(r^2, tr^2, k)$.

8.5 Comparison with Stinson and Wei's Traceability Schemes [158, 159]

In [158] Stinson and Wei proposed some traceability schemes using set system. In [159] Stinson and Wei presented the condition for constructing traceability schemes. They stated the following theorem.

Theorem 8.5.1. *[159] Suppose there exists a set system $(X, \mathcal{A})$ satisfying the following conditions:*

1. $|A| = k \geq c^2\mu + 1$ *for any* $A \in \mathcal{A}$,
2. $|A_i \bigcap A_j| \leq \mu$ *for any* A_i, $A_j \in \mathcal{A}$, $i \neq j$.

Then the set system is a c-traceability scheme.

They obtained the following traceability schemes.

1. There exists a $\lfloor \sqrt{\frac{k-1}{t-1}} \rfloor - TS(v, \binom{v}{t}/\binom{k}{t}, k)$.
2. For a prime power q, there exists a $\lfloor \sqrt{\frac{q}{2}} \rfloor - TS(q^2+q+1, q^2+q+1, q+1)$.
3. For a prime power q, there exists a $\lfloor \sqrt{\frac{q}{2}} \rfloor - TS(q^2+1, q^3+q, q+1)$.
4. For a prime power q, there exists a $\lfloor \sqrt{\frac{q}{t+1}} \rfloor - TS(q^2+q, q^t, q+1)$.

If we use our constructions without the restriction that all the traitors contribute equally, we arrive at the following traceability schemes.

1. For a prime power q, there exists a $\sqrt{q} - TS(q^3+q^2+q+1, q^3+q^2+q+1, q+1)$.
2. For a prime power q, there exists a $\sqrt{q} - TS(q^4+q^3+q^2+1, q^5+q^3+q^2+1, q+1)$.
3. For a prime power q, there exists a $q - TS(q^8+q^5+q+1, q^7+q^5+q^2+1, q^2+1)$.
4. For a prime power q, there exists a $\sqrt{q} - TS((q^2+q+1)(q^2+q+2), ((q^2+q+1)^2+1)(q^2+q+1), q+1)$.

5. For a prime power q, there exists a $\sqrt{q-1} - TS(rq^2, q^5(q+1), q)$.

We see that in all the above cases our schemes support more users than Stinson and Wei's scheme. The number of colluders that can be found is also large in our case. The number of keys required is however equal. So we obtain traceability schemes better than Stinson and Wei.

8.6 The case when colluders do not contribute the same number of keys

We have assumed in the chapter that all the colluders contribute equally to the pirate decoder. This is a restrictive assumption, although it is possible in reality. This is because when a colluder contribute more number of keys to the pirate decoder, then she has a higher probability of being caught. However the other condition has been addressed by several researcher. So we consider the case where colluders violate this rule and contribute unequal number of keys. In the following theorem we derive a condition such that only the colluders can be detected and no innocent person will be convicted.

Theorem 8.6.1. *Suppose the colluder U_i contribute w_i (for $i = 1, 2, \ldots, c$) keys to form the pirate decoder. If there are a maximum of c colluders such that $c\mu_{max} < min\{w_1, w_2, \ldots, w_c\}$, then all the traitors can be correctly identified and no innocent person will be convicted.*

Proof. Let $c' \leq c$ colluders collude to form the pirate decoder. Suppose user U_i contribute the set of set F_i such that $|F_i| = w_i$ keys to the pirate decoder. Since any two users may contribute the same keys, $|F| \leq w_1 + w_2 + \cdots w_{c'}$. Since any user shares a maximum of μ_{max} keys, $|F_i \cap P(U_v)| \leq \mu_{max}$, where U_v $(v \neq 1, 2, \ldots, c')$ is an innocent user. Now, $|F| \leq |F_1| + |F_2| + \cdots + |F_{c'}|$, so $|F \cap U_v| \leq c'\mu_{max} \leq c\mu_{max}$. Since $c\mu_{max} < w_i$, $|F \cap U_v| < |F \cap U_i|$, hence the traitors can be found and no innocent person will be convicted. ∎

8.7 Concluding remarks

In this paper we give some explicit construction of key distribution for traceability schemes which will identify all traitors once the pirate decoder is inspected. We also introduce and study expanded designs. We conclude this thesis with a few open problems in the next Chapter.

Chapter 9

Conclusion

There are a lot of problems that can be worked on in future. Till date combinatorial designs like PBIBD, TD, GQ, 3-designs, Costas Arrays, have been used for key predistribution in sensor networks. The most obvious question is that can we use other designs for predistribution? Though some work on packing designs [30] and cover free families [26] have been studied, a lot of designs remain unexploited. We have observed in this thesis that many techniques that apply to key predistribution in sensor networks also apply to traitor tracing. So one way will be to try to use the different combinatorial structures used for traitor tracing to design efficient key distribution schemes for sensor networks. Also the study of these designs motivate us to explore the possibility of new designs. One work in this direction was the study of expanded product. The study motivates us to find new design which were unknown so far and contribute of the vast wealth of combinatorial designs.

Apart from these general questions, we observe in Chapter 3 that we could scale the system to twice its original size. The question is can we iteratively use the PBIBD designs to increase the scalability of the network?

In Chapter 4 though we present a general construction of key predistribution schemes using codes, we mention particularly Reed-Solomon codes. In future we would like to explore the possibility of other codes and choose the one which is best suited for sensor networks. In this regards we mention that we would like to find codes with the following properties

- Codes with large number of codewords
- Codes with large distance

We would also like to find non-binary codes where the distance between any two codewords is not exactly n, where n is the length of the code. This will ensure that any two nodes are directly connected.

In Chapter 5 we have considered a two dimensional grid-based deployment scheme. We note that our grid-based approach may be extended to n-dimensional or hypercube based schemes. This requires further investigation.

