Secure Protocols for Key Pre-distribution, Network Discovery, and Aggregation in Wireless Sensor Networks

by

Kevin J. Henry

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2015

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

The content presented in Chapter 3 was co-authored with Maura Paterson. All chapters of this thesis were authored under the supervision of Douglas Stinson.

Abstract

The term sensor network is used to refer to a broad class of networks where several small devices, called sensors, are deployed in order to gather data and report back to one or more base stations. Traditionally, sensors are assumed to be small, low-cost, battery-powered, wireless, computationally constrained, and memory constrained devices equipped with some sort of specialized sensing equipment. In many settings, these sensors must be resilient to individual node failure and malicious attacks by an adversary, despite their constrained nature.

This thesis is concerned with security during all phases of a sensor network's lifetime: pre-deployment, deployment, operation, and maintenance. This is accomplished by pre-loading nodes with symmetric keys according to a new family of combinatorial key pre-distribution schemes to facilitate secure communication between nodes using minimal storage overhead, and without requiring expensive public-key operations. This key pre-distribution technique is then utilized to construct a secure network discovery protocol, which allows a node to correctly learn the local network topology, even in the presence of active malicious nodes. Finally, a family of secure aggregation protocols are presented that allow for data to be efficiently collected from the entire network at a much lower cost than collecting readings individually, even if an active adversary is present.

The key pre-distribution schemes are built from a family of combinatorial designs that allow for a concise mathematical analysis of their performance, but unlike previous approaches, do not suffer from strict constraints on the network size or number of keys per node. The network discovery protocol is focused on providing nodes with an accurate view of the complete topology so that multiple node-disjoint paths can be established to a destination, even if an adversary is present at the time of deployment. This property allows for the use of many existing multi-path protocols that rely on the existence of such node-disjoint paths. The aggregation protocols are the first designed for simple linear networks, but generalize naturally to other classes of networks. Proofs of security are provided for all protocols.

Acknowledgements

I would like to thank all members, past and present, of the Cryptography, Security, and Privacy (CrySP) research lab for inspiring and shaping my work over the years.

Dedication

To my parents, who have never failed to offer their support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction to Sensor Network Research

The term *sensor network* is used to refer to a broad class of networks where several small devices, called *sensors*, are deployed in order to gather data and report it back to one or more *base stations*. Traditionally, sensors are assumed to be small, low-cost, battery-powered, wireless, computationally constrained, and memory constrained devices equipped with some sort of specialized sensing equipment. These descriptors are not precise, which makes formalizing a problem statement using sensors difficult in most cases. In practice, the term sensor can be applied to any communication-enabled sensing device whose processing power lies somewhere in the domain between a smart card and a cell phone. A sensor network is a collection of such sensors, ad-hoc or otherwise, which collectively coordinate their actions in order to sense data about their environment and forward it to a central location.

This thesis is concerned with a subset of the sensor network problem domain where nodes have extremely limited computational and memory constraints. In particular, we present a suite of protocols that allow resource-constrained sensor nodes to operate during all phases of a network's lifetime in the presence of an active malicious adversary. The severe limitations of sensors in this setting make many common cryptographic techniques, such as public-key cryptography, too expensive to consider. In general, we rely on symmetric-key cryptography and design protocols from the ground up, beginning with a minimal set of assumptions, to identify precisely what capabilities are necessary in order provide a provably secure solution. We provide complete proofs of security and simulation results where applicable.

The protocols presented in this thesis span the main phases of the lifetime of a sensor network. First, we present a flexible variant of an existing key pre-distribution scheme that allows for combinatorial key pre-distribution to be used in networks of arbitrary size, rather than just networks where the number of nodes is a prime or prime power. We then demonstrate how this scheme can be used to perform secure network discovery in the presence of an active malicious adversary, by using a consensus-based voting protocol.

The main operational goal of a sensor network is to collect and report data to a base station. We present a suite of aggregation protocols that makes this process more efficient, and ensures that damaged or compromised nodes cannot maliciously alter the network-wide aggregate without detection. These protocols are based on a family of linear networks; however, we demonstrate that such networks arise naturally in many common sensor network settings. The generalization of our approach can be applied to many other topologies while incurring only small overhead at each point two or more paths merge in a spanning tree.

1.1 Introduction to Sensor Networks

The most general interpretation of the term “sensor network” refers to any network in which low-cost, low-power nodes are deployed to collect and forward sensor readings, and as such, they encompass a wide variety of node capabilities and applications. Definitions in the literature are often vague, as there is no standard accepted definition of a sensor network. For example, according to Jawhar and Mohamed [53],

> “a sensor network can be described as a collection of sensor nodes which coordinate to perform some specific action.”

and according to Martin and Paterson [74],

> “There is no single, precise, definition of a wireless sensor network. As a result this term is applied to a wide family of networking environments that support a range of applications.”

The lack of a precise definition makes sensor networks a wide open space of research, with many different assumptions about the individual capabilities of sensor nodes and the settings where they will be deployed. One common platform for sensor network development is TinyOS [63], an operating system built specifically for sensor networks. TinyOS is often used in conjunction with Mica sensor hardware. Figure 1.1 shows a typical Mica2 sensor node alongside a Mica2Dot node, with the following specifications:

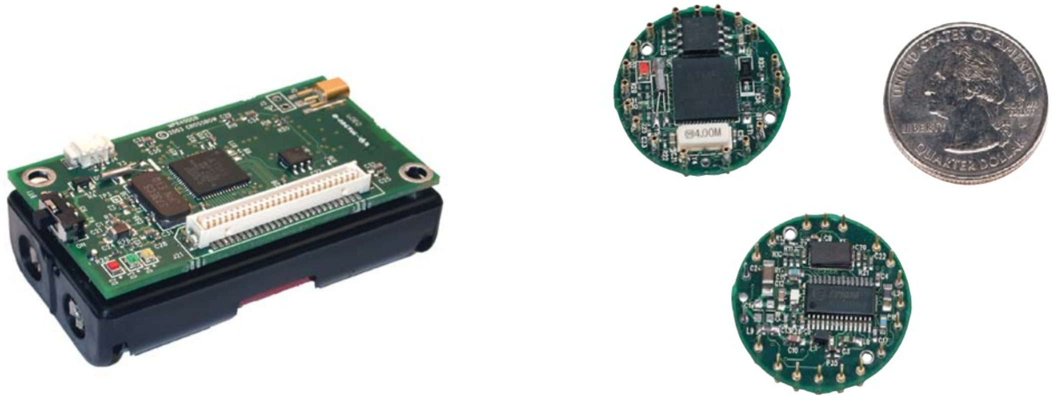


Figure 1.1: The Mica2 and Mica2Dot sensor nodes. The full sized Mica2 node is powered by 2 AA batteries, while the Mica2Dot is powered by a 1 inch coin cell. Images taken from the Mica2 [29] and Mica2Dot [30] data sheets

- Processor: ATmega 128L (8-bit, 16 MHz)
- Program Flash Memory: 128K bytes
- Measurement Flash Memory: 512K bytes
- Communication Range: 500–1000m
- Size: 58 x 32 x 7mm (Mica2), 25 x 6mm (Mica2Dot)
- Weight: 18g (Mica2), 3g (Mica2Dot)
- Power: 2x AA batteries (Mica2), 3V coin cell (Mica2Dot)

The Mica sensor platform was first released in 2002, and provided researchers with a low-cost, standardized platform upon which sensor network protocols could be implemented and tested. Mica nodes are physically large, allowing them to be powered with off-the-shelf hardware, and to be equipped with standard connectors for attaching different sensing equipment. In the decade following the release of TinyOS and the Mica platform, several other similar nodes have been released.

