Deterministic Merging of Blocks in Combinatorial Design based Key Predistribution in Distributed Wireless Sensor Network

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Technology
in
Computer Science & Engineering

by

Anupam Pattanayak

Department of Computer Science & Engineering
National Institute of Technology Rourkela
Orissa, India – 769 008

May 2009

Deterministic Merging of Blocks in Combinatorial Design based Key Predistribution in Distributed Wireless Sensor Network

M. Tech Thesis submitted to National Institute of Technology Rourkela in partial fulfillment of the requirement for the award of the degree of Master of Technology in Computer Science & Engineering with specialization in Information Security

by

Anupam Pattanayak

Computer Science & Engineering Department

National Institute of Technology Rourkela

e-mail: anupamp-cs210@nitrkl.ac.in

Under the supervision of

Prof. Dr. Banshidhar Majhi

Computer Science & Engineering Department

National Institute of Technology Rourkela

Rourkela, Orissa, India – 769008

e-mail: bmajhi@nitrkl.ac.in

Dedicated to my Parents

Whose efforts, sacrifice, patience, inspiration, and encouragement

are helping me to move forward

National Institute of Technology Rourkela
Rourkela, Orissa, India - 769008

CERTIFICATE

This is to certify that the M. Tech thesis report, entitled Deterministic Merging of Blocks in Combinatorial Design based Key Predistribution in Distributed Wireless Sensor Network by Anupam Pattanayak, has been submitted to National Institute of Technology Rourkela for partial fulfillment of the requirements for the award of the degree of Master of Technology in Computer Science & Engineering with specialization in Information Security during the session 2008-2009. This is the original and authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, this thesis report or the matter embodied in the thesis has not been submitted to any other University/Institute for the award of any Degree.

Professor Banshidhar Majhi,

Professor and Head,
Computer Science & Engineering Department,
National Institute of Technology Rourkela
India – 769 008

Date: 26.05.2009
Place: NIT Rourkela

ACKNOWLEDGEMENTS

It was my wish that I would do research in Cryptography. Coincidentally, I got admission in M. Tech in Computer Science & Engineering with specialization in Information Security in NIT Rourkela. In some way this thesis work is some short of *wish comes true*.

I express my gratitude and sincere thanks to my supervisor Professor Dr. Banshidhar Majhi. He provided all kind of supports needed to carry out this thesis, right from sending us to ISI Kolkata to get exposure of work being done in ISI in Information Security area. He has always been very kind to me and has kept his faith on my ability.

I thank Professor Bimal Roy, ISI Kolkata, who was our supervisor during summer training there in 2008. We did summer training on key predistribution in sensor network. I have continued my thesis work in the same area. Also I thank Sushmita Ruj, senior research fellow, CRG group, ISI Kolkata. I communicated many times to her for help regarding thesis work and she always extended her helping hand.

I express my thanks to all the faculty and staff members of our Computer Science & Engineering Department, NIT Rourkela where we got a very good research environment and it is due to everybody's conscious effort towards excellence.

I express my thanks to all my friends in NIT Rourkela who has directly or indirectly helped me and stood beside me always. The list of names is large, but I would like to mention few of them such as Shine, Shyju, Deepak, Srichandan, Om Prakash, Parmanand, Vikas, Shriram, Dibyendu, Subhasish, Aloke, Ramana, Niranjan, Thankre, Gaurav and my other friends.

It will remain incomplete if I miss to mention the great support provided by my mother, father, brother, and all my family members to me during the thesis work.

Anupam Pattanayak

ABSTRACT

Sensor nodes have severe constraints in terms of its resources such as processing power, memory, communication range, battery power. Due to wireless nature of communication between nodes in a wireless sensor network, any attacker can intercept the communicating messages between sensor nodes. So the need for securing these messages is obvious. Due to resource constraints of sensor nodes, public key cryptography can't be employed for securing the communication as public key cryptography demands much computational effort. So, private key cryptography is natural choice for securing the communication in wireless sensor network.

Key predistribution has become obvious choice for distributing keys in sensor nodes for secured communication in a wireless sensor network. A pool of keys is first taken, and then a set of keys from this key pool is installed in every sensor node before their deployment. The keys predistributed to a particular sensor node can be drawn from the key pool probabilistically or deterministically. Combinatorial design which was originated as a branch of statistics and later found its vast application in coding theory and of late in cryptography plays a vital role in deterministic key predistribution.

The connectivity and resiliency offered by some combinatorial design based key predistribution schemes can be sometimes offered by merging of blocks and then assign these merged blocks to sensor nodes. The question is how to choose blocks for merging? There is a prior general work on merging of blocks which has been studied on transversal design based key predistribution scheme. This approach is not deterministic, but heuristic. A deterministic algorithm for merging of blocks has been proposed. The orthogonal array based key predistribution scheme has been studied in detail and the non suitability of merging approach to improve its performance has been shown.

In addition, a key establishment algorithm for transversal design based key predistribution scheme has been proposed.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1. INTRODUCTION

A sensor node possesses small amount of resources in terms of processor, memory, battery power, and communication range. But when a large number of sensor nodes work together they are able to accomplish a good volume of task. Wireless Sensor Network represents a new frontier in technology that holds the promise of unprecedented levels of autonomy in the execution of complex systems by utilizing the capability of huge number of inexpensive sensor nodes. As a consequence we are able to interact with physical world directly. Wireless sensor networks would constitute an important part of the next evolution in automation. Also, the wireless sensor networks extend the existing Internet deep into the physical environment. The resulting new network is orders of magnitude of more expansive and dynamic than current TCP/IP network and is creating entirely new types of traffic that are quite different from what one finds on Internet now [24] and [34].

Wireless sensor network is an ad hoc network. Another example of ad hoc network is *mobile ad hoc network* (MANET). Wireless sensor network consists of several tiny sensor nodes. Each node has limited resources. When these several tiny sensor nodes form a wireless network and collaborate locally with each other in the neighborhood to perform some designated task, the task performed by the whole network is quite worthy. Depending on the type of nodes or architecture of the sensor network there are different kind of wireless sensor networks.

1.1 WIRELESS SENSOR NETWORK

A sensor is a tiny device. As its name signifies, it is capable of sensing some real world phenomenon such as temperature, light, movement of any object etc. In general, a sensor node consists of four basic units: (i) Sensing unit, (ii) Processing unit, (iii) Transceiver unit, and (iv) Power unit [1]. Each sensor node (or node, in short) operates unattended and has a microprocessor and a small amount of memory for signal processing and task scheduling. Each node is equipped with one or more sensing components such as acoustic microphone arrays, video or still cameras, infra-red, seismic, or magnetic sensors etc. Each node communicates wirelessly with a few other local nodes in its surrounding area that comes within its radio frequency (RF) communication range [35].

Development of wireless sensor network gained momentum during the last five years of last Millennium. The first example of sensor node is the *Berkley mote*, which was developed at University of California Berkley. A mote tightly integrates 8-bit microcontroller with a low-power radio and various sensors [15]. The famous Smart Dust project in University of California Berkley conceived the mote concept. Soon an operating system that is suitable for these tiny motes was developed. This OS was named *TinyOS*. Afterwards, there is all round development in every aspect of sensor nodes. The spectacular advancements in several hardware technologies made it possible. First, *System-on-Chip (SOC)* is capable of integrating whole systems on a single chip. Second, *Micro-Electro-Mechanical Systems (MEMS)* is now available to integrate a rich set of sensors onto the same CMOS chip. Third, commercial RF circuits enable short distance wireless communication with extremely low power consumption. These technologies coupled with advanced packaging techniques, have made it feasible to integrate sensing, computing, communication, and power components into a miniaturized sensor node [33].

A sensor network generally consists of several tiny sensor nodes and possibly a few powerful control nodes (also called *base stations*). Sensor nodes are usually densely deployed in a large scale and communicate with each other in short distances through wireless communication. Although individual sensor nodes have limited resources, they are capable of achieving worthy task of big volume when they work as a group. Information collected by and transmitted on a sensor network describes conditions of physical environments of the area where the sensor network is deployed. Sensor networks may interact with an IP network via a number of gateways. A gateway routes the user queries or commands to appropriate nodes in a sensor network. It also routes sensor data, sometimes aggregated and summarized, to users who have requested it or expected to use the information. To optimize performances and resources such as energy, we may need to redesign TCP/IP stack so that our needs and constraints are satisfied [35].

Although different application demands a different set of tasks to be carried out, three basic types of tasks carry out those tasks in a sensor network: sensing, processing, and communication. The *sensing task* uses different types of sensors to capture different signals from the physical world. All signals delay as they travel away from the source. A dense deployment of sensors helps to avoid this delay and maintain the sensed signal. Individual sensor nodes are capable of lightweight processing. In sensor networks, processing often combines multiple sensor outputs from local neighboring nodes. This is referred as *collaborative signal and data processing*. Collaborative

processing has the advantages that processing is more accurate and reliable, and only the aggregate result needs to be sent to a user across the network. *The sensor nodes acts as a front end in a computing hierarchy and perform preprocessing for later stages* [24]. WSN management must be *automatic*, i.e., *self-managed* with minimum interface with human, and robust to changes in network states while maintaining quality of service (QOS) [15].

One fundamental challenge in sensor networks is its *dynamics* [25]. Over the time some nodes will fail functioning properly for different reasons such as running out of energy, overheat in the sun, carried away by wind, crash due to software bug etc. These changes are difficult to predict in advance. So, sensor networks must be *self-configuring*, working without fine-grained control from users. They must also be *adaptive* to changes in environment – not to use a single configuration choice, but continually change the configuration in response to dynamics.

A key attribute of sensor networks is to be able to *self-form* [24]. That is, when randomly deployed, the nodes should be able to organize into an efficient network capable of gathering data in a useful and efficient manner. Often, gathering data in a useful manner demands that the exact location of a sensor to be known. This requires that sensors be able to determine their location. This location information is often reused for other purposes.

Mobility: Sensor nodes can be fixed or mobile. Currently most sensors are static and most existing work focuses on networks of static nodes, but in near future mobility feature would be included into the sensor networks.

Functional Layers of WSN: The sensor network is more application specific than traditional networks designed to accommodate various applications. The organization and architecture of a sensor network should be designed or adapted to suit a special task so as to optimizing the system performance, maximize the operation lifetime, and minimize the cost. According to [15] the different functional layers of distributed wireless sensor network are:

- *Sensing layer* – It performs the work of data acquisition
- *Communication layer* - It perform the task of *data correlation*, *data compression*, *data dissemination*, and *routing*. Function of this layer is to deliver statistical observation results

to the collecting centre (the *sink*). A *security layer* may also be inside the communication layer that deals with security.

- *Data fusion layer* – It processes data received from the communication layer and combines them using various signal processing, data fusion, artificial intelligence, and other decision-making techniques. After the appropriate calculation and analysis, the data fusion layer produces the final detection results of a sensor network.
- *User layer* – It is the uppermost layer. It provides man-machine interface.

Distributed Wireless Sensor Network (DWSN) is a set of geographically widely scattered sensors designed to collect information about the environment in which they are deployed. These sensors are interconnected through wireless communications. The physical measurements from the terminal sensor nodes are preprocessed locally into abstract and/or numerical estimates; then they are transmitted through an wireless interconnection communication network to a processing element, where they are integrated with the information gathered from other parts of the network according to some *data fusion* strategy. A group of neighboring sensors commanded by the same processing elements forms a cluster. The efficient synthesis of information, often from noisy and possibly faulty signals from the nodes, requires the some sort of attention to be given to the problems associated with (1) the architecture and fault tolerance of DWSN, (2) the proper synchronization of sensor signals, and (3) the integration of information to achieve lower computation and communication overhead. Once commissioned, a DWSN must organize itself, adapt to any environmental changes, node failure etc, and continue to provide reliable service.