Chapter 6 assumes the deployment region to be divided into square regions. One question is to find the optimum number of agents if the deployment region is irregularly divided.

We have defined $E(s)$ and $V(s)$ as parameters of resiliency. We can actually define a new parameter $E_{bound}(s)$ as the maximum of all the values of $E(s)$ taken over all possible combinations of s compromised nodes. There are $\binom{N}{s}$ ways of selecting s nodes that are to be compromised. Let s_i denote the i-th such selection, and $S = \{s_1, s_2, \ldots, s_{\binom{N}{s}}\}$ be the set of all the possible combinations of s nodes. Let $E_{s_i}(s)$ be the fraction of links disconnected when the set of nodes s_i is compromised. Then $E_{bound}(s)$ is defined as

$$E_{bound}(s) = \max_{s_i \in S} E_{s_i}(s)$$

Similarly we can define the parameter $V_{bound}(s)$ as the maximum of all the values of $V(s)$ taken over all possible combinations of s compromised nodes. Mathematically,

$$V_{bound}(s) = \max_{s_i \in S} V_{s_i}(s),$$

where $V_{s_i}(s)$ be the fraction of nodes disconnected when the set of nodes s_i is compromised. These security parameters need to be studied.

Two more security parameters were presented by Le Porquier De Vaux and Dumain [167]. They defined s_v and s_v^*, where s_v is the threshold value of s for which $V(s) = 0$. If $s > s_v$, sensor nodes are certainly disconnected. In sensor network, it is always interesting to know how resistant a network is and how many nodes can be compromised without disconnecting any node in an area. s_v^* has been defined as

$$s_v^* = \frac{s_v}{\text{Number of nodes in the network}}.$$

It must be noted that s_v and s_v^* do not depend of location of the s compromised nodes.

Two other measures of resiliency have been proposed by Pal Choudhury et al [35]. They define $E'(s)$ (Not to be confused with the definition of $E'(s)$ given in Table 6.1 given in Chapter 6) and $d(s)$. $E'(s)$, which is defined as the ratio of the number of links remaining after s nodes are compromised and the number of links present among the non-compromised nodes before compromise. We observe that $E'(s) = [E(s) \times \binom{b}{2}]/\binom{(b-s)}{2}$. $E'(s)$ also makes sense intuitively as it measures the effect the s compromised nodes have on the remaining non-compromised nodes. $d(s)$, has been defined as the number of distinct, disjoint cluster formed, when s nodes are compromised. $d(s) = 1$, if every node can communicate with all other nodes in a single hop or by establishing a path. A future work will be to study the resiliency of our schemes taking all these parameters into account.

Though we have presented key establishment algorithms (Chapter 7), there are a few questions which remain unsolved. During path key establishment, the path key is known to all the intermediate nodes. So if the intermediate nodes are compromised, then the path key is known to the attacker. Is there a way to prevent this? One way is to use multipath key enforcement (Discussed in Chatper 7), which decreases the probability of such a compromise. However multipath enforcement results in large communication delays. So is there a better method to ensure the secrecy of the path key?

In traitor tracing schemes (Chapter 8) several more questions need to be answered. Firstly we have assumed that each user contributes equally to the pirate decoder. Though this is a practical choice, sometimes users may not contribute equally to the pirate decoder. Are there combinatorial methods of predistribution of keys, such that even if the users do not contribute the same number of keys to the pirate decoder, we can trace all traitors. Another question is to design new traitor tracing schemes using combinatorial techniques.

One outcome of the thesis is the motivation to study designs and find out new designs which have not been constructed so far. Also in the above applications of sensor networks and traitor tracing we are interested in finding the number of common keys between two nodes or two users. So the obvious question would be to study the number of elements common between two blocks in any combinatorial designs. A whole lot of research in this area remains open. We hope that the interesting and nice structures of the designs will help us in this pursuit.

Bibliography

[1] Tinyos. Available at http://www.tinyos.net. Last assessed on January 22, 2009.

[2] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.

[3] Mohammad Al-Shurman and Seong-Moo Yoo. Key pre-distribution using mds codes in mobile ad hoc networks. In *ITNG*, pages 566–567. IEEE Computer Society, 2006.

[4] Adam Barth, Dan Boneh, and Brent Waters. Privacy in encrypted content distribution using private broadcast encryption. In Giovanni Di Crescenzo and Aviel D. Rubin, editors, *Financial Cryptography*, volume 4107 of *Lecture Notes in Computer Science*, pages 52–64. Springer, 2006.

[5] Omer Berkman, Michal Parnas, and Jiri Sgall. Efficient dynamic traitor tracing. In *SODA*, pages 586–595, 2000.

[6] Omer Berkman, Michal Parnas, and Jiri Sgall. Efficient dynamic traitor tracing. *SIAM J. Comput.*, 30(6):1802–1828, 2000.

[7] Simon R. Blackburn, Tuvi Etzion, Keith M. Martin, and Maura B. Paterson. Efficient key predistribution for grid-based wireless sensor networks. In Reihaneh Safavi-Naini, editor, *ICITS*, volume 5155 of *Lecture Notes in Computer Science*, pages 54–69. Springer, 2008.

[8] Simon R. Blackburn, Tuvi Etzion, Keith M. Martin, and Maura B. Paterson. Key predistribution techniques for grid-based wireless sensor networks, 2009. Available at ePrint Cryptology archive 2009/014.

[9] Rolf Blom. An optimal class of symmetric key generation systems. In *Proceeding of EUROCRYPT*, pages 335–338, 1984.

[10] Carlo Blundo, Alfredo De Santis, Amir Herzberg, Shay Kutten, Ugo Vaccaro, and Moti Yung. Perfectly-secure key distribution for dynamic conferences. *In*

Advances in Cryptology: Proceedings of CRYPTO'92, Santa Barbara, CA, Lecture Notes in Computer Science, 740:471–486, 1993.