A different branch of hardware-related sensor research has been pushing the limits on the physical size of sensor nodes, typically at the expense of communication range, memory,

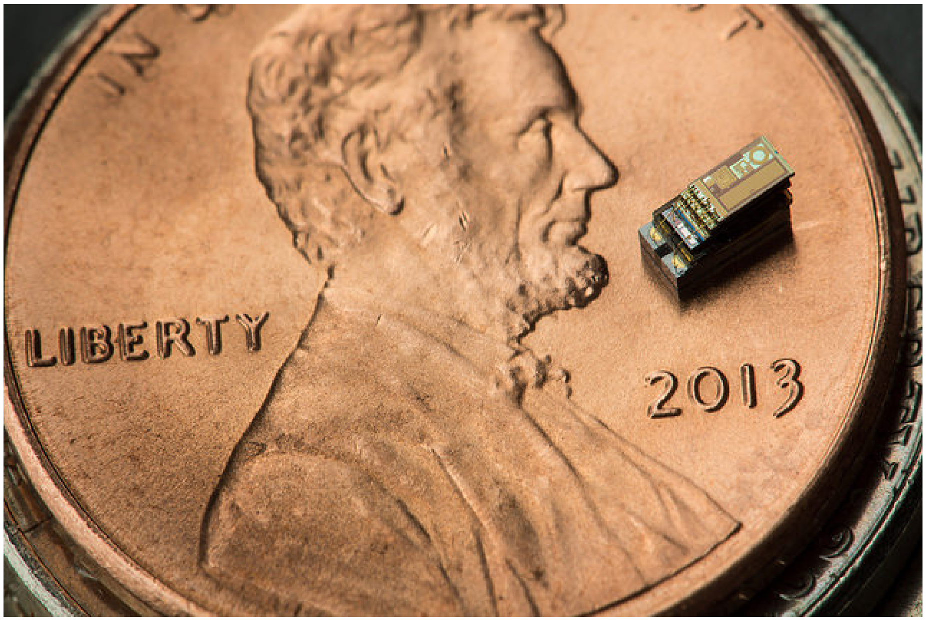


Figure 1.2: An example of a millimeter-scale smart dust node, based on the node presented by Lee et al. [62]. Photograph by Martin Vloet, Michigan Photography [102].

and computational ability. The goal of such work is to make the nodes as small and cheap as possible, while still retaining a useful amount of functionality. For example, the node presented by Warneke et al. [107], around the same time the Mica platform was released, has a total volume of 16 mm^3, and the node presented a decade later by Lee et al. [62] is less than 2 mm^3 in volume. Nodes on this scale are sometimes referred to as *smart dust*.

The contrast between Mica-like nodes and smart dust nodes exemplify the wide scope of sensor network research. Solutions for one class of node may make assumptions that are not relevant for a different class of nodes. In particular, the trend of reducing the size and cost of individual nodes in smart dust networks, potentially at the expense of speed and storage, suggests that there will always be a demand for protocols that make a minimal set of assumptions about the capability of each node.

1.1.1 Example Applications

Akyildiz et al. [2] provide a survey of potential sensor network applications, partially summarizing earlier proposals [38]. They identify several types of measurements that a sensor node may be deployed to measure. These include:

- Temperature
- Humidity
- Vehicular movement
- Light levels
- Pressure
- Soil conditions
- Noise levels
- The presence or absence of certain objects
- Mechanical stresses
- Motion characteristics (velocity, orientation, etc.).

In addition to specific sensor types, they also enumerate many deployment environments where sensor networks could be used. These include:

1. Military applications
   - Battlefield surveillance
   - Reconnaissance of opposing forces and terrain
   - Targeting
   - Damage assessment
   - Nuclear, biological, and chemical attack detection and reconnaissance.
2. Environmental applications
   - Forest fire detection
   - Biocomplexity mapping
   - Flood detections
   - Precision agriculture.

3. Health applications
    - Physiological data monitoring
    - Patient tracking/monitoring
    - Drug administration.
4. Home applications
    - Home automation
    - Smart environment.
5. Other applications
    - Climate control in office buildings
    - Vehicle monitoring
    - Inventory control.

Alongside their survey of applications, Akyildiz et al. also survey pre-deployment, deployment, post-deployment, and environmental issues that affect sensors throughout their lifetime. These phases roughly match the same phases considered in this thesis, which are discussed in detail in the next section. Yick et al. [112] provide another survey that identifies many of the same deployment scenarios, but they consider the requirements of each deployment scenario in more detail, and relate them to existing sensor network technologies that have emerged since their initial exploration.

Commercial hardware is now available that addresses most of the sensing applications discussed above. A notable example is the Libelium Waspmote [66], which provides an updated Mica-like platform with several types of swappable sensors. These include:

- Gas sensors
- Event sensors (pressure, impact, vibration, etc.)
- Water sensors (pH, temperature, dissolved gases, etc.)
- City monitoring (noise, dust, luminosity, humidity, etc.)
- Parking monitor (car detection)
- Agriculture (moisture, UV, temperature, plant characteristics, etc.)

- Video (cameras, IR, etc.)
- Radiation (Geiger counter)
- Smart meter (water, power, etc.)

Each of these sensors sits on top of a node that is of similar size to the Mica2 node in Figure 1.1, which has a physical footprint slightly larger than two AA batteries.

1.2 Security Issues in Wireless Sensor Networks

Sensor networks have been proposed for a variety of applications. In many of these applications, an active adversary or a global eavesdropper may be present, but even in the general case, there are several security considerations to be made. Sensor networks may be deployed in an unknown or hostile environment and are subject to physical damage upon deployment, or during operation, as a result. Nodes are battery-operated and expected to fail over time. Sensing equipment, or the nodes themselves, may also malfunction and behave in an unpredictable manner.

A common example that demonstrates several key security issues is the use of sensors to survey hostile or adversary-controlled territory. An airplane is loaded with several sensor nodes and flies over the area to be surveyed, dropping sensors as it goes (and potentially damaging some of them). The goal of these sensors is to self-organize and establish a communication network, collect data about their surroundings, and forward it back to a base station. The adversary wishes to prevent data collection, and is expected to capture and disable nodes, re-program nodes, insert fake nodes, and alter or block communication between nodes. Even if nodes can be made tamper-proof, the adversary may have control over the environment and can feed false information to nodes. Durisic et al. [36] provide a survey of military applications of wireless sensor networks.

Shi and Perrig [98], and Wang et al. [105] identify the following general security requirements:

- Availability — Ability to remain accessible in the presence of DoS attacks.
- Authorization — Ability to prevent unauthorized nodes from joining the network.
- Authentication — Ability to verify that a given message originated from a given node.

- Confidentiality — Ability to ensure that only the intended recipients can read messages.

- Integrity — Ability to ensure that messages have not been altered during transit.

- Fault Tolerance — Ability to continue operation in the presence of failure.

- Self Healing — Ability to recover from and adapt to changes in the network.

- Non-Repudiation — Ability to prevent a node from denying it sent a signed message.

- Repudiation — Ability to deny having sent a signed message.

- Freshness — Ability to detect if old messages are replayed.

- Reliability — Ability to guarantee a message is delivered.