1.2 APPLICATIONS OF WIRELESS SENSOR NETWORK

Application of wireless sensor network are in the field of Sensor networks are used in a number of different areas such as military, industry, health, environment, and home. In industry the sensor nodes can be placed in highly heated area where intervention of human is difficult to control the temperature. In health, sensor nodes can be placed inside the human body to catch the signals whenever blood pressure or heart pumping rates or blood sugar level deviates from the normal behavior. Locations of sensor nodes in the network are not pre determined in most of the cases.

This permits us to deploy sensor nodes in hostile environments such as border area of a hostile neighbor country by some means, for example, by using aeroplane.

1.3 WIRELESS SENSOR NETWORK CLASSIFICATIONS

Two main architectures for data communication in a sensor network are

1) Hierarchical network architecture
2) Flat network architecture

Hierarchical network architecture is energy efficient for collecting and aggregating data within a large target region, where each node in the region is a source node. Hence, hierarchical network protocols are used when data will be collected from the entire sensor network. The flat network architecture is more suitable for transferring data between a source destination pair separated by a large number of hops.

Hierarchical Network Architecture: One way of minimizing the data transmissions over long distances is to cluster the network so that signaling and controlling overhead can be reduced, while critical functions such as routing can be improved. One node in each cluster is designated as the cluster head (CH) and traffic between nodes of different clusters must always be routed through their respective CHs. The number of tiers within the network can vary according to the number of nodes. In hierarchical networks, cluster heads (i.e. relay nodes) are responsible for collecting and aggregating data from sensors.

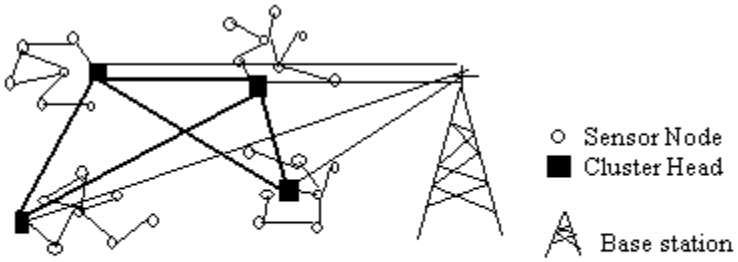

Figure 1.1 Hierarchical Wireless Sensor Network

Flat Network Architecture: Here all nodes are equal and connections are set up between nodes that are in close proximity to establish radio communication. Route discovery can be carried

out in sensor networks using *flooding* that does not require topology maintenance. In flooding, each node receiving data packets broadcasts until all nodes or the nodes or the node at which the packet was originated gets back the packet. But in sensor networks, flooding is minimized or avoided as nodes could receive multiple or duplicate copies of the same data packet due to nodes having common neighbors or sensing similar data. The query is disseminated throughout the network with the querying node acting as a source, and gradually the query is directed the requesting node along multiple paths. In flat ad hoc architecture, sensors collaboratively relay their data to access points (i.e. base stations or sinks).

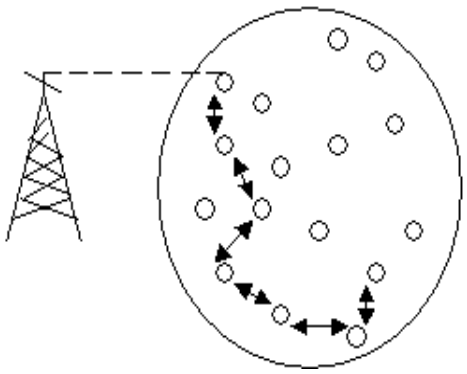

Figure 1.2 Distributed Wireless Sensor Network

1.4 SECURITY ISSUES IN WIRELESS SENSOR NETWORK

WSNs have recently emerged as an important means to observe and interact with the physical world. A sensor network generally consists of numerous tiny sensor nodes and possibly a few powerful control nodes (also called *base stations*). Sensor nodes are usually densely deployed in a large scale and communicate with each other in short distances through wireless communication. The control nodes may further process the data accumulated from the sensor nodes, disseminate control commands to the sensor nodes, and connect the network to a traditional wired network [16]. Sensor nodes are usually scattered randomly in the field and will form a sensor network after deployment in an ad hoc manner to accomplish its designated tasks. Usually no infrastructure support is provided for sensor networks.

After deployment, these sensor nodes are quickly self-organized together to form an ad-hoc network [16]. Each individual sensor node then monitors the conditions and activities in its local surroundings and reports its observations to a central server by communicating with its neighbors. WSN inherits security problems due to

(i) Wireless nature of communication,
(ii) Limitation of capability of individual sensor nodes,
(iii) Large size of the sensor networks,
(iv) Unknown and dynamic network topology, and
(v) Easy chance of physical attack. This results a challenge to design any efficient key management scheme.

In a wireless sensor network one can easily monitor the transmission between nodes as compared to wired networks because of the broadcast nature of communication. Encrypting communication between sensor nodes can partly solve this problem but it requires robust key exchange and distribution scheme. End-to-end encryption is impractical in WSN because of large number of communicating nodes and each node is incapable of storing large number of encryption keys due to memory capacity limitation [29]. So hop-by-hop encryption approach is usually used in which each sensor node stores only encryption keys shared with its immediate neighbors.

Many times, security is viewed as a standalone component of a system's architecture, where a separate module provides security to the whole system. This separation is usually a flawed approach to network security [23]. To secure a system, security must be integrated into every component of a system – hardware, software, communication. Security must pervade every aspect of system design.

Secrecy and authentication: Like traditional networks, most WSN applications require protection against eavesdropping, injection, and modification of packets. Cryptography is the standard defense. A natural system trade-offs between level of security provided and the system overhead in terms computation and memory usage arise when incorporating cryptography into sensor networks. For point-to-point communication, end-to-end cryptography achieves a high level of security but it requires that keys be set up among all end points and be incompatible with passive participation and local broadcast [23].

Future generation sensor network research and user community would respond to more security demands with more advanced use of cryptography. Cryptography entails a performance cost for

extra computation that often increases packet size. Cryptographic hardware support increases efficiency but also increases the financial cost of implementing a network.

Therefore, an important question facing sensor node researchers and practitioners is: Can reasonable security and performance levels be achieved with software-only cryptographic implementations, or is hardware support needed?

Software-only cryptography is quite practical with present sensor technology; hardware support is not required to achieve acceptable security and performance levels. For example, the University of California, Berkeley, implementation of TinySec incurs only an additional 5% –10% performance overhead using software-only methods. These experiments have also revealed an interesting phenomenon: Most of the performance overhead is due to the increase in packet size. In comparison, cryptographic computations have almost no effect on latency or throughput, since they can overlap with transmission [23]. This puts a limit on how much dedicated hardware helps; hardware reduces only the computational costs, not packet size.

Privacy: Privacy is a major concern in Sensor networks. The most obvious risk is that ubiquitous sensor technology might allow attacker to deploy secret surveillance networks for spying on unaware victims [23]. As surveillance technology has become cheaper and more effective, it has increasingly been implicated in privacy abuses. Advancement in technology will definitely make the problem worse with time. As devices get smaller in size, they will be easier to conceal; as devices get cheaper, surveillance networks will be more affordable. Accordingly there must be improvement in the research follow up to address this challenge.

Robustness to communication denial of service: Adversaries can severely limit the usefulness of a wireless sensor network through denial-of-service attacks [23]. In its simplest form, an adversary tries to disrupt the network's functionality by broadcasting a high-energy signal. If the transmission is powerful enough, the entire system's communication could be jammed. More sophisticated attacks are also possible. One standard defense against jamming employs spread-spectrum communication. However, cryptographically secure spread-spectrum radios are not commercially available. In addition, this defense is not secure against adversaries who might capture nodes and extract their cryptographic keys. The networked nature of sensor networks permits new, automated defenses against *denial of Service* (DoS). When the jamming affects only a part of the network, a

jamming-resistant network could defeat the attack by detecting the jamming, mapping the affected region, and then routing around the jammed area [23].

Secure routing: Routing and data forwarding is an essential service for enabling communication in sensor networks. Present routing protocols suffer from many security flaws [23]. For example, an attacker might launch denial-of-service attacks on the routing protocol, preventing communication. The simplest attacks involve injecting malicious routing information into the network, resulting in routing inconsistencies. Simple authentication might guard against injection attacks, but some routing protocols are susceptible to replay by the attacker of routing legitimate messages. Routing protocols are particularly susceptible to node-capture attacks. For instance, researchers have analyzed routing protocols used in WSN and observed that all are highly susceptible to node-capture attacks; in each case, the compromise of a single node is enough to take over the almost entire network or prevent any communication within it.

Resilience to node capture: One of the most challenging issues in WSN is how to provide resiliency against node capture attacks [23]. In traditional computing, physical security is often taken for granted; attackers are simply denied physical access to our computers. That paradigm is not applicable in case of WSN. In most of the applications, sensor nodes are likely to be placed in locations that are easily accessible to attackers. Such exposure raises the possibility that an attacker may capture sensor nodes, extract cryptographic keys, modify their code, or replace them with malicious nodes under the control of the attacker. Tamper-resistant packaging may be one defense, but it's expensive. Algorithmic solutions to the problem of node capture are preferred.

The challenge is to build networks that operate correctly despite the possibility of several nodes having being been compromised and thus might behave in an arbitrarily malicious way. A promising direction for building resilient networks is to replicate state across the network and use majority voting and other techniques to detect inconsistencies [23]. For example, several researchers have designed routing protocols that achieve some resilience against node capture by sending every packet along multiple, independent paths and checking at the destination for consistency among the packets that were received.

A second direction for resilience is to accumulate multiple, redundant views of the environment and crosscheck them for consistency. Defenses based on redundancy are particularly well suited to

sensor networks, as a large set of many cheap nodes are able to provide more reliable network operation than a small group of more sophisticated devices.

Network Security Services: Up to now, the low-level security primitives for WSN security are briefly mentioned. Now, high-level security mechanisms are secure group management, intrusion detection, and secure data aggregation etc.

Secure group management: Every node in a WSN is constrained in its computing and communication capabilities. However, the in-network data aggregation and analysis can be performed by groups of nodes. The actual nodes comprising this group, responsible for data aggregation and analysis, may change continuously. Many key services in WSN are also performed by groups. Consequently, secure protocols for group management are needed [23]. Also the secure inclusion of new group and supporting secure group communication are needed. The outcome of the group's computation is normally transmitted to a base station. The output must be authenticated to ensure that it comes from a valid group under the base station. A solution must also be efficient in terms of time and energy (or involve low computation and communication costs), precluding many classical group-management solutions.

Intrusion detection: A WSN is susceptible to many forms of intrusion. In wired networks, traffic and computation are generally monitored and analyzed for any possible anomaly at various concentration points. This is expensive in terms of the network's memory and energy consumption, as well as its inherently limited bandwidth [23]. In order to look for anomalies, applications and typical threat models must be understood and analyzed properly. It is important to understand how cooperating adversaries might attack the system. The use of secure groups may be a promising approach for decentralized intrusion detection.

Secure data aggregation. A benefit of a WSN is the subtle magnitude of sensing that large and dense sets of nodes can provide. The sensed values are aggregated to avoid overwhelming amounts of traffic reach to the base station [23]. Depending on the architecture of the WSN, aggregation may be carried out at many points in the network. All aggregation locations must be secured.