[11] Dan Boneh and Matthew K. Franklin. An efficient public key traitor tracing scheme. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 338–353. Springer, 1999.

[12] Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertext. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM Conference on Computer and Communications Security*, pages 501–510. ACM, 2008.

[13] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 573–592. Springer, 2006.

[14] Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data (extended abstract). In Don Coppersmith, editor, *CRYPTO*, volume 963 of *Lecture Notes in Computer Science*, pages 452–465. Springer, 1995.

[15] Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data. *IEEE Transactions on Information Theory*, 44(5):1897–1905, 1998.

[16] Ran Canetti, Juan A. Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas. Multicast security: A taxonomy and some efficient constructions. In *INFOCOM*, pages 708–716, 1999.

[17] David W. Carman, Peter S. Kruus, and Brian J. Matt. Constrains and approaches for distributed sensor network security, 2000. Tech. rep. 00-010. NAI Labs, Glenwood, MD.

[18] Seyit Ahmet Çamtepe and Bülent Yener. Combinatorial design of key distribution mechanisms for wireless sensor networks. In Pierangela Samarati, Peter Y. A. Ryan, Dieter Gollmann, and Refik Molva, editors, *ESORICS*, volume 3193 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2004.

[19] Seyit Ahmet Çamtepe and Bülent Yener. Key distribution mechanisms for wiless sensor networks: a survey, 2005. Technical Report TR-05-07 Rensselaer Polytechnic Institute, Computer Science Department, March 2005.

[20] Seyit Ahmet Çamtepe and Bülent Yener. Combinatorial design of key distribution mechanisms for wireless sensor networks. *IEEE/ACM Trans. Netw.*, 15(2):346–358, 2007.

[21] Seyit Ahmet Çamtepe, Bülent Yener, and Moti Yung. Expander graph based key distribution mechanisms in wireless sensor networks. In *IEEE International Conference on Communications (ICC) 2006*, pages 2262–2267, 2006.

[22] Dibyendu Chakrabarti. Applications of combinatorial designs in key predistribution in sensor networks. Ph. D. Thesis, Indian Statistical Institute, India, September, 2007.

[23] Dibyendu Chakrabarti, Subhamoy Maitra, and Bimal Roy. A key predistribution scheme for wireless sensor networks: Merging blocks in combinatorial design. In Jianying Zhou, Javier Lopez, Robert H. Deng, and Feng Bao, editors, *ISC*, volume 3650 of *Lecture Notes in Computer Science*, pages 89–103. Springer, 2005.

[24] Dibyendu Chakrabarti, Subhamoy Maitra, and Bimal Roy. A key predistribution scheme for wireless sensor networks: merging blocks in combinatorial design. *Int. J. Inf. Sec.*, 5(2):105–114, 2006.

[25] Dibyendu Chakrabarti and Jennifer Seberry. Combinatorial structures for design of wireless sensor networks. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *ACNS*, volume 3989 of *Lecture Notes in Computer Science*, pages 365–374, 2006.

[26] Aldar C.-F. Chan. Distributed symmetric key management for mobile ad hoc networks. In *INFOCOM*, 2004.

[27] Haowen Chan and Adrian Perrig. PIKE: peer intermediaries for key establishment in sensor networks. In *INFOCOM*, pages 524–535. IEEE, 2005.

[28] Haowen Chan, Adrian Perrig, and Dawn Xiaodong Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, pages 197–213. IEEE Computer Society, 2003.

[29] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, pages 11–19. ACM, 1988.

[30] Jianwei Chen, Li Xu, and Yi Mu. A new group rekeying scheme based on t-packing designs for ad hoc networks. In Jianzhong Li, Wang-Chien Lee, and Fabrizio Silvestri, editors, *Infoscale*, volume 304 of *ACM International Conference Proceeding Series*, pages 54:1–54:7. ACM, 2007.

[31] M. Chen, W. Cui, V. Wen, and A. Woo. Security and deployment issues in a sensor. Ninja Project: A Scalable Internet Services Architecture, Berkeley, 2000.

[32] Yi Cheng and Dharma P. Agrawal. An improved key distribution mechanism for large-scale hierarchical wireless sensor networks. *Ad Hoc Networks*, 5(1):35–48, 2007.

[33] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In Yvo Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 257–270. Springer, 1994.

[34] Benny Chor, Amos Fiat, Moni Naor, and Benny Pinkas. Tracing traitors. *IEEE Transactions on Information Theory*, 46(3):893–910, 2000.

[35] Parichoy Pal Choudhury, Pramita Bagchi, Sebanti Sengupta, and Anurag Ghosh. On effect of compromised nodes on security of wireless sensor network. Manuscript.

[36] Willard H. Clatworthy. *Tables of Two-Associate-Class Partially Balanced Designs*. NBS Applied Mathematics Series, vol. 63, 1973.

[37] Charles J. Colbourn and Jeffrey H. Dinitz. *The CRC Handbook of Combinatorial Designs*. CRC Press, 1995.

[38] Ingemar J. Cox, Joe Kilian, Frank Thomson Leighton, and Talal Shamoon. Secure spread spectrum watermarking for multimedia. *IEEE Transactions on Image Processing*, 6(12):1673–1687, 1997.

[39] Abhijit Das and Bimal Roy. A new key-predistribution scheme for highly mobile sensor networks. In Shrisha Rao, Mainak Chatterjee, Prasad Jayanti, C. Siva Ram Murthy, and Sanjoy Kumar Saha, editors, *ICDCN*, volume 4904 of *Lecture Notes in Computer Science*, pages 298–303. Springer, 2008.

[40] Ashok Kumar Das and Indranil Sengupta. An effective group-based key establishment scheme for large-scale wireless sensor networks using bivariate polynomials. In *COMSWARE*, pages 9–16. IEEE, 2008.