- Accountability — Ability to detect and punish malicious or malfunctioning nodes.

- Forward Secrecy — Ability to prevent a node from reading messages once it is removed from the network.

- Backward Secrecy — Ability to prevent a new node that joins the network from reading any previous messages.

- Scalability — Ability to accommodate both very small and very large networks.

- Flexibility — Ability to accommodate different deployment methods and environments.

- Resiliency — Ability to continue operation in the presence of compromised nodes.

Not all applications will require all security considerations. In general, these security requirements are not specific to sensor networks, and apply to virtually any networking application.

Karlof and Wagner [54] provide an intuitive approach to sensor network security for routing protocols by considering network assumptions, trust requirements, threat models, and security goals before considering specific routing attacks in detail. The attacks considered are:

- Message tampering and message replay — Maliciously altering routing information.

- Selective forwarding/dropping — Selectively dropping some messages but not others.

- Sinkhole attacks — Attempting to route as much traffic through a single malicious node as possible.

- Sybil attack — One node pretending to be multiple nodes.

- Wormhole attacks — Using a high-power transmitter to connect one or more malicious nodes, thus creating more attractive but adversarial-controlled links.

- `HELLO` flood — Spoofing or re-broadcasting `HELLO` messages to falsely convince nodes they are within communication range.

- Acknowledgment spoofing — Spoofing of link-level protocol communication messages to falsely convince a node its messages were or were not delivered.

In general, attacks against sensor networks can also be categorized with respect to the Open Systems Interconnection (OSI) model, with attacks occurring at one of the physical, data link, network, transport, or application layers.

Karlof and Wagner's threat model distinguishes between a *node-class attacker*, which is subject to the same constraints as a regular sensor node, and a *laptop-class attacker*, which has capabilities similar to a base station. Distinctions are also made between *insider attacks*, performed by a malfunctioning or compromised node, and *outsider attacks*, performed by an attacker adding unauthorized nodes to the network. As a final consideration, both *active* and *passive* adversaries are considered, with the former able to actively inject or alter messages into the network. Active adversaries are often considered to be *local*, able to communicate with only a small portion of the network, while passive adversaries are often considered to be *global*, and are able to eavesdrop on all messages sent over the network. In many applications, the base station is considered to be tamper-proof and is treated as a trusted third party. Cardenas et al. [17] provide a similar taxonomy of sensor network threat models based on a study of Supervisory Control and Data Acquisition (SCADA) networks.

When considering security issues, it is useful to approach sensor network operation as occurring in a set of discrete phases which can be analyzed independently. The lifetime of a sensor network can be broken down into four distinct phases:

1. Pre-deployment

2. Deployment/Setup

3. Operation

4. Management/Maintenance.

Each of these phases has its own set of security considerations, which will be discussed briefly in the next section. These phases are similar to the phases discussed by Carmen et al. [18]. In their model, the lifetime of a sensor network was broken into manufacture, storage, pre-deployment, deployment, mission, and mission completion. These parallel the four phases presented above, with the addition of the physical security of the nodes prior to deployment. Physical security during manufacture and storage is beyond the scope of this thesis.

The protocols presented in this thesis each use slightly different adversarial models, which are presented alongside each protocol in the subsequent chapters. In all cases we assume an adversary compromises and assumes control over a subset of nodes in the network. The key pre-distribution scheme in Chapter 3 is analyzed with respect to random node compromise across the entire network. The later chapters require constraints on the number of compromised nodes within certain subsets of the network. In some cases, a majority of honest nodes are required locally to prevent malicious nodes from injecting false information. In the case of linear topologies, considered in Chapter 5, we require that honest nodes occur with sufficient frequency that the adversary cannot segment the network by taking control of sufficiently many consecutive nodes.

1.2.1 Pre-deployment

This phase is concerned with all aspects of sensor node security before the network is deployed.

Node Capability

The physical characteristics of each node must be established before any security considerations can be made. In particular, the basic capabilities, such as memory, CPU, and battery power all determine the types of cryptographic techniques that can be utilized. While symmetric-key cryptography is often used in sensor networks, much research has been done to make public-key cryptography practical for resource-constrained nodes. Relevant works include the first elliptic curve cryptography (ECC) implementation for sensor networks [70], RSA and additional ECC implementations [104], and the TinyPBC [80] and TinyECC [68] libraries.

The communication medium in sensor networks is also an important consideration. Although most sensor network protocols assume omnidirectional wireless broadcasts as the communication medium, many [6,13,58,79] have begun investigating the use of directional antennas in sensor networks. In particular, Ash and Potter [6] and Kumar and Varma [58], utilize directional antennas in order to perform *localization*, which allows a sensor node to determine its approximate physical location after deployment. The communication medium also determines whether or not known methods for distance bounding [14,92], or fingerprinting [31,91], are available.

Additionally, special hardware, such as a GPS receiver or synchronized clocks, may be available in some settings.

Key Management

Because symmetric-key cryptography is usually preferred in sensor networks, the problem of pre-distributing keys to each sensor node is an important area of research. Eschenauer and Gligor [37] proposed a simple key pre-distribution scheme where each node is issued a random subset of keys from a larger key pool. This idea has been extended by others [24] as well. More recently, combinatorial designs have been utilized for key pre-distribution to increase efficiency or add additional useful properties. Examples include early proposals by Çamtepe and Yener [21], who utilize balanced incomplete block designs (BIBDs) and generalized quadrangles to manage keys. Martin explores the design space of sensor networks [73] and draws parallels between several design requirements and existing designs. Combinatorial approaches have also been used in specialized settings, such as for grid topologies [9], and settings where nodes are deployed in separate groups [76]. Transversal design-based systems [61] are used extensively in this thesis.

In systems where public-key cryptography is utilized, the key management step would also include issuing keys and certificates to each node before deployment.

1.2.2 Deployment/Setup

The deployment phase is concerned with the physical characteristics of the network after it is deployed, along with protocols that run once, such as network discovery, when the network comes online.

Topology

Because sensor networks are used in a variety of applications, there are a variety of topologies that arise naturally depending on how nodes are deployed and how keys are issued. More efficient protocols are possible when additional assumptions can be made about the structure of the network.

In many applications, nodes are assumed to be distributed completely at random, thus giving rise to a *random* topology. A common assumption is that nodes are distributed by airplane over a region. In this setting, nodes are still considered to be randomly distributed, however additional assumptions arise due to the nature of deployment. In particular, a node dropped from a specific point will likely land within an expected radius, and nodes dropped shortly before or after will have a higher probability of being nearby than nodes dropped much earlier or later. Such networks have a *partially random* topology.

Sensor networks that contain nodes of differing capabilities often give rise to a natural hierarchical topology. Nodes with greater communication range, computational ability, and larger batteries can aid in routing or aggregating messages for lower-power nodes in their neighborhood. In settings where there is complete control over the deployment location of nodes, the most natural topology is often a *grid*, as it provides a high level of coverage using a small amount of nodes. Some protocols have been designed specifically for grid-based sensor networks [9]; however, in general the area is not well studied.

Martin and Paterson [74] observe that the network topology has a tremendous impact on the design of sensor network protocols, and provide a simple, but comprehensive framework for categorizing networks based on the expected topology and key requirements. Their framework considers three categories:

1. **Homogeneity**
   - Homogeneous — All nodes have identical capability.
   - Hierarchical — Nodes have a natural hierarchy based on capability.
2. **Control Over Deployment/Topology**
   - Fixed, no control — Sensors are distributed randomly.
   - Fixed, partial control — Sensors are distributed with some control over general location, but with some randomness on exact location. For example, nodes distributed from a plane will likely land within an expected area and near other nodes dropped at the same time.