If the application tolerates approximate answers, powerful techniques are available; under appropriate trust assumptions, randomly sampling a small fraction of nodes and checking that they have behaved properly supports detection of many different types of attacks. In the following we discuss some possible threats to Wireless sensor network according to [25].

Passive Information Gathering: An intruder with a powerful receiver and well designed antenna can passively pick off the data stream.

Subversion of a Node: compromised sensor node may disclose its cryptographic keying material and sensor functionality may be available to the attacker. Secure sensor nodes must be tamper proof and should react to tampering.

False Node: An intruder might "add" a node to a system and feed false data or block the passage of true data. Typically, a false node is a computationally robust device that impersonates a sensor node.

Node Malfunctioning: A node in a WSN may malfunction and generate inaccurate or false data, or it may drop or garble packets in transit. Detecting and culling these nodes from WSN becomes an issue.

Denial of Service: A denial of service attack (*DoS attack*) is an event that causes weaken or reduces the network's capacity to execute its expected function. Protocols or design level vulnerabilities are the prime cause of DoS attacks [29]. Normally DoS attacks in wireless networks can occur at the physical layer, for example, via radio jamming. Also, the DoS attack can be launched in other layers or protocols also. Example of some DoS attacks includes: *Black Hole*, *Resource Exhaustation*, *Sinkholes*, *Induced Routing Loops*, *Wormholes*, and *Flooding* that are directed against the routing protocol employed by WSN.

Traffic Analysis: Although communications might be encrypted, an analysis of cause and effect, communication patterns and sensor activity might reveal enough information to enable the adversary to defeat or subvert the mission of WSN. Classically, traffic analysis is countered by communication systems employing *traffic flow security*. In this mode, the system transmits an encrypted stream continuously, encrypting idle messages when there is no valid traffic to be sent. With WSNs having limited-energy nodes, the practicality of traffic flow security becomes quite problematic. The effects of traffic analysis may be partially mitigated by encrypting the message header that contains addressing information.

1.5 KEY PREDISTRIBUTION

Since the communication between sensor nodes are wireless, it can easily be intercepted by an attacker. So these communications need to be secured. Due to resource constraints of sensor nodes, traditional means for security can't be applied in its entirety to the wireless sensor network. One solution is to distribute and install cryptographic keys in the sensor nodes before its deployment in the operational field. The problem of how to predistribute keys to the sensor nodes is known as key predistribution problem in wireless sensor network. Recently combinatorial design has been used widely for deterministic key predistribution.

Key predistribution means key is pre distributed in the sensor node before its deployment into the operational area. If two neighbor sensor nodes have the same key then they can secretly communicate with each other via symmetric key cryptography technique. We have already mentioned that symmetric key cryptography is ideal for secure communication in wireless sensor network due to limitation on the resources of sensor nodes. Researchers are working towards applying public key cryptography schemes such as elliptic key cryptography (ECC) for security in wireless sensor network. With the amazing advancement in the hardware technology, governed by *Moore's law*, this concept of using public key cryptography would be more common in near future.

When secured communication between two sensor nodes is required, then they can think of following either symmetric key cryptography or asymmetric key cryptography. Each of these techniques has some advantages and some disadvantages too. Asymmetric key cryptography requires huge computing resources that a tiny sensor node can't afford. So a symmetric key cryptography is preferred for secured communication in WSN. Again, key generation and distribution using Diffie-Hellman technique or public key infrastructure is also more or less infeasible in distributed wireless sensor network consisting of resource limited sensor nodes. So distributing key to a sensor node before their deployment is a good solution. So that a node can communicate with different nodes with different keys, its better idea that a set of keys be installed than only one key in the sensor node before its deployment. The universal set of keys for the network is called *key pool*. The set of keys that are predistributed to a node is called *key chain*. A key ID may be one dimensional value such as x or two-dimensional value such as (x, y). Generally the predistributed keys are merely key IDs, which are of much smaller length. A cryptographically

secured lengthy key is derived from a key ID. This process is known as *key generation*. The key ID or such parameters used for key generation is termed as keying elements. To map from key ID to actual key some function such as Hashing is used. In this thesis, we use the convention that whenever the term *key* is used it would mean the key ID that is obtained by key predistribution, with the understanding that mapping from a key ID to corresponding key is obvious.

In shared-key discovery phase if two nodes want to communicate and they are lying in one another's radio frequency range then they communicate to each other and find which one is the common key between them. To decide the common key, they must exchange some packets between them. This process is called *key establishment*. If two nodes N_i and N_j want to communicate with each other and there is no common key then they look for one or more intermediate nodes such that every pair of adjacent nodes share a common key, so that N_i and N_j are able to communicate with each other securely. This phase is known as *path-key establishment*. In our survey we focus on the key predistribution phase only.

1.6 METRICS OF KEY PREDISTRIBUTION SCHEME

Metrics to judge a given key predistribution scheme are: *connectivity*, *resiliency*, *scalability*, *complexity of shared-key discovery and path-key establishment.* Given any two nodes, *connectivity* defines the Probability that a common key between them. Resiliency refers to the sustainability of the sensor network when some of its nodes have compromised by attacker. There are two kinds of *resiliency*: E(s), and V(s). Given a sensor network, if s nodes are compromised, then

E(s) = number of communication links broken / total number of communication links,

V(s) = number of victim nodes / total number of nodes.

Scalability means given the same key pool and same key-chain length the ability to add new nodes in the distributed sensor network and assign them key-chains. Complexity of shared key discovery refers to the number of computation steps needed to find out the common key between any two sensor nodes. Complexity of path-key establishment refers to the number of computations needed to find common keys between adjacent intermediate nodes when there is no common key between two sensor nodes. We illustrate connectivity and resiliency using the example given below.

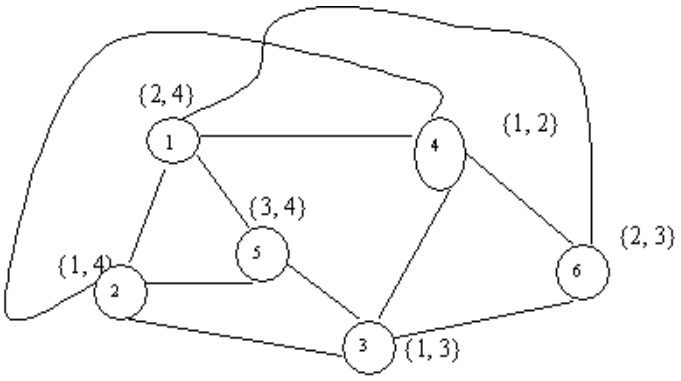

Figure 1.3: A sensor network of six sensor nodes with each node having two predistributed keys

In the above network there are six sensor nodes. The key pool is {1, 2, 3, 4}. Each sensor has two keys. We assume all these sensors are in the communication range of each other. So any two nodes in this network can communicate if they have a common key. For example, Node 3 and Node 4 share a common key 1. So there is a communication link between them. But nodes 4 and 5 do not share any key, so there is no communication link between them. In the above example, there are 6 nodes. So maximum number of connections possible in the network is $\binom{6}{2}$ = 15. The number of direct communication links that exist in the network is 11. So connection probability p = $\frac{11}{15}$ = 0.733.

In our example, let us suppose that s = 2 and attacker has captured the nodes 1 and 2. So she is able to extract the keys used by these nodes. This compromised key pool set is {1, 2, 4}. As a result we can check that among eleven communication links total nine communication links are compromised. So E(2) = $\frac{9}{11}$ = 0.818. And we see that due to this compromise of nodes 1 and 2, a different node that is node 4 is no more able to communicate with no other node. Because all keys of this node are compromised. In other words this node 4 has become victim node due to the compromise of nodes 1 and 2. Note that no other node turns victim node. So here V(2) = $\frac{1}{15}$ = 0.066.

1.7 EXISTING SCHEMES FOR KEY PREDISTRIBUTION

Solutions to predistribute keys and keying materials can be classified as:

(1) Probabilistic
(2) Deterministic
(3) Hybrid.

A sensor node can communicate with other node if the second one is lying within the circle of radio frequency range of the first one, and if both of them share a common key. In a key predistribution scheme, there are three phases. These are *key predistribution*, *shared-key discovery*, and *path-key establishment* [8]. In key predistribution phase a set of keys are chosen from a key pool following either in random order or in predetermined order. If the keys are drawn from the key pool in a random way, then the key predistribution scheme (KPS) is called *probabilistic key predistribution scheme*. If the keys are selected in a predetermined manner then it is *deterministic key predistribution scheme*. If the key predistribution mechanism is combination of both probabilistic and probabilistic approaches then it is *hybrid key predistribution scheme*.

1.7.1 PROBABILISTIC KEY PREDISTRIBUTION

The original solution is basic probabilistic key predistribution by Eschenauer et al. in [14]. It depends on probabilistic key sharing among the nodes of a random graph. In key establishment phase, two neighboring nodes exchange and compare to find a common key between them. Unlike schemes that use dedicated pair-wise keys, it may be possible in this solution that same key is used to secure more than one links.

Cluster key grouping scheme builds upon basic probabilistic scheme and divides key-chains into C clusters where each cluster has a *start key ID*. Remaining key IDs within a cluster are implicitly known from its start key ID. Only start key IDs are exchanged during shared-key discovery.

Transmission range adjustment scheme proposes sensor nodes to increase their transmission ranges during shared-key discovery phase. Nodes return to their original optimal transmission range once common keys are discovered.

Q-composite random KPS makes sure that a pair of neighboring sensor nodes should have more than q common keys to secure their link.

KPS using deployment knowledge scheme is based on the idea that distant sensor nodes do not need to have common keys in their key-chains.

1.7.2 DETERMINISTIC KEY PREDISTRIBUTION

There are many deterministic key predistribution schemes. Broadly, they can be classified into different categories that we mention below. One of relatively new classification is combinatorial design based key predistribution schemes. Our work is related with this combinatorial design based key predistribution. We will discuss some combinatorial design based key predistribution schemes in next chapter. Here we briefly mention other traditional deterministic key predistribution schemes.

Matrix based KPS: It generates an m x m key matrix for a WSN of size $N=m^2$. Each sensor is assigned a position (i, j), and receives all the keys in i^{th} column and in j^{th} row as its key-chain (total 2m keys). During shared-key discovery phase, any pair of sensor nodes exchange their position information (i, j) and (u, v) to locate common dedicated pair-wise keys $K_{i,v}$ and $K_{u,j}$. *Blom's Scheme* uses a public $(\lambda+1)$ x N matrix G and a private $(\lambda+1)$ x $(\lambda+1)$ symmetric matrix D which are both generated over GF(q) [2]. All possible link keys in a network of size N are represented as an N x N symmetric key matrix $k = (D \times G)^T \times G$. This solution is λ-secure. In key-setup phase, node S_i stores $column_i$ from G as public information, and row_i from $(D \times G)^T$ as private information. During key establishment phase, a pair of sensor nodes (S_i, S_j) first exchange their public information and then generate dedicated pair-wise key. S_i generate $K_{i,j} = row_i \times column_j$, and S_j generates $K_{j,I} = row_j \times column_i$. So $K_{i,j} = K_{j,i}$. Scalability of Blom's scheme is improved in *multiple space Blom's scheme* (MBS). This scheme assumes that underlying physical network graph is a complete bipartite graph.

Polynomial based Key Predistribution Scheme: It distributes a polynomial share (a partially evaluated polynomial) to each sensor node by using which every pair of nodes can generate a dedicated pair-wise key. Symmetric polynomial P(x, y) (i.e., P(x,y)=P(y,x)) of degree λ is used. The $\lambda+1$ number of coefficients are from GF(q). In key setup phase, sensor node S_i obtains its polynomial share P(i,y). To establish a common key with node S_j, S_i evaluates P(i,j). This solution is λ-secure.