[41] Jing Deng, Richard Han, and Shivakant Mishra. Enhancing base station security in wireless sensor networks. Tech. Rep. CU-CS-951-03, Department of Computer Science, University of Colorado. April 2003.

[42] Jing Deng, Richard Han, and Shivakant Mishra. A performance evaluation of intrusion-tolerant routing in wireless sensor networks. In Feng Zhao and Leonidas J. Guibas, editors, *IPSN*, volume 2634 of *Lecture Notes in Computer Science*, pages 349–364. Springer, 2003.

[43] T. Dierks and C. Allen. The TLS Protocol Version 1.0, 1999. RFC 2246, January 1999.

[44] Junwu Dong, Dingyi Pei, and Xueli Wang. A key predistribution scheme based on 3-designs. In Dingyi Pei, Moti Yung, Dongdai Lin, and Chuankun Wu, editors, *Inscrypt*, volume 4990 of *Lecture Notes in Computer Science*, pages 81–92. Springer, 2007.

[45] Junwu Dong, Dingyi Pei, and Xueli Wang. A class of key predistribution schemes based on orthogonal arrays. *Journal of Computer Science and Technology*, 23(5):825–831, 2008.

[46] John R. Douceur. The Sybil Attack. In Peter Druschel, M. Frans Kaashoek, and Antony I. T. Rowstron, editors, *IPTPS*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer, 2002.

[47] Wenliang Du, Jing Deng, Yunghsiang S. Han, Shigang Chen, and Pramod K. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *INFOCOM*, pages 261–268, 2004.

[48] Wenliang Du, Jing Deng, Yunghsiang S. Han, and Pramod K. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM Conference on Computer and Communications Security*, pages 42–51. ACM, 2003.

[49] Wenliang Du, Jing Deng, Yunghsiang S. Han, and Pramod K. Varshney. A key predistribution scheme for sensor networks using deployment knowledge. *IEEE Trans. Dependable Sec. Comput.*, 3(1):62–77, 2006.

[50] Wenliang Du, Jing Deng, Yunghsiang S. Han, Pramod K. Varshney, Jonathan Katz, and Aram Khalili. A pairwise key predistribution scheme for wireless sensor networks. *ACM Trans. Inf. Syst. Secur.*, 8(2):228–258, 2005.

[51] Xiaojiang Du, Mohsen Guizani, Yang Xiao, Song Ci, and Hsiao-Hwa Chen. A routing-driven key management scheme for heterogeneous sensor networks. IEEE transaction on Wireless Communications. To appear.

[52] Xiaojiang Du, Yang Xiao, Mohsen Guizani, and Hsiao-Hwa Chen. An effective key management scheme for heterogeneous sensor networks. *Ad Hoc Networks*, 5(1):24–34, 2007.

[53] Mohamed Eltoweissy, Mohammad Hossain Heydari, Linda Morales, and Ivan Hal Sudborough. Combinatorial optimization of group key management. *J. Network Syst. Manage.*, 12(1):33–50, 2004.

[54] Paul Erdös and Alfred Rényi. On random graphs. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

[55] Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In Vijayalakshmi Atluri, editor, *ACM Conference on Computer and Communications Security*, pages 41–47. ACM, 2002.

[56] Adrian Carlos Ferreira, Marcos Aurélio Vilaça, Leonardo B. Oliveira, Eduardo Habib, Hao Chi Wong, and Antonio Alfredo Ferreira Loureiro. On the security of cluster-based communication protocols for wireless sensor networks. In Pascal Lorenz and Petre Dini, editors, *ICN (1)*, volume 3420 of *Lecture Notes in Computer Science*, pages 449–458. Springer, 2005.

[57] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer, 1993.

[58] Amos Fiat and Tamir Tassa. Dynamic traitor training. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 354–371. Springer, 1999.

[59] Amos Fiat and Tamir Tassa. Dynamic traitor tracing. *J. Cryptology*, 14(3):211–223, 2001.

[60] Eli Gafni, Jessica Staddon, and Yiqun Lisa Yin. Efficient methods for integrating traceability and broadcast encryption. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 372–387. Springer, 1999.

[61] Gunnar Gaubatz, Jens-Peter Kaps, and Berk Sunar. Public key cryptography in sensor networks - revisited. In Claude Castelluccia, Hannes Hartenstein, Christof Paar, and Dirk Westhoff, editors, *ESAS*, volume 3313 of *Lecture Notes in Computer Science*, pages 2–18. Springer, 2004.

[62] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[63] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229. ACM, 1987.

[64] Gaurav Gupta and Mohamed Younis. Performance evaluation of load-balanced clustering of wireless sensor networks. In *In Proceedings of IEEE International conference on communications (ICC)*, pages 1577–1581, 2003.

[65] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing elliptic curve cryptography and RSA on 8-bit cpus. In Marc Joye and Jean-Jacques Quisquater, editors, *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 119–132. Springer, 2004.

[66] Helena Handschuh and M. Anwar Hasan, editors. *Selected Areas in Cryptography, 11th International Workshop, SAC 2004, Waterloo, Canada, August 9-10, 2004, Revised Selected Papers*, volume 3357 of *Lecture Notes in Computer Science*. Springer, 2004.

[67] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *IEEE Hawaii International Conference on System Sciences*, pages 1–10, 2000.

[68] Henk D. L. Hollmann, Jack H. van Lint, Jean-Paul M. G. Linnartz, and Ludo M. G. M. Tolhuizen. On codes with the identifiable parent property. *J. Comb. Theory, Ser. A*, 82(2):121–133, 1998.

[69] Dijiang Huang and Deep Medhi. Secure pairwise key establishment in large-scale sensor networks: An area partitioning and multigroup key predistribution approach. *TOSN*, 3(3):16:1–16:34, 2007.