- Fixed, full control — Network topology is predefined.
- Locally mobile — A node is mobile within a well-defined subset of the network.
- Fully mobile — Nodes are mobile throughout the entire network.

3. **Communication structure**

   - t-complete — All subsets of size t should be capable of secure communication.
   - locally t-complete — All local subsets of size t should be capable of secure communication, where "local" refers to nodes that are neighbors in some sense, such as those that are within each other's communication range.
   - regionally t-complete — All subsets of size t in a given geographic region should be capable of secure communication.

The distinction between locally complete and regionally complete can be used, for example, to distinguish between cases where nodes deployed in a fixed topology should share keys with other nearby nodes in their communication range, while mobile nodes should share keys with other nodes deployed in the same general region, even if this region is significantly larger than any node's communication range. In the context of the above framework, the protocols presented in this thesis are mainly designed for homogeneous, fixed networks, with regionally or locally t-complete communication structure.

Jawhar and Mohamad [53] have observed that sensor networks have found several applications in settings where the network topology is inherently linear. For example, monitoring of pipelines, railways, subway tunnels, power lines, and border control have all been proposed using sensor networks. Very little work exists in the literature that specifically targets linear topologies. This thesis presents a new family of protocols designed for linear topologies, which generalizes naturally to grid- and tree-based topologies.

Network/Neighbor Authentication

Before messages can be routed in a sensor network, each node must determine its neighbors and the local network topology. This process is the first step for routing protocols, such as ARAN [97], S-AODV [114], and DV-SRP [83]. Poturalski, et al. [88] study the problem of neighbor discovery in sensor networks and demonstrated that approaches that rely on either message transmission time or distance between nodes cannot be secure in many settings; however, protocols that rely on both transmission time and location can be secure. Their model assumes nodes cannot collaborate with one another to determine the network topology. Distance alone is not sufficient, as an adversary utilizing a wormhole attack, where a message is recorded in one location of the network, and then replayed at a different point in the network, can falsely convince two nodes they are close together. If an adversary is able to quickly forward a message, or relay messages between two nodes it controls at a high enough speed, then time alone is not sufficient in some cases. This thesis presents a collaborative approach to network discovery in the presence of active malicious nodes.

Key Establishment

Cryptographic keys allow nodes to encrypt messages or compute message authentication codes (MACs), thus providing confidentiality against a passive adversary, and authenticity and integrity against an active adversary. Even when keys are pre-distributed to nodes, it is possible that many pairs of nodes wishing to communicate do not have a key in common, or may wish to establish a new unique pairwise key. In these situations, common neighbors between the two nodes can be used in order to establish a new shared key known only to these two nodes. Such approaches include those of Chan, Perrig, and Song [24], Ling and Znati [67] and Wu and Stinson [108], each of which utilize multiple node-disjoint paths in order to securely establish a key between two nodes. The key establishment problem is not considered in this thesis; however, the existence of multi-path protocols for key establishment provides motivation for the network discovery protocol presented in Chapter 4.

Localization

Some sensor network applications require that sensors are able to estimate their physical location in the network. This ability allows sensor readings to be tied to a physical location, as well as allowing the detection of certain types of attackers within the network. Localization protocols allow a node to determine either its absolute position, or its position relative to other nodes in the network. Ash and Potter [6], and Kumar and Varma [58] provide two different approaches utilizing wireless antennas to perform localization. Cheng et al. [27] provide a comprehensive survey on localization in sensor networks. An attacker may wish to inject false location information into the network in order to accomplish a wormhole attack, or a Sybil attack, which could both be detected if precise locations of each node were known. If a node relies on special “anchor” nodes, or infers its location from the location of other nodes, it must be able to determine its location even if some of its neighbors are malicious. This problem has been considered by Alfaro et al. [4], who provide a protocol that allows a node to determine its location using neighbor-supplied data, as long as the number of malicious neighbors is below a certain threshold.

Localization protocols are utilized in Chapter 4 as a tool to perform collaborative network discovery. Including location information in route discovery messages allows for the detection of malicious nodes claiming to have the ability to communicate with nodes outside of their communication range.

1.2.3 Operation

The operation phase is where a sensor network spends most of its lifetime. During this phase nodes are actively collecting sensor readings and forwarding them to the base station.

Routing

Routing messages within a sensor network is dependent on knowledge gained during the network discovery phase. ARAN [97], S-AODV [114], and DV-SRP [83] are examples of general routing protocols for sensor networks, and they generally focus on determining the shortest path from a node to its destination. The limited availability of battery power to send messages must be taken into consideration when routing messages, so that no single node receives a significantly higher load than others while routing messages. A more complete survey of sensor network routing protocols is given by Al-Karaki and Kamal [3] and Akkaya and Younis [1], and a survey of multi-path routing protocols is given by Stavrou and Pistillides [99].

Network discovery and routing are closely related. The above protocols provide methods for discovering a route from a source to a destination, while the network discovery problem considered in this thesis is concerned with learning the complete topology within a subset of the network. With complete knowledge of the topology, determining a route to a destination is made much easier.

Aggregation

Because broadcasting a message is usually the most draining operation a sensor node performs, it is desirable to minimize the total number of messages sent when collecting sensor readings. For this reason, sensor readings may be aggregated hop-by-hop rather than having each node send its individual reading back to the base station. Rajagopalan and

Varshney [90] survey several techniques for data aggregation in wireless sensor networks. In an adversarial setting, a malicious node may be in charge of forwarding a single message that contains readings from many other nodes in the network. Various approaches to enforcing correct behavior or detecting malicious behavior include hash tree-based approaches [25,40], directed diffusion-based approaches [96], concealed aggregation using homomorphic encryption [81,93], and homomorphic signatures [64]. In general, aggregation techniques are closely related to the underlying network topology and the communication model of the sensor network. In homogeneous networks, push and pull diffusion techniques are common. In the former, nodes initiate the propagation of sensor readings toward the base station, while in the latter the base station itself initiates the protocol by advertising the type of information it is seeking. In hierarchical and group-based networks, the underlying topology provides natural points for aggregating data, such as a clusterhead aggregating readings for all nodes it is coordinating. The aggregation problem is considered in much greater detail in Chapter 5.

Perfectly Secure Message Transmission

Perfectly Secure Message Transmission (PSMT) [32,106] allows a message to be transmitted from a source to a destination in such a way that the sender is guaranteed that the message is delivered and is unaltered as long as the assumptions of the protocol are met. The message is split across multiple node-disjoint paths and is reliably delivered if fewer than a third of the paths contain an adversarial node. Interactive variants of PSMT can also be constructed as long as fewer than one half of the paths contain an adversarial node. Perfectly secure message transmission has been used as a building block for key establishment schemes [108], and can be used as a tool for building secure reliable aggregation protocols.

This thesis does not improve any prior results in perfectly secure message transmissions, but solves the related problem of determining multiple node-disjoint paths within the network so that multi-path protocols, such as PSMT, can be utilized. The discovery of such paths is not considered in the above proposals.

1.2.4 Management/Maintenance

During the lifetime of the network, non-sensing tasks, such as key refreshing or the addition of new nodes, may be required.