Location-based Pair-wise Keys Scheme: It uses bivariate polynomials considering deployment knowledge during key setup phase. Deployment area is divided into R rows and C columns, total R x C cells. The scheme is based on *polynomial based key pre-distribution scheme*. For each cell at c^{th} column and r^{th} row, a unique polynomial $f_{c,r}(x,y)$ is generated. Each sensor node stores polynomial share of its home cell and four immediate neighboring cells, total of five polynomials. During key establishment phase, two sensor nodes simply exchange their cell coordinates to agree on a polynomial share.

1.7.3 HYBRID KEY PREDISTRIBUTION

In hybrid solutions, keys and keying materials are distributed to sensor nodes based on both probabilistic and deterministic techniques. Probabilistic part generally helps to improve scalability and key resilience while the deterministic part improves the key connectivity.

Multiple space key predistribution scheme improves resilience of Blom's scheme: It uses a public matrix G and a set of ω private symmetric matrices D. these matrices form w spaces (D_i,G) for i=1,…, ω (i.e. i^{th} space uses private symmetric matrix D_i and public matrix G. In key setup phase, for each sensor node, a set of τ spaces are randomly selected among these ω spaces. Thus, each sensor node stores τ private row information and public column information. In shared-key discovery phase, each pair of nodes agrees on a common space for which they have to exchange an extra message which includes τ space IDs. Once agreed upon a space, during key establishment phase, a dedicated pair-wise key is generated as in Blom's scheme. It is possible that a pair of nodes does not share a common space; in that case they have to establish a pair-wise key through one or more secure paths connecting them.

Polynomial pool-based key predistribution scheme: It considers the fact that not all pairs of sensor nodes have to establish a key. It combines *polynomial based key predistribution scheme* with the pool idea to improve resiliency and scalability. In key setup phase, a pool F of λ-degree symmetric polynomials over finite field GF(q) is generated. For each sensor node S_i, a subset $F_i \subseteq$ Fof polynomials is picked and polynomial shares $f_i(y) = P(x=i,y)$ for each $P(x,y) \in F_i$ are generated.

Key management in hierarchical wireless sensor network: A Hierarchical WSN (HWSN) includes one or more computationally robust base stations. Sensor nodes are deployed in one or two-hop neighborhood around base stations or resource rich sensor nodes (called cluster heads). Base stations are usually assumed to be trusted and used as the key distribution centers. In a HWSN, pair-wise, group-wise and network-wise keys are required to secure unicast, multicast and broadcast types of communications among sensor nodes, cluster heads and base stations.

1.8 MOTIVATION

The algebraic properties of some combinatorial design help us to get suitable deterministic key predistribution schemes for distributed wireless sensor network. To map a particular combinatorial design to key predistribution scheme, the universal set of design acts as key pool of sensor network, blocks are mapped to key chain of individual sensor nodes. Chakrabarti et al. first gave a novel idea of merging blocks of a particular design and then assign these merged blocks to individual sensor nodes before their deployment [9]. Due to this merging of blocks, now every node has to take extra burden of storing all the keys, but its connectivity and resiliency improves considerably. We have to decide a merging factor; that is the number of blocks that are merged together. Now *the question is how to select blocks for merging*. Our motivation was to devise a deterministic merging scheme. We have proposed a deterministic merging scheme for transversal design based key predistribution scheme. Another motivation was to examine other combinatorial design based key predistribution schemes to find if merging of blocks would be helpful to improve their performances or not.

1.9 THESIS LAYOUT

This thesis includes the work done in three technical reports [20], [21] and [22]. A brief outline of the chapters in this thesis is given in this section.

Chapter 1 has described about the wireless sensor network, types of wireless sensor networks, security issues in wireless sensor network, key predistribution, metrics to judge a key predistribution scheme, few existing key predistribution schemes, and the motivation for this thesis work.

Chapter 2 briefly introduces the subject combinatorial design. Various combinatorial designs that have been used in key predistribution schemes such as transversal design, partially balanced incomplete block design, orthogonal array, projective plane, and t-design are mentioned. Then two key predistribution schemes, namely transversal design based key predistribution scheme and orthogonal array based key predistribution schemes are described briefly.

Chapter 3 discusses about applicability of merging of blocks for possible improvement of resiliency and connectivity in orthogonal array based key predistribution scheme. We have observed that this orthogonal array based key predistribution has very poor resiliency. Then this chapter mentions our realization that merging of nodes doesn't help to improve resiliency due to small key pool size in orthogonal array based key predistribution scheme. This chapter reflects our work in [20].

Chapter 4 mentions the existing heuristic technique for merging of blocks to improve both connectivity and resiliency. Then our proposal for deterministic merging technique for transversal design based key predistribution scheme has been described. Then it describes a deterministic key establishment algorithm for transversal design based key predistribution scheme, proposed by us. This chapter is based on our work [22].

Chapter 5 concludes this thesis.

2 COMBINATORIAL DESIGN BASED KEY PREDISTRIBUTION SCHEMES

The subject combinatorial design found its application initially in the design of experiments in statistics for long days. Then it was used extensively in the field of coding theory. Also combinatorial design techniques have been used in numerous other area of computer science such as Boolean function, authentication code, visual cryptography, multiple accesses to channel, software testing etc. For detail on these areas where combinatorial designs have been used, one can refer [1]. In this chapter we will discuss the works done on key predistribution schemes that are devised following different combinatorial design techniques. For detail discussion on combinatorial design one can refer [30] and [31]

2.1 INTRODUCTION TO COMBINATORIAL DESIGN

Combinatorial design theory is interested in arranging elements of a finite set into subsets to satisfy certain properties. It is the study of families of subsets with various prescribed regularity properties. Members of the universal set S in a combinatorial design are usually called *treatments*, or *varieties*, and the subsets chosen are called *blocks*. A *regular design* based on a v-set S is a collection of k-sets from S such that every member of S belongs to r of the k-set blocks. It is usual to write b for the number of blocks in a design. So a regular design has four parameters: *v*, *b*, *r*, and *k*. However these parameters are not independent. A regular design is represented as (*v*, *b*, *r*, *k*)-design [30].

In any regular design, $b*k = v*r$.

A block design is proper if all of its block have same length. The number of blocks that contain a given treatment is the replication number, *r*. If all v treatments occur in a block of a design, then the block is called *complete*. If a regular design has the property this property then that design is called *complete design*. Complete design is of very little interest unless some further structure is imposed (such as in *Latin Square*). In a design, if at least one block is incomplete then that design is an *incomplete* design. If v = b, the design is called *symmetric*.

If x and y are any two different treatments in an incomplete design, we shall refer to the number of blocks that contain both x and y as the *covalency* of x and y, and write it as λ_{xy}. Many important designs are concerned with this covalency function. In BIBD (discussed below) this λ_{xy} is constant.

2.2 VARIOUS KINDS OF COMBINATORIAL DESIGNS

Here we briefly mention the combinatorial designs that have been applied in the field of key predistribution in wireless sensor network till recently. Before mentioning those combinatorial designs, we discuss balanced incomplete block design whose properties are easy to comprehend.

2.2.1 BALANCED INCOMPLETE BLOCK DESIGN

A Balanced Incomplete Block Design (BIBD) is an arrangement of v distinct objects into b blocks such that each block contains exactly k distinct objects, each object occurs in exactly r different blocks, and every pair of distinct objects occurs together in exactly λ blocks. The design can be expressed as (v, k, λ), or equivalently (v, b, r, k, λ), where: $\lambda(v-1) = r(k-1)$ and $bk = vr$ [30].

Following is an example of (9,12,4,3,1)-BIBD:

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

A BIBD is symmetric BIBD when b=v and therefore r=k. It is represented as (v, k, λ)-design. For every prime power $q \geq 2$, there exists a symmetric design $(q^2+q+1, q+1, 1)$.

2.2.2 TRANSVERSAL DESIGN

A *Group-divisible design* of type g^u and block sizek is a triplet (X, H, A), where X is a finite set of cardinality gu, H is a partition of X into u parts (called groups) of size g, and A is a set of subsets of X (called blocks), that satisfy following properties [30] and [31]:

1) $|H_1 \cap A_1| \leq 1$ for $H_1 \in H$ and every $A_1 \in A$.

2) Each pair of elements of X from different groups occurs in exactly one block in A.

A transversal design (TD) is a group-divisible design. A TD(k, n) is equivalent to a set of k-2 mutually orthogonal latin squares of order n. A *(v, b, r, k)-configuration* is a (v, b, r, k)-1 design such that the number of common point between any two blocks is either zero or one. A (v, b, r, k, λ)-BIBD is a (v, b, r, k)-1 design such that any pair of points occur in exactly λ blocks [30].

2.2.3 PARTIALLY BALANCED INCOMPLETE BLOCK DESIGN

An association scheme with m associate classes on the set X is a family of m symmetric anti-reflexive binary relations on X such that [31]:

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, $1 \leq i \leq m$, and
3. for each i, if x and y are i associates, then there are p^i_{jl} elements of X which are both j-th associates of x and *l*-associates of y. The numbers vi $(1 \leq i \leq m)$ and p^i_{jl} $(1 \leq i,j,l \leq m)$ are called the *parameters of the association schemes*.

A *triangular association scheme* is a partially balanced design with two associate classes [31]. Here the association scheme is an array of n rows and n columns, and number of varieties is n(n-1)/2. It has the following properties:

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

A *partially balanced incomplete block design with m associate classes*, denoted by PBIBD(m) 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 ii-th ($1 \leq i \leq m$) associates, then they occur together in λ_i blocks [31]. Such design is denoted by PB[k, λ_1, λ_2, … , λ_m; v].

2.2.4 ORTHOGONAL ARRAY

An N x K array A with elements from S (|S|=s) is said to be OA with s levels, strength t and index λ ($0 \leq t \leq k$) if every N x t sub-array of A contains each t-tuple based on S exactly λ times as a row. When $\lambda=1$ we say OA has index unity. N, k, s, t, λ are parameters of OA. λ can be derived from other parameters ($\lambda=N/s^t$). So an OA is represented as OA(N, k, s, t).

3.2.5 PROJECTIVE PLANE

A *Projective Plane* consists of a set of lines and a set of points, and a relation between points and lines called incidence, having the following properties [5]:

(i) Given any two distinct points, there is exactly one line incident with both of them.
(ii) Given any two distinct lines, there is exactly one point incident with both of them.
(iii) Every point has q+1 lines through it.
(iv) Every line contains q+1 points.

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

A *projective space* PG(d, q) of dimension d over a field F is constructed from the vector space of dimension d+1 over F as follows [5]:

(i) Objects are all subspaces of the vector space,

(ii) Two objects are incident if one contains the other,

Subspaces with dimension 1, 2, and d are called points, lines, and hyperplanes respectively. A *partial linear space* arranges objects into subspaces in the form of incidence structure. These subsets are called lines. In a partial linear space [5]:

(i) Each line is incident with at least two points, and

(ii) Any two points are incident with at most one line.

A *partial geometry* [5] is a partial linear space where:

(i) Number of points on a line is constant,

(ii) Number of lines through a point is constant, and

(iii) Given a point x on a line L, the number of points on L that are collinear with x is constant.

Finite generalized quadrangle (GQ) is a special class of partial geometry.

3.2.6 t-DESIGN

A t-design is a design (X, A) such that the following properties are satisfied [30]:

(i) $|X| = v$,

(ii) each block contains exactly k points, and

(iii) every set of t distinct points is contained in exactly λ blocks.