[70] Dijiang Huang, Manish Mehta, Deep Medhi, and Lein Harn. Location-aware key management scheme for wireless sensor networks. In *2nd ACM workshop on Security of ad hoc and sensor networks, SASN, Washington, USA*, pages 29–42, 2004.

[71] Sajid Hussain, Firdous Kausar, and Ashraf Masood. An efficient key distribution scheme for heterogeneous sensor networks. In Mohsen Guizani, Hsiao-Hwa Chen, and Xi Zhang, editors, *IWCMC*, pages 388–392. ACM, 2007.

[72] Joengmin Hwang and Yongdae Kim. Revisiting random key pre-distribution schemes for wireless sensor networks. In Sanjeev Setia and Vipin Swarup, editors, *SASN*, pages 43–52. ACM, 2004.

[73] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *MOBICOM*, pages 56–67, 2000.

[74] Gaurav Jolly, Mustafa C. Kusçu, Pallavi Kokate, and Mohamed F. Younis. A low-energy key management protocol for wireless sensor networks. In *ISCC*, pages 335–340. IEEE Computer Society, 2003.

[75] R. Kalidindi, R. Kannan, S.S. Iyengar, and A. Durresi. Sub-grid based key vector assignment: A key pre-distribution scheme for distributed sensor networks. *Journal of Pervasive Computing and Communications*, 2(1):35–43, 2006.

[76] Aman Kansal, Arun A. Somasundara, David Jea, Mani B. Srivastava, and Deborah Estrin. Intelligent fluid infrastructure for embedded networking. In *MobiSys*, pages 111–124. USENIX, 2004.

[77] C. Karlof, Y. Li, and J. Polastre. Arrive: An architecture for robust routing in volatile environments. Technical report UCB/CSD-03-1233, University of California, 2003.

[78] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. *Ad Hoc Networks*, 1(2-3):293–315, 2003.

[79] Brad Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *MOBICOM*, pages 243–254, 2000.

[80] Aggelos Kiayias and Moti Yung. On crafty pirates and foxy tracers. In Tomas Sander, editor, *Digital Rights Management Workshop*, volume 2320 of *Lecture Notes in Computer Science*, pages 22–39. Springer, 2001.

[81] Aggelos Kiayias and Moti Yung. Breaking and repairing asymmetric public-key traitor tracing. In Joan Feigenbaum, editor, *Digital Rights Management Workshop*, volume 2696 of *Lecture Notes in Computer Science*, pages 32–50. Springer, 2002.

[82] Hirotaka Komaki, Yuji Watanabe, Goichiro Hanaoka, and Hideki Imai. Efficient asymmetric self-enforcement scheme with public traceability. In Kwangjo Kim, editor, *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 225–239. Springer, 2001.

[83] Kaoru Kurosawa and Yvo Desmedt. Optimum traitor tracing and asymmetric schemes. In *EUROCRYPT*, pages 145–157, 1998.

[84] Jooyoung Lee. Combinatorial approaches to key predistribution for Distributed Sensor Networks, 2005. PhD Thesis, University of Waterloo, Canada, 2005.

[85] Jooyoung Lee and Douglas R. Stinson. Deterministic key predistribution schemes for distributed sensor networks. In Handschuh and Hasan [66], pages 294–307.

[86] Jooyoung Lee and Douglas R. Stinson. A combinatorial approach to key predistribution for distributed sensor networks. In *IEEE Wireless Communications and Networking Conference, WCNC 2005, New Orleans, LA, USA*, pages 1200–1205, 2005.

[87] Jooyoung Lee and Douglas R. Stinson. Tree-based key distribution patterns. In Bart Preneel and Stafford E. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 189–204. Springer, 2005.

[88] Jooyoung Lee and Douglas R. Stinson. Common intersection designs. *Journal of Combinatorial Designs*, 14:251–269, 2006.

[89] Jooyoung Lee and Douglas R. Stinson. On the construction of practical key predistribution schemes for distributed sensor networks using combinatorial designs. *ACM Trans. Inf. Syst. Secur.*, 11(2), 2008.

[90] An Liu and Peng Ning. TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. In *IPSN*, pages 245–256. IEEE Computer Society, 2008.

[91] Donggang Liu and Peng Ning. Establishing pairwise keys in distributed sensor networks. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM Conference on Computer and Communications Security*, pages 52–61. ACM, 2003.

[92] Donggang Liu and Peng Ning. Location-based pairwise key establishments for static sensor networks. In Sanjeev Setia and Vipin Swarup, editors, *SASN*, pages 72–82. ACM, 2003.

[93] Donggang Liu and Peng Ning. Improving key predistribution with deployment knowledge in static sensor networks. *TOSN*, 1(2):204–239, 2005.

[94] Donggang Liu, Peng Ning, and Wenliang Du. Group-based key predistribution in wireless sensor networks. In Markus Jakobsson and Radha Poovendran, editors, *Workshop on Wireless Security*, pages 11–20. ACM, 2005.

[95] Donggang Liu, Peng Ning, and Wenliang Du. Group-based key predistribution for wireless sensor networks. *TOSN*, 4(2):11:1–11:30, 2008.

[96] Donggang Liu, Peng Ning, and Rongfang Li. Establishing pairwise keys in distributed sensor networks. *ACM Trans. Inf. Syst. Secur.*, 8(1):41–77, 2005.

[97] Boushra Maala, Yacine Challal, and Abdelmadjid Bouabdallah. HERO: Hierarchical key management protocol for heterogeneous wireless sensor networks. *Wireless Sensor and Actor Networks II*, 264:125–136, 2008.

[98] Florence Jessie MacWilliams and Neil J. A. Sloane. *The theory of error correcting codes*. North-Holland, 1988.

[99] Samuel Madden, Robert Szewczyk, Michael J. Franklin, and David E. Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *WMCSA*, pages 49–58. IEEE Computer Society, 2002.

[100] David Malan, Matt Welsh, and Michael Smith. A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. In *First IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON)*, pages 119–132, 2004.