Adding/Removing Nodes

Due to the limited lifetime of sensor nodes, it may be desirable in some settings for new nodes to be periodically added to the network to replace nodes that have failed. Networks operating in this setting require that protocols used for network discovery, routing, and key pre-distribution be flexible enough to accommodate the addition of nodes post-deployment. Additionally, nodes deployed in a hostile environment may be compromised or become damaged. Honest nodes should possess the ability to detect and inform others of malicious nodes when possible, in an effort to limit their ability to interfere with normal network operation.

The flexible key pre-distribution schemes presented in Chapter 3 allow for future network growth with deterministic performance with respect to both connectivity and resilience.

Key Refreshing

When compared to the problems of key pre-distribution and key establishment, the problem of key refreshing in sensor networks has seen relatively little attention. As an adversary captures deployed nodes, more keys are leaked and the overall security of the network is diminished. By periodically refreshing keys in the network, forward secrecy can be achieved, eliminating the ability of the adversary to decrypt messages that have been recorded prior to a node's compromise. This problem has been considered by Blackburn et al. [10], as well as Guo and Leung [42], for example. A broader survey is given by He et al. [45].

Adversary Detection

The problem of detecting malicious nodes in sensor networks has been well studied. A wireless intrusion detection system was proposed by Chen et al. [26], while the problem of detecting Sybil attacks/cloned nodes has been considered by Li and Gong [64, 65], among others. The problem of wormhole detection is also well studied, with results by Hu et al. [50], Bose et al. [15], and Dong et al. [33]. Detecting a passive adversary is difficult, which is why many secure sensor network protocols require that two nodes only exchange sensitive information if they have some sort of cryptographic key in common. Key pre-distribution schemes, like the family presented in Chapter 3, facilitate initial secure communication the network, and the network discovery protocol presented in Chapter 4 allows nodes to learn the local network topology so that multi-path key establishment can be used to establish a key with a neighbor, even if an active adversary is present. The aggregation protocols in Chapter 5 actively detect malicious behavior and alert the base station when it occurs.

1.3 Organization and Contributions

This thesis presents a suite of secure protocols that touch on all phases of a sensor network's lifetime, thereby allowing nodes to self-organize and operate in the presence of an active malicious adversary. Moreover, the protocols presented here do not utilize computationally expensive primitives, such as public-key cryptography, making them suitable for use on virtually any class of sensor hardware. The specific contributions, as they relate to each phase, are summarized here. For reference, a summary of notation used in this thesis is provided in Appendix A.

1.3.1 Pre-deployment

The pre-deployment phase is concerned with all aspects of a node prior to deployment. Our goal is to begin with a minimal set of assumptions about node capability, and add specific requirements as needed. For this reason, we rely only on the use of symmetric keys, and assume no public-key infrastructure is available. This approach provides functionality that targets the widest possible range of applications. The availability of protocols that make minimal assumptions about node capability allows resources to be spent on minimizing the size and cost of nodes, rather than on increasing their capability to accommodate secure protocols.

The main contribution in pre-deployment is given in Chapters 2 and 3, where a family of flexible key pre-distribution schemes is presented. Chapter 2 explains the theory behind combinatorial designs and demonstrates how they can be used for key pre-distribution. These combinatorial designs are highly structured and well-studied, with a wide body of results to draw from, and allow for precise mathematical analysis. Unfortunately, their highly structured nature comes at the cost of rigid parameter choices.

Chapter 3 presents two new families of combinatorial key pre-distribution schemes that directly address the problem of inflexible parameter choice. The first family decomposes the underlying combinatorial design into several smaller designs, allowing for a precise performance analysis of numerous fixed-size subsets of the complete design. This decomposition greatly expands the range of network sizes that the scheme can be applied to, without sacrificing the ability to precisely analyze the expected performance of the scheme. The second family considers the use of random subsets of arbitrary size, thus allowing the construction of networks of any size with predictable performance. The trade-off for complete flexibility in network size is a loss of precise mathematical analysis of performance. We compensate for this by providing several simulations that confirm the expected performance of our approach.

Chapter 3 contains joint work published with Maura Paterson [47].

1.3.2 Deployment/Setup

Chapter 4 presents a collaborative approach to secure network discovery in the presence of an active adversary. We assume that nodes have been distributed in a random topology with no prior knowledge of their expected neighbors in the network. The only pre-loaded information that each node possesses is a set of keys issued according to the "linear" key pre-distribution scheme presented in Chapter 3.

The protocol in Chapter 4 is partly motivated by the existence of multi-path protocols, such as multi-path key establishment and perfectly secure message transmission. These protocols assume that an adversary has compromised some subset of the network, and utilize multiple node-disjoint paths to allow nodes to securely establish keys or send messages. These approaches assume the existence of multiple node-disjoint paths, and leave the problem of actually establishing said paths as an open problem.

We present a protocol for establishing the local network topology to an arbitrary depth even when an active adversary is present from the very beginning. This adversarial assumption is stronger than assumptions many other protocols use, where it is assumed that the adversary is not present until some time after deployment. The protocol is consensus-based, and only accepts the existence of a node and its neighbors if a majority of the received information is consistent. This property prevents an adversary from injecting false information as long as adversarial nodes do not form a majority in any portion of the network. Simulated results are included to demonstrate the effect of random node compromise on the assumption that the majority of nodes in a given neighborhood are honest.

The network discovery protocol presented here ensures that each node can present only one identity to the network, and that its identity must be consistent with the keys it was issued. This property ensures that a malicious node cannot present multiple identities or convince an honest node that it lies on more than one specific path between two nodes.

Therefore, paths produced by our protocol are suitable for use in protocols like perfectly secure message transmission and multi-path key establishment.

This protocol was presented previously [48].

1.3.3 Operation

The main goal of a sensor network is to capture and forward data to a central location. A common technique for saving energy during this process is to aggregate data hop-by-hop as it flows towards the base station, which allows for a single flow of information to capture the state of the network, rather than having each node report its readings individually.

Chapter 5 presents a family of aggregation protocols designed to aggregate over linear networks. Although linear networks are simple, they are a naturally arising topology in sensor networks. Many of the existing aggregation techniques assume a tree-based structure, which, while applicable to a linear network, tend to provide worst-case performance when used with them. Our protocols exploit the linear topology of the network to provide data aggregation with real-time adversary detection, which we accomplish by utilizing multiple natural key pre-distribution schemes for linear networks.

Although our aggregation protocols are based on a linear topology, we demonstrate that linear networks arise naturally in many general settings and demonstrate a method for utilizing our protocols over arbitrary topologies.

1.3.4 Maintenance

The main contributions in this thesis are three families of secure protocols for combinatorial key pre-distribution, network discovery, and data aggregation, which directly targets the pre-deployment, deployment, and operation phases, respectively. While none of these directly target the maintenance phase of a sensor network, all three protocols have an impact on it.

The flexible key pre-distribution schemes presented in Chapter 3 allow for a high degree of flexibility over parameter choice. This flexibility allows us to choose parameters for a larger than necessary number of nodes, while retaining predictable performance when only a subset of these nodes are deployed. Thus, parameters can be chosen to allow for new nodes to be added to the network without any significant change in the performance of the underlying key pre-distribution scheme, and without having to issue additional keys to already deployed nodes.

Chapter 4 describes a network discovery protocol that is secure in the presence of an active adversary. While this protocol is intended to be run by all nodes concurrently, the same approach can be repeated when new nodes are added into the network, thus allowing them to discover the local network topology.