The general term t-design is used to denote any t-(v, k, λ)-design. A 2-(v, k, λ)-design is just a (v, k, λ)-BIBID.

2.3 KEY ESTABLISHMENT AND TRUST SETUP

When setting up a sensor network, one of the first requirements is to establish cryptographic keys for later use. Researchers have proposed a variety of protocols over several decades for this well-studied problem. Why can't the same key-establishment protocols be used in sensor networks? The inherent properties of sensor networks render previous protocols impractical. Many current sensor devices have limited computational power, making public-key cryptographic primitives too expensive in terms of system overhead. Key-establishment techniques need to scale to networks with hundreds or thousands of nodes. Moreover, the communication patterns of sensor networks differ from traditional networks; sensor nodes may need to set up keys with their neighbors and with data aggregation nodes.

The simplest solution for key establishment is a network-wide shared key. Unfortunately, the compromise of even a single node in a network would reveal the secret key and thus allow decryption of all network traffic. One variant on this idea is to use a single shared key to establish a set of link keys, one per pair of communicating nodes, then erase the network-wide key after setting up the session keys. However, this variant of the key-establishment process does not allow addition of new nodes after initial deployment. Public-key cryptography (such as Diffie-Hellman key establishment) is another option beyond the capabilities of today's sensor networks. Its main advantage is that a node can set up a secure key with any other node in the network.

Yet another approach is to pre-configure the network with a shared unique symmetric key between each pair of nodes, though it doesn't scale well. In a sensor network with n nodes, each node needs to store n - 1 keys, and total $n * (n- 1)/2$ keys need to be established in the network.

Bootstrapping keys using a trusted base station is another option. Here, each node needs to share only a single key with the base station and set up keys with other nodes through the base station [6]. This arrangement makes the base station a single point of failure, but because there is only one base station, the network may incorporate tamper-resistant packaging for the base station, ameliorating the threat of physical attack.

Eschenauer and Gligor developed random-key predistribution protocol in which a large pool of symmetric keys is chosen and a random subset of the pool is distributed to each sensor node [14]. Two nodes that want to communicate search their pools to determine whether they share a common key; if they do, they use it to establish a session key. Not every pair of nodes shares a common key, but if the key-establishment probability is sufficiently great, nodes can still set up keys with sufficiently many nodes to obtain a fully connected network. This means of establishing keys

avoids having to include a central trusted base station. This scheme is sometimes referred as *basic scheme*, and this basic scheme led the idea of applying combinatorial design for in key predistribution.

The disadvantage of this approach is that attackers who compromised sufficiently many nodes could also reconstruct the complete key pool and break the scheme. In the future, we expect to see research on better random-key predistribution schemes providing resilience to node compromise, as well as investigation of hardware support for public-key cryptography and more efficient public-key schemes (such as elliptic curve cryptography).

Ultimately, we need a secure and efficient key-distribution mechanism allowing simple key establishment for large-scale sensor networks.

With the continuous enhancement in the capacity of sensors, It is certain that in future use of public key cryptography such as elliptic curve based cryptography is feasible for secure communication. But till now, symmetric key cryptography is the most viable option for secured communication.

Sensor nodes can form a secured WSN by

1) using predistributed keys,
2) exchanging information with intermediate neighbors, or
3) exchanging information with computationally robust nodes.

2.4 COMBINATORIAL DESIGN BASED KEY PREDISTRIBUTION

The first key predistribution scheme that used combinatorial design is by Camtepe and Yener [5]. They have used projective plane and generalized quadrangle to predistribute keys. Key predistribution scheme proposed by Lee and Stinson uses *Transversal design* [17]. PBIBD based key predistribution scheme has been proposed by Ruj, and Roy in [27]. Ruj and Roy have also proposed some key establishment algorithms for combinatorial design based key predistribution schemes [28]. J. Dong et al. has proposed orthogonal array based KPS in [12]. 3-design based KPS has been proposed in [13] by J. Dong, D. Pei et al. A very good article by Martin on applicability of combinatorial designs in key predistribution can be found in [19]. To get overview with examples of these combinatorial design based key predistribution schemes one can refer Pattanayak and

Majhi in [21]. Our work is related with transversal design and orthogonal array. We describe the transversal design based key predistribution in this chapter. To maintain flow of the next chapter we have included the orthogonal array based key predistribution scheme in next chapter.

2.4.1 TRANSVERSAL DESIGN BASED KEY PREDISTRIBUTION SCHEME

This scheme is proposed by Lee and Stinson [17]. This scheme uses *Transversal design*, which is a special kind of group-divisible design. Here, any two distinct blocks intersect in zero or one point.

2.4.1.1 Method To Derive TD(t,p)

We write the method to construct transversal design in next page as given in [17].

Step 1. Define $X = \{ 0, 1, \ldots , k-1\} \times Z_p$.

Step 2. For $0 \leq x \leq k-1$, define $H_x = \{ x \} \times Z_p$.

Step 3. Define $H = \{ H_x \mid 0 \leq x \leq k-1 \}$.

Step 4. For every ordered pair $(i, j) \in Z_p \times Z_p$,

define a block $A_{i,j} = \{ (x, ix+j \bmod q) \mid 0 \leq x \leq k-1 \}$.

Step 5. Obtain $A = \{ A_{i,j} \mid (i, j) \in Z_p \times Z_p \}$.

Step 6. (X, H, A) is a TD(k, p).

2.4.1.2 Example of Transversal Design based KPS

To understand the above algorithm, we give the following example showing how do the blocks are formed. In this example number of blocks = 49, block size = 4, a prime power = 7. Now the blocks are constructed as below:

$A_{0,0}$: (0, 0), (1, 0), (2, 0), (3, 0) $A_{0,1}$: (0, 1), (1, 1), (2, 1), (3, 1)

$A_{0,2}$: (0, 2), (1, 2), (2, 2), (3, 2) $A_{0,3}$: (0, 3), (1, 3), (2, 3), (3, 3)

$A_{0,4}$: (0, 4), (1, 4), (2, 4), (3, 4) $A_{0,5}$: (0, 5), (1, 5), (2, 5), (3, 5)

$A_{0,6}$: (0, 6), (1, 6), (2, 6), (3, 6) $A_{1,0}$: (0, 0), (1, 1), (2, 2), (3, 3)

$A_{1,1}$: (0, 1), (1, 2), (2, 3), (3, 4) $A_{1,2}$: (0, 2), (1, 3), (2, 4), (3, 5)

$A_{1,3}$: (0, 3), (1, 4), (2, 5), (3, 6) $A_{1,4}$: (0, 4), (1, 5), (2, 6), (3, 0)

$A_{1,5}$: (0, 5), (1, 6), (2, 0), (3, 1) $A_{1,6}$: (0, 6), (1, 0), (2, 1), (3, 2)

$A_{2,0}$: (0, 0), (1, 2), (2, 4), (3, 6) $A_{2,1}$: (0, 1), (1, 3), (2, 5), (3, 0)

$A_{2,2}$: (0, 2), (1, 4), (2, 6), (3, 1) $A_{2,3}$: (0, 3), (1, 5), (2, 0), (3, 2)

$A_{2,4}$: (0, 4), (1, 6), (2, 1), (3, 3) $A_{2,5}$: (0, 5), (1, 0), (2, 2), (3, 4)

$A_{2,6}$: (0, 6), (1, 1), (2, 3), (3, 5) $A_{3,0}$: (0, 0), (1, 3), (2, 6), (3, 2)

$A_{3,1}$: (0, 1), (1, 4), (2, 0), (3, 3) $A_{3,2}$: (0, 2), (1, 5), (2, 1), (3, 4)

$A_{3,3}$: (0, 3), (1, 6), (2, 2), (3, 5) $A_{3,4}$: (0, 4), (1, 0), (2, 3), (3, 6)

$A_{3,5}$: (0, 5), (1, 1), (2, 4), (3, 0) $A_{3,6}$: (0, 6), (1, 2), (2, 5), (3, 1)

$A_{4,0}$: (0, 0), (1, 4), (2, 1), (3, 5) $A_{4,1}$: (0, 1), (1, 5), (2, 2), (3, 6)

$A_{4,2}$: (0, 2), (1, 6), (2, 3), (3, 0) $A_{4,3}$: (0, 3), (1, 0), (2, 4), (3, 1)

$A_{4,4}$: (0, 4), (1, 1), (2, 5), (3, 2) $A_{4,5}$: (0, 5), (1, 2), (2, 6), (3, 3)

$A_{4,6}$: (0, 6), (1, 3), (2, 0), (3, 4) $A_{5,0}$: (0, 0), (1, 5), (2, 3), (3, 1)

$A_{5,1}$: (0, 1), (1, 6), (2, 4), (3, 2) $A_{5,2}$: (0, 2), (1, 0), (2, 5), (3, 3)

$A_{5,3}$: (0, 3), (1, 1), (2, 6), (3, 4) $A_{5,4}$: (0, 4), (1, 2), (2, 0), (3, 5)

$A_{5,5}$: (0, 5), (1, 3), (2, 1), (3, 6) $A_{5,6}$: (0, 6), (1, 4), (2, 2), (3, 0)

$A_{6,0}$: (0, 0), (1, 6), (2, 5), (3, 4) $A_{6,1}$: (0, 1), (1, 0), (2, 6), (3, 5)

$A_{6,2}$: (0, 2), (1, 1), (2, 0), (3, 6) $A_{6,3}$: (0, 3), (1, 2), (2, 1), (3, 0)

$A_{6,4}$: (0, 4), (1, 3), (2, 2), (3, 1) $A_{6,5}$: (0, 5), (1, 4), (2, 3), (3, 2)

$A_{6,6}$: (0, 6), (1, 5), (2, 4), (3, 3)

2.4.1.3 μ-Common Intersection Designs (μ-CID)

Suppose that two nodes N_i and N_j are located within each other's communication range, but do not share a common key. So we look for a common neighbor N_h that shares a common key with both N_i and N_j.

Suppose (X, A) is a (v, b, r, k)-configuration. Remember that A *(v, b, r, k)-configuration* is a (v, b, r, k)-1 design such that the number of common point between any two blocks is either zero or one. A (v, b, r, k, λ)-BIBD is a (v, b, r, k)-1 design such that any pair of points occur in exactly λ blocks [30]. Then (X, A) is a μ-CID provided that

$$\left| \{ A_h \in A \mid A_i \cap A_h \neq \phi \text{ and } A_j \cap A_h \neq \phi \} \right| \geq \mu \quad \text{Whenever } A_i \cap A_j = \phi.$$

It is always preferable to construct a (v, b, r, k)-configuration with μ as high as possible. This maximum μ is denoted as $\mu^*(v, b, r, k)$.

2.3.1.4 Analysis of KPS based on Transversal Design

Let (v, b, r, k)-configuration, which is a μ-CID, is used for KPS. Suppose Ni and N_j are two nodes located within each other's communication range. The probability that Ni and N_j share a common key is

$$P = \left(\frac{k*(r-1)}{b-1} \right)$$

If the number of nodes, compromised by attacker, is s then resiliency V(s) is as follows:

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

3 NON APPLICABILITY OF MERGING OF BLOCKS IN ORTHOGONAL ARRAY BASED KEY PREDISTRIBUTION

J Dong et al proposed key predistribution scheme based on orthogonal array design in [12]. Before discussing on the unsuitability of this scheme for merging of blocks, we present the orthogonal array based key predistribution scheme.

3.1 ORHTOGONAL ARRAY BASED KEY PREDISTRIBUTION SCHEME

Our motivation was to check if merging of blocks in orthogonal array can improve the connectivity and resiliency offered by orthogonal array based key predistribution scheme. To do that we have simulated the orthogonal array key predistribution scheme. J. Dong et al. have used OA of index one. Further they have used *Bush's construction* for constructing OA of index one. In the following, we write the Bush's orthogonal array Construction method of index unity.