[101] Keith M. Martin. The combinatorics of cryptographic key establishment. *Surveys in Combinatorics, London Mathematical Society Lecture Note Series, Cambridge University Press*, 346:223–273, 2007.

[102] Keith M. Martin and Maura B. Paterson. An application-oriented framework for wireless sensor network key establishment. In *Third Workshop on Cryptography for Ad-hoc Networks WCAN'07*, 2007.

[103] Keith M. Martin and Maura B. Paterson. An application-oriented framework for wireless sensor network key establishment. *Electronic Notes in Theoretical Computer Science*, 192(2):31–41, 2008.

[104] Keith M. Martin, Maura B. Paterson, and Douglas R. Stinson. Key predistribution for homogeneous wireless sensor networks with group deployment of nodes, 2008. Available at ePrint Cryptology archive 2008/412.

[105] Simon McNicol. Traitor tracing using generalized Reed-Solomon codes and combinatorial designs, 2005. PhD Thesis, RMIT University, July 2005.

[106] Simon McNicol, Serdar Boztas, and Asha Rao. Traitor tracing against powerful attacks. In *IEEE International Symposium on Information Theory*, pages 1878–1882, 2005.

[107] Simon McNicol, Serdar Boztas, and Asha Rao. Traitor tracing against powerful attacks using combinatorial designs. In Marc P. C. Fossorier, Hideki Imai, Shu Lin, and Alain Poli, editors, *AAECC*, volume 3857 of *Lecture Notes in Computer Science*, pages 215–224. Springer, 2006.

[108] Ralph C. Merkle. Secure communications over insecure channels. *Commun. ACM*, 21(4):294–299, 1978.

[109] Chris J. Mitchell and Fred Piper. Key storage in secure networks. *Discrete Applied Mathematics*, 21:215–228, 1988.

[110] Abedelaziz Mohaisen, YoungJae Maeng, and DaeHun Nyang. On grid-based key pre-distribution: Toward a better connectivity in wireless sensor network. In Takashi Washio, Zhi-Hua Zhou, Joshua Zhexue Huang, Xiaohua Hu, Jinyan Li, Chao Xie, Jieyue He, Deqing Zou, Kuan-Ching Li, and Mário M. Freire, editors, *PAKDD Workshops*, volume 4819 of *Lecture Notes in Computer Science*, pages 527–537. Springer, 2007.

[111] Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer, 2001.

[112] Moni Naor and Benny Pinkas. Threshold traitor tracing. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 502–517. Springer, 1998.

[113] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *STOC*, pages 245–254, 1999.

[114] Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. In Yair Frankel, editor, *Financial Cryptography*, volume 1962 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2000.

[115] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.

[116] Leonardo B. Oliveira, Adrian Carlos Ferreira, Marcos Aurélio Vilaça, Hao Chi Wong, Marshall W. Bern, Ricardo Dahab, and Antonio Alfredo Ferreira Loureiro. SecLEACH - On the security of clustered sensor networks. *Signal Processing*, 87(12):2882–2895, 2007.

[117] Leonardo B. Oliveira, Hao Chi Wong, Marshall W. Bern, Ricardo Dahab, and Antonio Alfredo Ferreira Loureiro. SecLEACH - A random key distribution solution for securing clustered sensor networks. In *NCA*, pages 145–154. IEEE Computer Society, 2006.

[118] Maura B. Paterson and Douglas B. Stinson. Two attacks on a sensor network key distribution scheme of Cheng and Agrawal. *Journal of Mathematical Cryptology*, 2:393–403, 2008.

[119] S. E. Payne and J. A. Thas. *Finite generalized quadrangles*. Pitman, Boston, 1984.

[120] Adrian Perrig, Ran Canetti, J. D. Tygar, and Dawn Xiaodong Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, pages 56–73, 2000.

[121] Adrian Perrig, Robert Szewczyk, Victor Wen, David E. Culler, and J. D. Tygar. SPINS: security protocols for sensor netowrks. In *MOBICOM*, pages 189–199, 2001.

[122] Birgit Pfitzmann. Trials of traced traitors. In Ross J. Anderson, editor, *Information Hiding*, volume 1174 of *Lecture Notes in Computer Science*, pages 49–64. Springer, 1996.

[123] Birgit Pfitzmann and Matthias Schunter. Asymmetric fingerprinting (extended abstract). In *EUROCRYPT*, pages 84–95, 1996.

[124] Birgit Pfitzmann and Michael Waidner. Asymmetric fingerprinting for larger collusions. In *ACM Conference on Computer and Communications Security*, pages 151–160, 1997.

[125] Roberto Di Pietro, Luigi V. Mancini, and Sushil Jajodia. Providing secrecy in key management protocols for large wireless sensors networks. *Ad Hoc Networks*, 1(4):455–468, 2003.

[126] Roberto Di Pietro, Luigi V. Mancini, and Alessandro Mei. Energy efficient node-to-node authentication and communication confidentiality in wireless sensor networks. *Wireless Networks*, 12(6):709–721, 2006.

[127] Hossein Pishro-Nik. Analysis of finite unreliable sensor grids. In *WiOpt*, pages 326–335. IEEE, 2006.

[128] Sylvia Ratnasamy, Brad Karp, Li Yin, Fang Yu, Deborah Estrin, Ramesh Govindan, and Scott Shenker. GHT: a geographic hash table for data-centric storage. In Cauligi S. Raghavendra and Krishna M. Sivalingam, editors, *WSNA*, pages 78–87. ACM, 2002.

[129] Rodrigo Roman, Jianying Zhou, and Javier Lopez. On the security of wireless sensor networks. In Osvaldo Gervasi, Marina L. Gavrilova, Vipin Kumar, Antonio Laganà, Heow Pueh Lee, Youngsong Mun, David Taniar, and Chih Jeng Kenneth Tan, editors, *ICCSA (3)*, volume 3482 of *Lecture Notes in Computer Science*, pages 681–690. Springer, 2005.