Aggregation techniques prolong the lifetime of a sensor network by reducing the total energy expenditure when collecting data from all nodes. The aggregation protocols in Chapter 5 are designed to allow for runtime detection of malicious behavior so that the protocol can be aborted early and the malicious nodes can be dealt with. Depending on the deployment scenario, nodes could update the topology to route around potentially malicious nodes, or the network administrator could be alerted to replace said nodes.

Chapter 2

An Overview of Combinatorial Key Pre-distribution

The previous chapter provided a brief introduction to sensor network research. Although sensor nodes are understood to be computationally limited, memory limited, and power limited, they encompass such a wide variety of applications that it is difficult to provide specific details. Several libraries exist that provide public key and elliptic curve cryptography on sensor nodes, and there are multiple platforms which are powerful enough to utilize them. At the same time, researchers continue to push for smaller nodes, often at the expense of computational ability or available storage.

The continual shrinking of sensor nodes suggests that there will always a class of sensors for which public key cryptography is simply too expensive to implement. In these settings, it makes sense to pre-load sensors with a small set of symmetric keys that will enable secure communication after deployment. These keys are issued according to a *key pre-distribution scheme (KPS)*, with each node possessing some small *keyring* from a larger master *keylist*.

This chapter is concerned with the pre-deployment phase of a sensor network, and is intended to establish the necessary background theory to derive a new family of flexible key pre-distribution schemes for sensor networks. In particular, we are concerned with *combinatorial* key pre-distribution schemes, which utilize well-studied combinatorial designs to determine which keys are issued to which nodes.

We begin with a brief introduction to the key pre-distribution problem and establish some useful metrics in evaluating the performance of a KPS. This is followed by an overview of several families of combinatorial designs and a demonstration of how they can be naturally mapped to key pre-distribution schemes. Two of these schemes, the “linear” and "quadratic" schemes, are presented and analyzed in detail, as they form the basis for a new family of KPSs in the next chapter.

2.1 Key Pre-distribution

In 2002, Eschenauer and Gligor [37] identified three fundamental problems with respect to the use of symmetric key cryptography in WSNs:

1. **Key Pre-distribution:**
   - How do we assign keys to individual sensor nodes?
2. **Shared-Key Discovery:**
   - Two nodes can only communicate if they are within each other's communication range, and they have a common shared key.
   - How do two nodes determine if they share a common key?
3. **Path-Key Establishment:**
   - Nodes that cannot communicate directly should be able to communicate via one or more *multi-hop* paths.
   - How can we efficiently determine secure multi-hop paths?
   - Ideally, a *two-hop* path is preferred.

By multi-hop path, we mean a path of the form $n_1 \to n_2 \to \ldots \to n_k$ where $k \geq 3$. A two-hop path is a path of the form $n_1 \to n_2 \to n_3$. A solution to the key pre-distribution problem can be evaluated with respect to several metrics, the most important of which are described below.

Storage Requirements

Because sensor nodes are typically constrained with respect to storage, the total number of keys, denoted by k, should be kept "small". Although the total amount of space that can be dedicated to key storage may vary between different models of sensors, it is common to aim for a system that can achieve acceptable performance with $k < 100$. In general, the smaller the value of k, the more difficult it becomes to balance connectivity and resilience.

Network Connectivity

In the most general network settings, no information about the network topology is known prior to deployment. Therefore, it is desirable that any random pair of nodes in the network have a reasonably high probability of sharing a key with each other. The probability that two randomly chosen nodes share at least one key is denoted by Pr_1.

Network Resilience

In an adversarial setting, it is assumed that one or more nodes will eventually be compromised, and all keys possessed by a compromised node will be revealed to the adversary. Therefore, the impact of an adversary learning a subset of the keys must be understood. If s nodes are compromised at random and have their keys revealed, then fail(s) denotes the probability that a random link between two honest nodes in the network is compromised as a result. The metric fail(1) is often used, as it demonstrates the impact of a single random node becoming compromised. Note that a *smaller* value of fail(1) corresponds to a *higher* degree of resilience in the network.

In general, computing a precise value for fail(1) is straightforward, but computing a general formula for fail(s) is not. Lee and Stinson [60] provide a more accurate estimate for fail(s) for two of the KPSs considered later in this chapter, with results summarized in Tables 2.1 and 2.2.

Computational and Communication Requirements

Intuitively, increasing the number of keys per node, k, leads to a greater degree of connectivity, as the probability that two nodes share a key, Pr_1, increases. However, this comes at the cost of increasing fail(1) as result, as a larger number of keys are leaked when a single node is compromised. It is preferable if a KPS can optimize the trade off between connectivity and resilience without incurring high storage cost, and that it be flexible enough that parameters can be chosen to suit a wide range of network requirements. Because sensor nodes are computationally limited and energy limited, any scheme attempting to address these problems must do so without incurring a high computation or communication cost.

Additional Considerations

While most key pre-distribution schemes are concerned with striking a balance between storage, connectivity, and resilience, it is also important that a KPS be flexible enough to apply to a broad class of networks. For example, it may be desirable for a KPS to allow for one parameter, such as storage, to be fixed, while still allowing the balance between resilience and connectivity to be tweaked. Some applications may require support for nodes to be added or removed from the network. Others approaches may rely on information about the network, such as topology, being known in advance.

2.1.1 Naive Schemes

The main metrics considered in the previous section are easily demonstrated through the following naive solutions to the key pre-distribution problem.

Scheme 1. ***The Trivial KPS.*** *Given a network $\mathcal{N}$, choose a single master key K. For each $n \in \mathcal{N}$, give key K to node n.*

The trivial KPS simply assigns the same master key to each node in the network. This scheme achieves optimal storage ($k = 1$), only a single key per node, as well as perfect connectivity ($Pr_1 = 1$), as any pair of nodes possess the shared key K. The trade-off for perfect connectivity and storage in this case is worst-case performance with respect to resilience. In the trivial KPS, $\text{fail}(1) = 1$, meaning every link in the network is no longer secure once a single node is compromised. This scheme demonstrates the theoretical lower bound for storage and connectivity, and an upper bound for resilience and connectivity.

Scheme 2. ***The Pairwise KPS.*** *Given a network of $\mathcal{N}$ of N nodes, for each pair of distinct nodes $n_i, n_j \in \mathcal{N}$, give nodes n_i and n_j the distinct pairwise key $K_{i,j}$.*

The pairwise scheme assigns a unique key to every distinct pair of nodes in the network, thus achieving perfect connectivity ($Pr_1 = 1$), as well as perfect resilience ($\text{fail}(1) = 0$). These properties come at the cost of worst-case storage, with $k = N - 1$. The pairwise scheme demonstrates an upper bound on necessary storage, as well as a theoretical lower bound on resilience.

Scheme 3. ***The Unique Key KPS.*** *For a network $\mathcal{N}$, give each node $n \in \mathcal{N}$ a unique key.*

The unique scheme assigns each node in the network a unique key. This scheme achieves optimal storage ($k = 1$), as well as perfect resilience (as there are no links to compromise), but at the cost of worst-case connectivity ($Pr_1 = 0$). Note that the performance in this case is identical to issuing no keys at all. We assume that at least one key will be issued to each node in a KPS.

2.1.2 Randomized Schemes

The Eschenauer-Gligor Randomized KPS

Along with identifying the main key pre-distribution problems outlined in this chapter, Eschenauer and Gligor [37] also proposed a simple solution that achieves a better balance between resilience and storage than the naive schemes.