3.1.1 Orthogonal Array of Index Unity using Bush's Construction

Step 1: Start.

Step 2: GF(s) is a Galois Field with $s=q^n$ elements, where q is a prime number and n is a positive integer. These elements are denoted by e_i for i=0, 1, … ,s-1.

Step 3: Consider the polynomial

$$y_j(x)=a_{t-1}*x^{t-1} + a_{t-2}*x^{t-2} + \ldots + a_1*x + a_0, \text{ where } a_i \in GF(s)$$

Step 4: So we can have s^t polynomials (j=0,1,…,$s^t - 1$).

Step 5: Form an s by s^t array by inserting u at OA[i, j] such that

$$y_j(e_i)= e_u \bmod q.$$

Step 6: Stop.

3.1.2 Example of Bush's Construction

Let $q^n=5$ and t=2.. Note that q always has to be a prime number. So q^n is a prime power, where n is a positive integer. So we get s=5, and $e_0=0$, $e_1=1$, $e_2=2$, $e_3=3$, $e_4=4$, and the polynomial is $y_j(x)= a_1*x + a_0$, where $a_i \in$ GF(5). So there are 25 polynomials such as:

$y_0(x)= 0*x + 0$

$y_1(x)= 0*x + 1$

$y_2(x)= 0*x + 2$

$y_3(x)= 0*x + 3$

$y_4(x)= 0*x + 4$

$y_5(x)= 1*x + 0$

$y_6(x)= 1*x + 1$

$y_7(x)= 1*x + 2$

$y_8(x)= 1*x + 3$

$y_9(x)= 1*x + 4$

$y_{10}(x)= 2*x + 0$, and so on.

3.1.3 Construction of the orthogonal Array

Suppose we take a polynomial $y_8(x)= x +3$

So, $y_8(0) = 3$

$y_8(1) = 4$

$y_8(2) = 5 \bmod 5 = 0$

$y_8(3) = 6 \bmod 5 = 1$

$y_8(4) = 7 \bmod 5 = 2$

so we get the 8^{th} row of array: 3 4 0 1 2 y

this y i.e., OA[i,q+1] term is filled up as the coefficient of the leading term. So here y=1.

3.1.4 The OA based Key Predistribution Scheme

This OA is used to construct a combinatorial design (X, B) with v = |X| = q*(q+1) = q^2+1, b = q^t , k = q+1. That is, Size of key pool = q^2+1, number of sensor nodes = q^t, number of keys in each node = q+1. Elements from different columns of OA are considered as different elements of OA. So, Key Pool, X = $\{a_{i,j} \mid 0 \leq i \leq q-1, 0 \leq j \leq q\}$. For example, we take the same example as above, i.e. q^n=5, t=2. The resultant orthogonal array following Bush's construction is shown in the table below.

Column 0	Column 1	Column 2	Column 3	Column 4	Column 5
0	0	0	0	0	0
1	1	1	1	1	1
2	2	2	2	2	2
3	3	3	3	3	3
4	4	4	4	4	4
0	1	2	3	4	1
1	2	3	4	0	1
2	3	4	0	1	1
3	4	0	1	2	1
4	0	1	2	3	1
0	2	4	1	3	2
1	3	0	2	4	2
2	4	1	3	0	2
3	0	2	4	1	2
4	1	3	0	2	2
0	3	1	4	2	3
1	4	2	0	3	3
2	0	3	1	4	3
3	1	4	2	0	3
4	2	0	3	1	3
0	4	3	2	1	4
1	0	4	3	2	4
2	1	0	4	3	4
3	2	1	0	4	4
4	3	2	1	0	4

Table 3.1: An orthogonal array obtained following Bush's construction

In the KPS, the elements from different columns of OA are considered as different elements of OA. That is, in the above example, elements of the first row would become (0,0), (0,1), (0,2), (0,3), (0,4), (0,5). So we get a two-dimensional array of elements of the form (x,y). Each of these elements is a key identifier. Some mapping technique is used to transform this key identifier to a cryptographic key of sufficient bit-length. In KPS literature, this key-identifier is loosely referred as key. Here key pool is the set $\{(x,y) \mid 0 \leq x \leq 4, \text{ and } 0 \leq y \leq 5\}$. Each row represents the key-chain that is assigned to a particular sensor node. So the resultant key predistribution is shown in the following table.

Sensor Node	Key-chain					
	Key_0	Key_1	Key_2	Key_3	Key_4	Key_5
0	(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)
1	(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)
2	(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)
3	(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)
4	(4,0)	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)
5	(0,0)	(1,1)	(2,2)	(3,3)	(4,4)	(1,5)
6	(1,0)	(2,1)	(3,2)	(4,3)	(0,4)	(1,5)
7	(2,0)	(3,1)	(4,2)	(0,3)	(1,4)	(1,5)
8	(3,0)	(4,1)	(0,2)	(1,3)	(2,4)	(1,5)
9	(4,0)	(0,1)	(1,2)	(2,3)	(3,4)	(1,5)
10	(0,0)	(2,1)	(4,2)	(1,3)	(3,4)	(2,5)
11	(1,0)	(3,1)	(0,2)	(2,3)	(4,4)	(2,5)
12	(2,0)	(4,1)	(1,2)	(3,3)	(0,4)	(2,5)
13	(3,0)	(0,1)	(2,2)	(4,3)	(1,4)	(2,5)
14	(4,0)	(1,1)	(3,2)	(0,3)	(2,4)	(2,5)
15	(0,0)	(3,1)	(1,2)	(4,3)	(2,4)	(3,5)
16	(1,0)	(4,1)	(2,2)	(0,3)	(3,4)	(3,5)
17	(2,0)	(0,1)	(3,2)	(1,3)	(4,4)	(3,5)
18	(3,0)	(1,1)	(4,2)	(2,3)	(0,4)	(3,5)
19	(4,0)	(2,1)	(0,2)	(3,3)	(1,4)	(3,5)
20	(0,0)	(4,1)	(3,2)	(2,3)	(1,4)	(4,5)
21	(1,0)	(0,1)	(4,2)	(3,3)	(2,4)	(4,5)
22	(2,0)	(1,1)	(0,2)	(4,3)	(3,4)	(4,5)
23	(3,0)	(2,1)	(1,2)	(0,3)	(4,4)	(4,5)
24	(4,0)	(3,1)	(2,2)	(1,3)	(0,4)	(4,5)

Table 3.2: Key predistribution following J. Dong et al. scheme

With continuation from the discussion on orthogonal array based key predistribution scheme in last chapter, we see that in [12], Dong et al. have compared their scheme with other existing KPS's based on two parameters:

(1) Connection Probability
(2) Probability fail(1)

The parameter fail(1) denotes the probability of a node becomes victim node when a single node is compromised. In the following table we present the connection probability and resiliency of this scheme.

Value of q	Value of t	N	Connection Probability, P	E(1)	V(1)
5	3	125	0.707	0.218	0.707
5	4	625	0.757	0.216	0.757
7	3	343	0.648	0.963	0.733
7	4	2401	0.726	0.148	0.0004
7	5	16807	0.715	0.148	0.00006
11	2	121	0.992	0.0916	0.00826
11	3	1331	0.593	0.092	0.0007
11	4	14641	0.701	0.092	0.00006
13	2	169	0.994	0.774	0.0059
13	3	2197	0.579	0.077	0.0004
13	4	28561	0.695	0.0776	0.000035

Table 3.3: Measurement of Connection Probability and of Resiliency when 1 node is compromised in J. Dong et al. scheme

3.2 UNSUITABILITY OF ORTHOGONAL ARRAY BASED KPS FOR MERGING OF BLOCKS

Upon simulation of this scheme we observe very serious flaw in the scheme. When we compromise 5% or 10% of the total number of sensor nodes, which is very natural, we see that resiliency becomes very poor. Both the E(s) and V(s) becomes close to 1.0. That means all the connections are getting compromised if we compromise only 5% or 10% of nodes. Also almost all the remaining sensor nodes are becoming victim nodes in the sense they can no longer communicate with other nodes as their all keys get compromised. In the following table we summarize these observations.

Value of q	Value of t	N	Connection probability	No. of compromised nodes, s	E(s)	V(s)
5	2	25	0.966	3	0.554	0.12
5	2	25	0.966	5	0.698	0.20
5	3	125	0.707	7	0.798	0.21
5	3	125	0.707	13	0.982	0.872
7	3	343	0.648	17	0.963	0.733
7	3	343	0.648	34	1.0	0.998
7	4	2401	0.726	120	1.0	0.999
7	4	2401	0.726	240	1.0	0.999

Table 3.4: Resiliency of the KPS when 5% and 10% nodes are compromised

Since number of nodes, N = q*t, we can get larger N by either increasing q or by increasing t or by increasing both. In the above simulation we have taken prime power q small. For similar values of N, we wanted to study if there is any change in the performance of the scheme if we take prime power q larger and keep t small. We observe that the scenario does not change much. The result of simulation is shown below.

Value of q	Value of t	N	Connection probability p_1	Number of compromised nodes, s	E(s)	V(s)
11	2	121	0.992	6	0.442	0.049
11	2	121	0.992	12	0.687	0.099
19	2	361	0.997	18	0.632	0.049
19	2	361	0.997	36	0.863	0.111
47	2	2209	0.999	110	0.913	0.054
47	2	2209	0.999	220	0.993	0.745

Table 3.5: Resiliency of the KPS when 5% and 10% nodes are compromised with changed parameters for similar N as of Table 3.4.

The reason for such poor resiliency is the small key pool size. This small key pool size is also the reason of good connectivity of this OA based KPS. This key pre-distribution scheme is suitable only for sensor network of very small size. Now due to this small key pool size, if we want to apply merging there will be high degree of connectivity between most of the sensor node pairs. And as a result, the compromise of few nodes would severely affect the resiliency of the network.

4 MERGING OF BLOCKS

4.1 MOTIVATION

We have seen that by following a suitable combinatorial design we can have a good key predistribution scheme. If we follow any such particular combinatorial design, the number of common keys among any two nodes is fixed. For example in Projective plane based KPS the common key between any pair of node is always one. In transversal design based KPS the number of common key between any pair of nodes is either 0 or 1. In PBIBD based KPS with number of nodes $N = n(n-1)/2$, any two nodes share either $n-2$ or four common keys.

The intuition for merging of blocks was that if we can merge a fixed number of blocks, and then by predistribute those merged blocks to sensor nodes as their key-chains, we could improve the connectivity and resiliency of KPS. Chakrabarty, Maitra, and Roy [9] devised a general merging scheme that can be applied to any combinatorial design based key predistribution scheme. But the resultant key predistribution may not be always effective realized their intuition on transversal design based KPS and they were able to improve both connectivity and resiliency significantly. Only disadvantage is that due to merging of blocks, every node will now have to store more number of keys in its memory.

4.2 PRIOR WORK ON MERGING OF BLOCKS

In KPS based on transversal design, common key between any two nodes is at most one. Chakrabarti et al observed this and thought of increasing this connectivity. For this they devised a hybrid scheme based on transversal design [8]. In this scheme a merging factor z is decided. And z many blocks are merged to form a merged-block and this forms the key-chain for a sensor node. In this way not only they increased the connectivity provided by the KPS but also improved the resiliency offered. Below we present their heuristic algorithm for merging of blocks which works better than the scenario when blocks are merged randomly [8].