[130] Kay Römer and Friedemann Mattern. The design space of wireless sensor networks. *IEEE Wireless Communications*, 11(6):54–61, 2004.

[131] Sushmita Ruj, Subhamoy Maitra, and Bimal Roy. Key predistribution using transversal design on a grid of wireless sensor network. *Ad Hoc & Sensor Wireless Networks*, 5(3-4):247–264, 2008.

[132] Sushmita Ruj and Bimal Roy. Key predistribution using partially balanced designs in wireless sensor networks. International Journal of High Performance Computing and Networking (IJHPCN), Accepted.

[133] Sushmita Ruj and Bimal Roy. Revisiting key predistribution using transversal designs for a grid-based deployment scheme. International Journal of Distributed Sensor Networks, Accepted.

[134] Sushmita Ruj and Bimal Roy. Key predistribution using partially balanced designs in wireless sensor networks. In Ivan Stojmenovic, Ruppa K. Thulasiram, Laurence Tianruo Yang, Weijia Jia, Minyi Guo, and Rodrigo Fernandes de Mello, editors, *ISPA*, volume 4742 of *Lecture Notes in Computer Science*, pages 431–445. Springer, 2007.

[135] Sushmita Ruj and Bimal Roy. Key distribution schemes using combinatorial designs to identify all traitors. *Congressus Numerantium*, 193:195–214, 2008.

[136] Sushmita Ruj and Bimal Roy. Key establishment algorithms for some deterministic key predistribution schemes. In Alfonso Rodríguez, Mariemma Inmaculada Yagüe del Valle, and Eduardo Fernández-Medina, editors, *WOSIS*, pages 68–77. INSTICC Press, 2008.

[137] Sushmita Ruj and Bimal Roy. Key predistribution schemes using codes in wireless sensor networks. In *Proceedings of The 4th International Conference on Information Security and Cryptology, Inscrypt*, pages 297–311, 2008.

[138] Sushmita Ruj and Bimal Roy. Key predistribution using combinatorial designs for grid-group deployment scheme in wireless sensor networks. *ACM Transactions on Sensor Networks*, 6(1), 2010 (in Press).

[139] Mohammed Golam Sadi, Dong Seong Kim, and Jong Sou Park. GBR: Grid based random key predistribution for wireless sensor network. In *ICPADS (2)*, pages 310–315. IEEE Computer Society, 2005.

[140] Reihaneh Safavi-Naini and Yejing Wang. A combinatorial approach to asymmetric traitor tracing. In Ding-Zhu Du, Peter Eades, Vladimir Estivill-Castro, Xuemin Lin, and Arun Sharma, editors, *COCOON*, volume 1858 of *Lecture Notes in Computer Science*, pages 416–425. Springer, 2000.

[141] Reihaneh Safavi-Naini and Yejing Wang. Sequential traitor tracing. In Mihir Bellare, editor, *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 316–332. Springer, 2000.

[142] Reihaneh Safavi-Naini and Yejing Wang. New results on frame-proof codes and traceability schemes. *IEEE Transactions on Information Theory*, 47(7):3029–3033, 2001.

[143] Reihaneh Safavi-Naini and Yejing Wang. Sequential traitor tracing. *IEEE Transactions on Information Theory*, 49(5):1319–1326, 2003.

[144] Sanjay Shakkottai, R. Srikant, and Ness B. Shroff. Unreliable sensor grids: Coverage, connectivity and diameter. In *INFOCOM*, 2003.

[145] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[146] Ryuichi Sakai Shigeo Mitsunari and Masao Kasahara and. A new traitor tracing. *IEICE Transactions*, E85-A(2):481–484, 2002.

[147] Alice Silverberg, Jessica Staddon, and Judy L. Walker. Efficient traitor tracing algorithms using list decoding. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 175–192. Springer, 2001.

[148] Alice Silverberg, Jessica Staddon, and Judy L. Walker. Applications of list decoding to tracing traitors. *IEEE Transactions on Information Theory*, 49(5):1312–1318, 2003.

[149] Katerina Simonova, Alan C. H. Ling, and Xiaoyang Sean Wang. Location-aware key predistribution scheme for wide area wireless sensor networks. In Sencun Zhu and Donggang Liu, editors, *SASN*, pages 157–168. ACM, 2006.

[150] Jessica Staddon, Dirk Balfanz, and Glenn Durfee. Efficient tracing of failed nodes in sensor networks. In Cauligi S. Raghavendra and Krishna M. Sivalingam, editors, *WSNA*, pages 122–130. ACM, 2002.

[151] Jessica Staddon, Douglas R. Stinson, and Ruizhong Wei. Combinatorial properties of frameproof and traceability codes. *IEEE Transactions on Information Theory*, 47(3):1042–1049, 2001.

[152] Jennifer G. Steiner, B. Clifford Neuman, and Jeffrey I. Schiller. Kerberos: An authentication service for open network systems. In *USENIX Winter*, pages 191–202, 1988.

[153] Douglas R. Stinson. On some methods for unconditionally secure key distribution and broadcast encryption. *Des. Codes Cryptography*, 12(3):215–243, 1997.

[154] Douglas R. Stinson. *Combinatorial Designs: Constructions and Analysis*. Springer, 2004.

[155] Douglas R. Stinson. *Cryptography: Theory and Practice, Third Edition*. CRC Press Inc., Boca Raton, 2006.

[156] Douglas R. Stinson and Tran van Trung. Some new results on key distribution patterns and broadcast encryption. *Des. Codes Cryptography*, 14(3):261–279, 1998.

[157] Douglas R. Stinson, Tran van Trung, and Ruizhong Wei. Secure frameproof codes, key distribution patterns, group testing algorithms and related structures. *Journal of Statistical Planning and Inference*, 86:595–617, 2000.