Scheme 4. ***The Randomized KPS.*** *[37] Given a network $\mathcal{N}$, choose a master key list $\mathcal{K}$ of v distinct keys. For each node $n \in \mathcal{N}$, give a random subset of size k from $\mathcal{K}$ to node n.*

Figure 2.1 contains a simple example with a master key list $\mathcal{K}$ of size $v = 100$ where $\mathcal{K} = \{1, \ldots, 100\}$, and each node is issued ten keys.

In the Eschenauer-Gligor Randomized KPS, each node is issued a random k-subset from a master key list $\mathcal{K}$ of size v. Two nodes perform shared-key discovery by exchanging a list of key identifiers and simply checking their own keyring against the received list. Path-key establishment is performed by comparing the key lists of neighbors for common keys to determine a secure path to any nearby node with which a key is not shared. A variant of this approach is to generate the identifiers in each node's keyring using a keyed pseudo-random function. This approach shifts a majority of the communication cost to computational cost, as nodes simply exchange seeds rather than entire lists of keys. Therefore, the computational and communication costs are:

- Exchanging key lists: $O(k)$ communication and $O(k)$ computation
- Exchanging seeds: $O(1)$ communication and $O(k \log k)$ computation (if the key list must be sorted; more efficient approaches may be possible).

The connectivity of the scheme is given by

$$\begin{aligned} Pr_1 &= 1 - \frac{\binom{v-k}{k}}{\binom{v}{k}} \\ &= 1 - \frac{((v-k)!)^2}{v!(v-2k)!} \\ &\approx \frac{k^2}{v} \qquad \text{(when } v \gg k\text{)}. \end{aligned}$$

N_1	$= \{\mathbf{17}, \mathbf{21}, \mathbf{31}, \mathbf{41}, \mathbf{47}, \mathbf{73}, \mathbf{75}, \mathbf{85}, \mathbf{93}, \mathbf{97}\}$	Shared keys with N_1
N_2	$= \{19, 54, 59, 66, 67, 72, 72, 80, 91, 92\}$	
N_3	$= \{28, 42, 44, 45, 51, 63, 65, 73, 80, \mathbf{97}\}$	$\{97\}$
N_4	$= \{2, 8, 19, 21, 22, 42, 66, 71, \mathbf{97}, 100\}$	$\{97\}$
N_5	$= \{3, 12, 13, 60, 77, 90, 92, 94, 98, 99\}$	$\emptyset$
N_6	$= \{39, 42, 45, 46, 51, 52, 61, 69, 82, 89\}$	$\emptyset$
N_7	$= \{14, 23, \mathbf{31}, 33, 34, 55, 57, 74, 81, 87\}$	$\{31\}$
N_8	$= \{14, 18, 40, 54, 58, 68, 71, \mathbf{85}, 87, 93\}$	$\{85\}$
N_9	$= \{11, 18, 20, 42, 49, 64, 76, 81, 89, 99\}$	$\emptyset$
N_{10}	$= \{6, 18, 26, 37, 39, 57, 66, 81, 84, \mathbf{93}\}$	$\{93\}$

Figure 2.1: Example of the Eschenauer-Gligor KPS with a master key list $\mathcal{K} = \{1, \ldots, 100\}$, where each node is assigned $k = 10$ keys. Node N_1 shares a common key with nodes N_3, N_4, N_7, N_8, and N_{10}. The link between uncompromised nodes N_1 and N_3 is only compromised if node N_4 becomes compromised.

The resilience of the scheme is given by

$$\begin{aligned} \text{fail}(1) &\approx \frac{\binom{v-1}{k-1}}{\binom{v}{k}} \\ &= \frac{k}{v}. \end{aligned}$$

This approximation holds when each pair of nodes are using a single key to communicate. In practice, nodes sharing more than one common key could derive a new key from their entire set of shared keys. If $v \gg k^2$, then it is expected that most links in the network will be secured using only a single key. A general formula for fail(s) is given by Kendall et al. [55].

Using the above formulas, it can be shown that relatively high levels of connectivity can be achieved with a small number of keys per node. For example, let $v = 10000$ and $k = 75$. Then we have

$$Pr_1 \approx \frac{75^2}{10000} = 0.5625 \qquad \text{fail}(1) \approx \frac{75}{10000} = 0.0075.$$

Therefore, each node in the network has a greater than 50% chance of sharing a key with a random neighbor, but a smaller than 1% chance that a given link with a neighbor will be insecure if the adversary captures a random node.

Because the keyring for each node is selected independently, the randomized KPS allows additional nodes to be added to the network by simply choosing a new random keyring for each new node. Depending on the application, the randomized KPS may require additional bookkeeping by a network administrator to keep track of which keys are assigned to which nodes.

The q-composite KPS

In the previous section it was noted that in the randomized KPS, two nodes may share more than one key. Chan at al. [24] have advocated choosing parameters such that two nodes communicate if and only if they have at least q keys in common. From these q keys, a *key derivation function* can be used to derive a new pairwise key between nodes. The goal is to provide a higher degree of resilience, as a larger numbers of nodes must be compromised before the adversary is expected to learn a specific set of q keys.

Scheme 5. ***The q-composite KPS.*** *[24] Given a network $\mathcal{N}$, choose a master key list $\mathcal{K}$ of v distinct keys. For each node $n \in \mathcal{N}$, give a random subset of keys from $\mathcal{K}$ of size k to node n. Two nodes must have at least q keys in common to establish a secure connection.*

Two nodes that share at least $q' \geq q$ keys $\{k_1, \ldots, k_{q'}\}$ can derive a new shared key by computing

$$k' = \text{HASH}(k_1||k_2||\ldots||k_{q'}),$$

where HASH is a suitable hash function, and the keys are sorted in some agreed upon order. Analysis of the q-composite scheme is more complicated than the randomized scheme. The probability that two nodes share exactly i keys is given by

$$Pr(i) = \frac{\binom{v}{i}\binom{v-i}{2(k-i)}\binom{2(k-i)}{k-i}}{\binom{v}{k}^2}.$$

Therefore, we can compute the probability that two nodes have at least q keys in common by

$$Pr_1 = 1 - (Pr(0) + Pr(1) + \ldots + Pr(q-1)).$$

The resilience of the scheme is given by

$$\text{fail}(s) = \sum_{i=q}^{k} \left(1 - \left(1 - \frac{k}{v}\right)^s\right)^i \frac{Pr(i)}{Pr_1}.$$

Using the above formulas we can demonstrate how changing the value q affects performance. Let $v = 5000$ and $k = 75$.

q	Pr_1	fail(1)
1	0.6808	0.008223
2	0.3108	0.000152
3	0.1019	0.000002

For each additional key two nodes use to communicate, the probability than an adversary can compromise the link drops dramatically. However, this is accompanied by a significant drop in Pr_1 as well.

The computation and communication costs of the q-composite scheme are identical to the randomized scheme, with the small additional cost of the key derivation function. Similarly, it is trivial to add nodes to the network, and the same bookkeeping must be done to track which keys have issued to which nodes.