1. flag = true; count = 0; all the blocks are marked as unused;

2. an array node[..] is available, where each element of the array can also store z many blocks;

3. while(flag) {

(a) choose a random block, mark it as used and put it in node[count];

(b) for(i=1; i<z; i=i+1) {

i. search all the unused blocks in random fashion and put the first available on one in *node[count]* which has no common key with the existing blocks already in *node[count]*;

ii. mark this block as used;

iii. if such a block is not available then break the for loop and assign flag = false;

(c) } (end for)

(d) if flag = true then count = count + 1;

4. } (end while)

5. report that count many nodes are formed such that there is no intra node connectivity;

6. for rest of the (r^2 – count.z) many blocks, merge z blocks randomly to form a node (they may have intra-node connectivity) to get ($\left\lfloor \frac{r^2}{z} \right\rfloor - count$) many extra nodes. This constitutes the initial configuration;

7. calculate the adjacency matrix;

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

9. perform 1000 such iteration;

10. end;

The subroutine ***move*** has been defined as follows:

1. From the list of pair of nodes sharing maximum number of common keys, select one pair of nodes randomly. Call them a and b;

2. From the list of pair of nodes sharing no common key, select one pair of nodes randomly. Call them c and d;
3. Select one block each from a and b, and remove them such that the removed blocks intersect each other and a and b are still connected upon their removal. Let the removed blocks be α and β respectively.
4. Select one block each from c and d. Let the removed blocks be γ and δ respectively.
5. Put γ in a, δ in b, Put α in c, β in d.
6. Undo the above changes.

4.3 DETERMINISTIC APPROACH FOR MERGING OF BLOCKS

The above merging approach is heuristic, and not deterministic. The way blocks are chosen for merging are initially random with the condition that there should be no common key among the merged blocks. When one cannot proceed with merging with this condition of no inter-node connectivity, but still nodes are remaining for merging, the above said approach merges the remaining node without satisfying the inter-node connectivity. Then it applies mutation to the merged blocks to bring inter-node connectivity close to minimum.

To merge the blocks to form key-chain we first need to decide the merging factor, z. We should choose the merging factor in such a way that the increased load on memory of sensor node to store enlarged key-chain is not too high as well as the connectivity of the overall network improves considerably.

Recall that p is the prime power chosen for the transversal design. We observe that every consecutive p blocks, starting from block zero, don't have any common key among themselves. One can easily verify this observation in Table 1. Whenever we will be referring 'consecutive blocks of p blocks' or 'p consecutive blocks', we must remember that we have started with the first block only. For example if p=5, then we will have a transversal design with 25 blocks. So first group of consecutive blocks are block 0 – block 4, next group of consecutive blocks are block 5 – block 9, and so on. So in other words whenever we are referring the word 'consecutive blocks' or 'group of consecutive blocks', the condition is that block-id % p must be zero. If the block id is of

the form (x, y) where $0 \leq x, y \leq p-1$, then the condition is that y must be zero of a block for being the starting block of consecutive p blocks.

we merge consecutive blocks (merging factor z, $z \leq p$) taken from consecutive p blocks. but if $z<p$ then z%p blocks will be left out of merging consideration, and we get z/p (integer division) number of merged blocks. We have to consider different scenarios.

1. If p%z=0 (this happens when $p=x^y$ where x is a prime, and z=x) then we can simply take every consecutive group of z blocks and then merge them to form merged-blocks and assign to sensor nodes.

2. p%z=1. We know, among p consecutive blocks (if we start from the first block itself) there will be no common key. So we try to form p/z number of merged-blocks in such a way that remaining blocks, one from every group of p blocks will not have any common key. Say there are x such remaining blocks. In reality, this x is always equal to p. Then we can form x/z number of merged blocks with no common key. If we want to eliminate possibility of having common key or a node with less than z*k number of keys then we will stop here only, and do not try to use the remaining blocks.

3. p%z=2. In this case from every group of p consecutive blocks we will have two remaining blocks. So we first take two groups of p blocks from all the group of consecutive p blocks such that there is no inter-group common key. After this, we take the remaining p/z blocks from each group and merge them to form the p.(p/z) merged blocks. Among each of the two groups of p blocks which were taken out at first, we continue with merging any z number of blocks to form p/z merged-blocks. Although there is a possibility of still remaining more than z number of blocks in total, our observation suggests that they always have common key among them. That is, we can't proceed to have another z number of blocks with no common key. So we can either leave those blocks or to allow few merged-blocks with inter-block common key.

4. For p%z >2 we can proceed with the same logic as above but, our observation suggests that in that cases the number of nodes that will be remaining after taking all merged-blocks will be little bit higher. So either we have to compromise with few merged-blocks with inter-block common key, or just leave those blocks unused which may turn little bit disadvantageous.

4.3.1 Deterministic Merging Algorithm

Now we formally present the algorithm for merging of blocks:

1. Start.
2. If $p=x^y$ and p%z=0 (i.e., z=x) then
    a. For i=0;i<p;i++
        i. For j=0;j<p/z;j++
            1. Take any z number blocks from the i-th group og consecutive blocks and form a merged-block.

        [end of inner-for loop]

    [end of outer-for loop]

3. If p%z > 0 then
    a. For i=0;i<p%z;i++
        i. Take a counter j and initialize with any value between 0 and p; i.e., $0 \leq j \leq p-1$.
        ii. For k=0;k<p;k++
            1. Choose (j+1)-th block from the k-th consecutive blocks. If this (j+1)-th block is the last block in this group then in the next iteration we will select 0-th block from next group instead of (j+1)-th block.

        [end of inner-for loop]

    [end of outer-for loop]

    b. Out of these each group of p blocks, form p/z merged-blocks, with each merged-block is made following the next step.
    c. By taking any z blocks together and merging them to get a merged-block with no inter-block common key.
    d. for i=0;i<p;i++
        i. From each of the group of consecutive blocks choose any unused z blocks and merge them to form a merged-block with no inter-block common key.

[end of for loop]

e. To deal with remaining blocks that are still not used for merging, either compromise to have merged-blocks with common inter-block key, or simply scrap these blocks depending upon given options.

4. In case we decided to allow few merge-blocks with common key it may happen that at last there will x number be blocks left where x<z. Depending upon the options, here also we can choose to have either a merged-block consists of x number of blocks, or not to use these x number of blocks.
5. Assign each of these merged-blocks to a sensor node.
6. Stop.

Now let us illustrate the above example with an example. Suppose p=5, and z=3. Other parameters are as follows- N =25, k=4, as it was earlier.

Node ID	Key-chain	Node ID	Key-chain	Node ID	Key-chain
1	(0,0), (1,0), (2,0), (3,0)	2	(0,1), (1,1), (2,1), (3,1)	3	(0,2), (1,2), (2,2), (3,2)
4	(0,3), (1,3), (2,3), (3,3)	5	(0,4), (1,4), (2,4), (3,4)	6	(0,0), (1,1), (2,2), (3,3)
7	(0,1), (1,2), (2,3), (3,4)	8	(0,2), (1,3), (2,4), (3,0)	9	(0,3), (1,4), (2,0), (3,1)
10	(0,4), (1,0), (2,1), (3,2)	11	(0,0), (1,2), (2,4), (3,1)	12	(0,1), (1,3), (2,0), (3,2)
13	(0,2), (1,4), (2,1), (3,3)	14	(0,3), (1,0), (2,2), (3,4)	15	(0,4), (1,1), (2,3), (3,0)
16	(0,0), (1,3), (2,1), (3,4)	17	(0,1), (1,4), (2,2), (3,0)	18	(0,2), (1,0), (2,3), (3,1)
19	(0,3), (1,1), (2,4), (3,2)	20	(0,4), (1,2), (2,0), (3,3)	21	(0,0), (1,4), (2,3), (3,2)
22	(0,1), (1,0), (2,4), (3,3)	23	(0,2), (1,1), (2,0), (3,4)	24	(0,3), (1,2), (2,1), (3,0)

Table 4.1: A transversal design TD(4,5).

Here, p%z=5%3=2. First, we choose j=0, which satisfies the condition $0 \leq j \leq p-1$. So we choose the block 0, block 6, block 12, block 18, and block 24. This forms a group of 5 blocks (blocks 0, 6, 12, 18, and 24) with no inter-block common key. Next, we choose j=2. So in the process of forming a group of 5 blocks with no common key we first choose block 2, then block 8, next block 14. Now since 14 is the last block of the group consecutive 5 blocks starting at block 11,

we choose the next block as the first block in the next group which is block 15, and next we choose block 21. So we form another group of 5 blocks (blocks 2, 8, 14, 15 and 21) with no inter-block common key. From these two groups of blocks we choose any 3 blocks and make merged-blocks. Say, we take blocks {0, 6, 12} to form a merged-block, and then take blocks {2,8, 14} to form the next merged block. Next we form p merged-blocks as follows: First merged-block consists of blocks {1, 3, 4}, second merged-block consists of blocks {5, 7, 9}, third merged-block consists of {10, 11, 13}, fourth merged-block consists of {16, 17, 19}, and fifth merged-block consists of {20, 22, 23}. So we obtain total 2+5=7 merged blocks. Still we have 3remaining blocks – namely block 15, 18, 21, and 24. We see that we can merge any three of these to get a new merged-block but that would give us a merged-block with inter-block common key. So depending upon our constraints either we consider this merged-block with inter-block common key, or we leave these four blocks as unusable. Note that even if we consider this merged-block, say blocks {15, 18, 21}, we will still have one block left unused namely block 24. We can assign this single block to a sensor node but then it will have small common key-chain.

4.4 KEY ESTABLISHMENT IN TRANSVERSAL DESIGN BASED KPS

As we know three important part of any key predistribution scheme are *key predistribution*, *key establishment*, and *path-key establishment*. Till now our focus was mainly confined on key predistribution phase. If any a node wants to communicate with a node, say in its neighbor, then they must first exchange some messages to determine if they both share a common key. The advantage of using combinatorial design is that many times we can find a pattern among the key-chains and the block-id. This advantage of using combinatorial design is exploited to obtain the common key between a pair of nodes without comparing their individual keys.

4.4.1 A Deterministic Key Establishment Approach for Transversal Design based KPS

We first illustrate our proposal by an example. In transversal design the each block consists of keys (i.e., treatments or varieties) of components. If we consider the whole design as a two-dimensional object, we can see that first component of any key in column i is of the form (i, x) where $0 \leq x \leq k-1$. Let us again take the same example of previous section.

Node ID	Key-chain	Node ID	Key-chain	Node ID	Key-chain
1	(0,0), (1,0), (2,0), (3,0)	2	(0,1), (1,1), (2,1), (3,1)	3	(0,2), (1,2), (2,2), (3,2)
4	(0,3), (1,3), (2,3), (3,3)	5	(0,4), (1,4), (2,4), (3,4)	6	(0,0), (1,1), (2,2), (3,3)
7	(0,1), (1,2), (2,3), (3,4)	8	(0,2), (1,3), (2,4), (3,0)	9	(0,3), (1,4), (2,0), (3,1)
10	(0,4), (1,0), (2,1), (3,2)	11	(0,0), (1,2), (2,4), (3,1)	12	(0,1), (1,3), (2,0), (3,2)
13	(0,2), (1,4), (2,1), (3,3)	14	(0,3), (1,0), (2,2), (3,4)	15	(0,4), (1,1), (2,3), (3,0)
16	(0,0), (1,3), (2,1), (3,4)	17	(0,1), (1,4), (2,2), (3,0)	18	(0,2), (1,0), (2,3), (3,1)
19	(0,3), (1,1), (2,4), (3,2)	20	(0,4), (1,2), (2,0), (3,3)	21	(0,0), (1,4), (2,3), (3,2)
22	(0,1), (1,0), (2,4), (3,3)	23	(0,2), (1,1), (2,0), (3,4)	24	(0,3), (1,2), (2,1), (3,0)

Table 4.2 A transversal design TD(4,5).