[158] Douglas R. Stinson and Ruizhong Wei. Combinatorial properties and constructions of traceability schemes and frameproof codes. *SIAM J. Discrete Math.*, 11(1):41–53, 1998.

[159] Douglas R. Stinson and Ruizhong Wei. Key preassigned traceability schemes for broadcast encryption. In Stafford E. Tavares and Henk Meijer, editors, *Selected Areas in Cryptography*, volume 1556 of *Lecture Notes in Computer Science*, pages 144–156. Springer, 1998.

[160] Douglas R. Stinson and Ruizhong Wei. An application of ramp schemes to broadcast encryption. *Inf. Process. Lett.*, 69(3):131–135, 1999.

[161] Ivan Stojmenovic. *Handbook of sensor networks: Algorithms and Architecture*. Wiley Interscience, 2005.

[162] Anne Penfold Street and Deborah J. Street. *Combinatorics of Experimental Design*. Clarendon Press, Oxford, 1987.

[163] Madhu Sudan. Algorithmic introduction to coding theory, Lecture 4, 2001. http://people.csail.mit.edu/madhu/FT01/scribe/lect4.ps. Last accessed on January 22, 2009.

[164] Yuldi Tirta, Zhiyuan Li, Yung-Hsiang Lu, and Saurabh Bagchi. Efficient collection of sensor data in remote fields using mobile collectors. In Ronald P. Luijten, Luiz A. DaSilva, and Antonius P. J. Engbersen, editors, *ICCCN*, pages 515–520. IEEE, 2004.

[165] Vu Dong Tô, Reihaneh Safavi-Naini, and Fangguo Zhang. New traitor tracing schemes using bilinear map. In Moti Yung, editor, *Digital Rights Management Workshop*, pages 67–76. ACM, 2003.

[166] Manohar Narhar Vartak. On an application of kronecker product of matrices to statistical designs. *Annals of Mathematical Statistics*, 26(3):420–438, 1955.

[167] Louis Le Porquier De Vaux and Nicolas Dumain. Key pre-distribution in sensor networks. Training course report, Military Academy of Saint-Cyr, Department of Engineering Science, 2008.

[168] Guiling Wang, Guohong Cao, and Thomas F. La Porta. Movement-assisted sensor deployment. In *INFOCOM*, pages 640–652, 2004.

[169] Yejing Wang, Jennifer Seberry, Beata J. Wysocki, Tadeusz A. Wysocki, Le Chung Tran, Ying Zhao, and Tianbing Xia. An exposure property of block designs. *Australasian Journal of Combinatorics*, 33:147–156, 2005.

[170] Yuji Watanabe, Goichiro Hanaoka, and Hideki Imai. Efficient asymmetric public-key traitor tracing without trusted agents. In David Naccache, editor, *CT-RSA*, volume 2020 of *Lecture Notes in Computer Science*, pages 392–407. Springer, 2001.

[171] Ruizhong Wei and Jiang Wu. Product construction of key distribution schemes for sensor networks. In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography*, volume 3357 of *Lecture Notes in Computer Science*, pages 280–293. Springer, 2004.

[172] Yang Xiao, Venkata Krishna Rayi, Bo Sun, Xiaojiang Du, Fei Hu, and Michael Galloway. A survey of key management schemes in wireless sensor networks. *Computer Communications*, 30(11-12):2314–2341, 2007.

[173] Fan Ye, Haiyun Luo, Jerry Cheng, Songwu Lu, and Lixia Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In Ian F. Akyildiz, Jason Yi-Bing Lin, Ravi Jain, Vaduvur Bharghavan, and Andrew T. Campbell, editors, *MOBICOM*, pages 148–159. ACM, 2002.

[174] Mohamed F. Younis, Kajaldeep Ghumman, and Mohamed Eltoweissy. Location-aware combinatorial key management scheme for clustered sensor networks. *IEEE Trans. Parallel Distrib. Syst.*, 17(8):865–882, 2006.

[175] Zhen Yu and Yong Guan. A key pre-distribution scheme using deployment knowledge for wireless sensor networks. In *IPSN*, pages 261–268. IEEE, 2005.

[176] Zhen Yu and Yong Guan. A key management scheme using deployment knowledge for wireless sensor networks. *IEEE Transactions of Parallel and Distributed Systems*, 19(10):1411–1425, 2008.

[177] Wensheng Zhang, Guohong Cao, and Thomas F. La Porta. Data dissemination with ring-based index for wireless sensor networks. In *ICNP*, pages 305–314. IEEE Computer Society, 2003.

[178] Wensheng Zhang, Hui Song, Sencun Zhu, and Guohong Cao. Least privilege and privilege deprivation: towards tolerating mobile sink compromises in wireless sensor networks. In P. R. Kumar, Andrew T. Campbell, and Roger Wattenhofer, editors, *MobiHoc*, pages 378–389. ACM, 2005.

[179] Li Zhou, Jinfeng Ni, and Chinya V. Ravishankar. Supporting secure communication and data collection in mobile sensor networks. In *INFOCOM*. IEEE, 2006.

[180] Lidong Zhou and Z.J Haas. Securing ad hoc networks. *IEEE Networks*, 13(6):24–30, 1999.

[181] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. LEAP: efficient security mechanisms for large-scale distributed sensor networks. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM Conference on Computer and Communications Security*, pages 62–72. ACM, 2003.

[182] Sencun Zhu, Shouhuai Xu, Sanjeev Setia, and Sushil Jajodia. Establishing pairwise keys for secure communication in ad hoc networks: A probabilistic approach. In *ICNP*, pages 326–335. IEEE Computer Society, 2003.

[183] Yi Zou and Krishnendu Chakrabarty. Sensor deployment and target localization based on virtual forces. In *INFOCOM*, pages 1293–1303, 2003.