We see that the second component of first column always repeats after a cycle of four digits - 0 to 4. In other words we can restate it as follows. The distance between two second component elements in the i^{th} ($0 \leq i \leq 4$) blocks in every group of p consecutive blocks is zero. In the second column, this distance increases by one. In this example, the first block in first group (i.e., block 0) contains the element 0 as the second component of second column, then the corresponding element in first block of second group (i.e., block 5) is 1, and in the first block of third group (i.e., block 10) is 2. In the third column, this distance increases by two. And this pattern holds true for all the columns. The algorithm for key establishment is given in the below. *To make the algorithm easy to understand we have presented the algorithm for KPS with block size four, which can be generalized easily* with little bit of extra effort.

1. Input: key-id, KEY_ID of a sensor node
2. if (key-id is one-dimensional)
    a. Take an integer variable x, and initialize x=KEY_ID
    b. Take another integer variable y, and initialize y=x%p
    c. Take another integer variable z, and initialize z=x/p
    d. a=y
    e. initialize b=0
    f. for(j=1;j≤z;j++)
        i. b=b+1

        [end of for loop]

g. b=b%p

h. for(j=1;j≤y;j++)

    i. b=b+1

    [end of for loop]

i. b=b%p

j. initialize c=0

k. for(j=1;j≤z;j++)

    i. c=c+2

    [end of for loop]

l. c=c%p

m. for(j=1;j≤y;j++)

    i. c=c+1

    [end of for loop]

n. c=c%p

o. initialize d=0

p. for(j=1;j≤z;j++)

    i. d=d+3

    [end of for loop]

q. d=d%p;

r. for(j=1;j≤y;j++)

    i. d=d+1

    [end of for loop]

s. d=d%p

t. so the keys of node with id KEY_ID are (0,a), (1,b),(2,c),(3,d)

[end of if]

3. if (KEY_ID two-dimensional of the form (w,x))

a. y=x
b. z=w
c. go to step 2 d.

[end of if]

4. stop

So to establish the common key with other node y, a node x should know the id of node y. Given that, node x can compute all the keys of x by itself and then compare with its own to determine if x has any common key with y. There exists method that directly computes the common key, if that exists, by using multiplicative inverse operation [17]. Our method is lengthy, but does not use multiplicative inverse operation.

5 CONCLUSION

Designs provide balanced set systems. This balance leads to some good algorithmic properties. One consequence is that it helps us to get efficient description of key establishment algorithms when designs are used for key predistribution. Key predistribution using combinatorial designs are an area of active research.

We have studied transversal design based key predistribution scheme and orthogonal array based key predistribution scheme in detail. The low resiliency of the orthogonal array based key predistribution as observed by us suggests that one can further look for different orthogonal array construction schemes, other than Bush's construction method, for an improved key predistribution scheme based on orthogonal array. We found that one would not benefit by merging blocks of orthogonal array to get a modified key predistribution in terms of resiliency and connectivity due to low key pool size and the high connectivity of orthogonal array based key predistribution scheme.

A deterministic algorithm for merging of blocks in transversal design has been proposed for obtaining a modified key predistribution scheme. The existing work on this was not deterministic, but heuristic one. Also, we have proposed deterministic a key establishment algorithm for transversal design which does not use any computationally intensive operation such as multiplicative inverse as used by some existing algorithms.

6 BIBLIOGRAPHY

[1] Ian F. Akyildiz, Weilian Su, Yogesh S., and Erdal Cayirci, A Survey on Sensor Networks, IEEE Communications Magazine, Vol. 40, No. 8, pp. 102-114, August 2002.

[2] R. Blom. An Optimal Class of Symmetric Key Generation Systems. Proceedings of Eurocrypt 1984, LNCS 209, pp. 335-338, 1985.

[3] C. Blundo, A. Santis, A. Herzberg, S. Kutten, U. Vacarro, and M. Yung. Perfectly- Secure Key Distribution for Dynamic Conferences. Proceedings of Crypto 1992. LNCS 740, pp. 471-486, 1993.

[4] Edger Callaway. Wireless Sensor Networks Architectures and Protocols. Auerabach Publications. 2003.

[5] S. A. Camtepe, B. Yener, Combinatorial Design of Key Distribution Mechanisms for Wireless Sensor Networks, IEEE/ACM Transactions on Networking, Vol. 15, No. 2, pp 346-358, April 2007.

[6] S. A. Camtepe, B. Yener, Key Distribution Mechanisms for Wireless Sensor Netowork: a Survey, Rensselaer Polytechnic Institute Technical Report TR-05-07. March 2005.

[7] David W. Carman. New directions in Sensor Network Key Management, International Journal of Distributed Sensor Networks, Vol. 1, pp 3-15, 2005.

[8] E. Cayirci, C. Rong. Security in Wireless Ad Hoc and Sensor Networks. John Wiley & Sons. 2008.

[9] D Chakrabarti, S Maitra, B Roy. A key pre-distribution scheme for wireless sensor networks: merging blocks in combinatorial design. International Journal of Information Security, Vol. 5, No. 2, pp 105-114, 2006.

[10] C. J. Colbourn & P. C. Van Oorschot, Applications of Combinatorial Design in Computer Science, ACM Computing Surveys, Vol. 21, No. 2, June 1989.

[11] C. J. Colbourn, J. H. Dinitz, D. R. Stinson, Application of Combinatorial Design in Communications, Cryptography, and Networking. London Mathematical Society Lecture Note Series 187, Cambridge University Press. 1999.

[12] Junwu Dong, Dingyi Pei, Xueli Wang, A Class of Key Predistribution Based on Orthogonal Arrays, Journal of Computer Science and Technology. Vol.23 No. 5, pp 825-831, September 2008.

[13] Junwu Dong, Dingyi Pei, Xueli Wang, A Class of Key Predistribution Based on 3-Designs, Inscrypt 2007, LNCS 4990, pp.81-92, 2008.

[14] Laurent Eschenauer, Virgil D. Gligor, A Key-Management Scheme for Distributed Sensor Networks, Proceedings of the 9th ACM conference on Computer and communications security. pp 41 – 47, 2002.

[15] Mohammad Illyas, Imad Mahgoub. Handbook of Sensor Network: Compact Wireless and Wired Sensing Systems. CRC Press, 2004.

[16] J. Lopez (Editor), J. Zhou (Editor). Wireless Sensor Networks Security. IOS Press. 2007.

[17] J. Lee, D. R. Stinson, A Combinatorial Approach to Key Predistribution for Distributed Sensor Networks. IEEE Wireless and Communications and Networking Conference. 2005.

[18] J. Lee, D. R. Stinson. On the Construction of Practical Key Predistribution Schemes for Distributed Sensor Networks Using Combinatorial Designs. ACM Transactions on Information and System Security (TISSEC). Volume 11 , No. 2, pp 1-35, March 2008.

[19] K. M. Martin. On the Applicability of Combinatorial Designs to key predistribution for wireless sensor networks. arXiv e-print Archive Report arXiv:0902.0458v1 [cs.CR]. 2009.

[20] Anupam Pattanayak, B. Majhi. Weakness of Key Predistribution Scheme Proposed by J. Dong et al. Cryptology eprint Archive. Report 2009/114. 2009.

[21] Anupam Pattanayak, B. Majhi. Key Predistribution Schemes in Distributed Wireless Sensor Network using Combinatorial Designs Revisited. Cryptology eprint Archive. Report 2009/131. 2009.

[22] Anupam Pattanayak, B. Majhi. Weakness A Deterministic Approach of Merging of Blocks in Transversal Design based Key Predistribution. Cryptology eprint Archive. Report 2009/156. 2009.

[23] Adrian Perrig, J. Stankovic, D. Wagner, C. Rosenblatt. Communications of the ACM. Vol. 47, pp 53-57, 2004.

[24] Shashi Phoha (Editor), Thomas F. La Porta (Editor), Christopher Griffin (Editor). Sensor Network Operations. Wiley–IEEE Press. 2006.

[25] C. S. Raghavendra (Editor), Krishna M. Sivalingam (Editor), Taeib F. Znati (Editor). Wireless Sensor Network. Springer. 2006.

[26] Praveen Rentala, Ravi Musunuri, Shashidhar Gandham, Udit Saxena, Survey On Sensor Networks, Proceedings of International Conference on Mobile Computing and Networking, 2001.

[27] Sushmita Ruj, Bimal Roy, Key Predistribution using PBIBD in Wireless Sensor Network, ISPA 2007, LNCS 4742, pp. 431-445, 2007.

[28] Sushmita Ruj, Bimal Roy, Key Establishment Algorithms for some Deterministic Key Predistribution Schemes, WOSIS 2008: 68-77

[29] R. A. Shaikh, S. Lee, Y. J. Song, Y. Zhung. Securing Distributed Wireless Sensor Networks: Issues and Guidelines. IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing. Vol. 2, pp 226-231, 2006.

[30] D. R. Stinson, Combinatorial Designs: Constructions and Analysis. Springer-Verlag. New York. 2004.

[31] Street A. P., Street D. J, Combinatorics of Experimental Design, Clarendon Press, Oxford 1987.

[32] Marcos A. M. Vieira, Claudinor N. Coelho. Jr., DiÓgenes C. Da Silva Jr., José M. da Mata, Survey on Wireless Sensor Network Devices, IEEE Conference Emerging Technologies and Factory Automation, 2003.

[33] Yang Yu, V. K. Prasanna, B. Krishnamachari. Information processing and routing in Wireless Sensor Network. World Scientific Publishing. 2006.

[34] Feng Zhao, Leonidas Guibas.Wireless Sensor Networks: An Introduction Processing Approach. Morgan Kauffman. 2004.

[34] Yingshu Li, My T. Thai, Weili Wu. Wireless Sensor Networks and Applications. Springer, 2008

[35] Feng Zhao, Leonidas Guibas.Wireless Sensor Networks: An Introduction Processing Approach. Morgan Kauffman. 2004.

DISSEMINATION OF THESIS WORK

TECHNICAL REPORTS:

1. Anupam Pattanayak, B. Majhi. Weakness of Key Predistribution Scheme Proposed by J. Dong et al. Cryptology eprint Archive. Report 2009/114. 2009.
2. Anupam Pattanayak, B. Majhi. Key Predistribution Schemes in Distributed Wireless Sensor Network using Combinatorial Designs Revisited. Cryptology eprint Archive. Report 2009/131. 2009.
3. Anupam Pattanayak, B. Majhi. Weakness A Deterministic Approach of Merging of Blocks in Transversal Design based Key Predistribution. Cryptology eprint Archive. Report 2009/156. 2009.

COMMUNICATED PAPERS:

1. Anupam Pattanayak, B. Majhi. A Deterministic Approach of Merging of Blocks in Transversal Design based Key Predistribution. Submitted to IEEE Globecom 2009 Communications and Information Security Symposium, to be held in Hawai, USA.
2. Anupam Pattanayak, Banshidhar Majhi. On Resiliency of Orthogonal Array based Key Predistribution. Submitted to National Workshop on Cryptology 2009, to be held in SV NIT Surat.

Following two papers were *accepted* in WorldComp International Conference on Security and Management 2009, to be held in Las Vegas, USA.

1. Anupam Pattanayak, B. Majhi. On Resiliency of a Particular Key Predistribution Scheme.
2. Anupam Pattanayak, B. Majhi. Key Predistribution Schemes in Distributed Wireless Sensor Network Using Combinatorial Designs Revisited.