2.1.3 But Why Rely on Randomness?

Randomized approaches to key pre-distribution provide a simple and straightforward solution to the key pre-distribution problem, but a consequence of assigning keys randomly is the inability to make precise statements about the performance of the KPS. Although the expected connectivity of the randomized and q-composite schemes is quite high with the appropriate choice of parameters, there is still a small chance that the key graph of the KPS may be disconnected. It also possible that certain keys will be held by a larger than expected number of nodes, leading to a loss of resilience, or that some keys may be held by a very small number of nodes, leading to unnecessary storage. Because keys are assigned randomly, a network administrator requires a large amount of randomness to assign keys, and must also keep track of which keys are issued to which nodes if they wish to perform simple tasks such as computing network statistics or assessing the impact of a specific node compromise.

In the previous sections it was established that when using a randomized KPS, nodes must have a mechanism by which they can determine which keys are shared with their neighbors. One option is use $O(k)$ communication and $O(k)$ computation to search the key list. An alternative is to use $O(1)$ communication to transmit a seed for a pseudorandom function, and $O(k \log k)$ computation to compute, sort, and search the key list. Due to the fact that communication tends to be orders of magnitude more expensive than computation, the latter case is generally preferable for any practical set of parameters. In memory-limited settings, another benefit of generating key identifiers from a seed is the ability to store another node's list of keys simply by storing the seed.

The benefits of using a random seed to generate a node's list of keys exemplifies why deterministic approaches may be a better option for key pre-distribution. A fully deterministic key pre-distribution scheme does not rely on any randomness to distribute keys, allows for precise analysis, and does not require and additional bookkeeping by a network administrator. If keys are distributed according to a more structured object than a simple ordered list, then it is also possible to achieve computational costs that are lower than the approaches described in this section.

A natural approach to deterministically distributing keys is to use some sort of structured, symmetric pattern. Because entire branches of mathematics and combinatorics are concerned with the study of repeating and overlapping structures, there is a rich body of already-established results that can be naturally applied to the key pre-distribution problem. Furthermore, these structures may have additional properties that are useful in the context of sensor networks. The remainder of this chapter is dedicated to evaluating a subset of these approaches.

2.2 Combinatorial Design Theory

Combinatorial design theory is concerned with arranging elements of a finite set in such a way that certain "nice" or "balanced" properties are achieved. The notion of a "nice" or "balanced" property is vague, but may, for example, refer to the separation of elements into different subsets such that certain numerical properties hold. The question of whether or not such an arrangement exists, and, if so, how many exist and how they can be efficiently constructed, all fall under the umbrella of design theory. More specifically, this section is concerned with a subset of designs known as *block designs*, which lend themselves naturally to the key pre-distribution problem.

A full description of the relationship between a design and a key pre-distribution scheme is given in Section 2.3, but the relevant parallels are also described alongside the definitions and examples of designs in this section.

2.2.1 Set Systems

The most general combinatorial design is a *set system*.

Definition 1. *A set system, or design, is a pair $(X, \mathcal{A})$, where*

1. *X is a set of elements, called points, and*
2. *$\mathcal{A}$ is a collection of non-empty subsets of X, called blocks.*

As an example, consider the following:

$$\begin{aligned} X &= \{1,2,3,4,5,6,7\} \\ \mathcal{A} &= \{\{1,2,3\},\{1,4,5\},\{1,6,7\},\{2,4,7\},\{2,5,6\},\{3,4,6\},\{3,5,7\}\}\,. \end{aligned}$$

This set system also has some additional structure not required by the definition of a set system. For example, choose any point $x \in X$. The *degree* of x is the number of blocks in $\mathcal{A}$ that contain x. In this example, every point has degree 3. Such a system is said to be *regular*, of degree 3. Similarly, choose any block $b \in \mathcal{A}$. The *rank* of b is the number of points in the block. In this example, each block has rank 3. Such a system is said to be *uniform* of rank 3.

If the points in X are cryptographic keys, then blocks can be thought of as keyrings. A uniform design is one where each keyring is the same size, and a regular design is one where each key appears in the same number of keyrings. These properties provide the foundation for building a KPS from a combinatorial design.

2.2.2 Configurations

A set system that is both regular and uniform suggests a strong notion of symmetry within the design. When constructing a KPS from a such a design, both regularity and uniformity are desirable properties. Such systems are referred to as *configurations.*

Definition 2. *A (v, b, r, k)-configuration is a set system $(X, \mathcal{A})$ where*

1. *$|X| = v$ and $|\mathcal{A}| = b$,*
2. *$(X, \mathcal{A})$ is regular of degree r,*
3. *$(X, \mathcal{A})$ is uniform of rank k, and*
4. *every pair of points occurs in at most one block.*

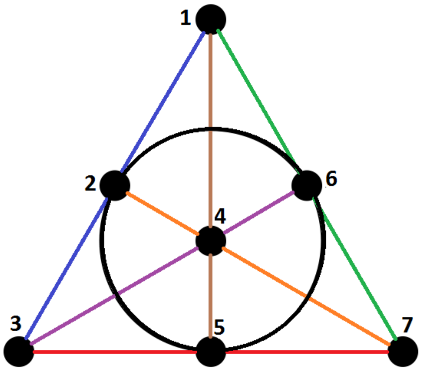


Figure 2.2: A $(7, 7, 3, 3)$-configuration, also known as the Fano plane. Each dot is a point, and each straight line (or circle) denotes a block. Note that any pair of points in this configuration has only a single line (or circle) that passes through both of them.

The set system described in the previous section also happens to be a configuration. Figure 2.2 shows a visualization of this configuration, and makes it easy to see that every pair of points occurs in exactly one block. Alternatively, this property can be restated as any two blocks intersect at at most a single point, which suggests an interesting property for a KPS derived from a configuration. In a configuration-based KPS, any pair of keys appears in at most one keyring.

Consider a configuration where every pair of points occurs in exactly one block. Such a design is highly structured and straightforward to analyze, which allows us to make precise statements about the behavior of a configuration-based KPS. Each block in such a system contains k points, and each of these points is contained in exactly r blocks in the design. Therefore, given a block $A \in \mathcal{A}$ and a point $x \in A$, there exist $r - 1$ other blocks in the design that contain x. Furthermore, because every pair of points occurs in at most one block, these $r - 1$ blocks intersect A only at the point x. This property holds for each of the k points in A. Because there are $b - 1$ other blocks in the design, we can compute the probability that two random blocks intersect by

$$Pr_1 = \frac{k(r-1)}{b-1}. \tag{2.1}$$

Recall the natural mapping of points to keys and blocks to keyrings in a KPS, and assume that two blocks intersect at point x. A pair of nodes, each possessing one of these keyrings, are able to form a secure link. An adversary is able to compromise this link if it compromises another node whose keyring contains x. There are $b-2$ other nodes in the network, and only $r-2$ of them possess point x. Therefore,

$$\mathrm{fail}(1) = \frac{r-2}{b-2} \tag{2.2}$$

Equations 2.1 and 2.2 also allow us to demonstrate the trade-off between connectivity and resilience in a configuration-based KPS through the following

$$\rho = \frac{Pr_1}{\mathrm{fail}(1)} = \frac{k(b-2)(r-1)}{(b-1)(r-2)} \approx k. \tag{2.3}$$

This demonstrates that in a naive construction of a configuration-based KPS, there is a direct linear relationship between the connectivity and resilience of the system. If each node stores k keys, then the connectivity will be approximately k times larger than fail(1). Note that this is the same approximate performance as the randomized KPS considered previously. Recall the formulas $Pr_1 \approx \frac{k^2}{v}$ and $\mathrm{fail}(1) = \frac{k}{v}$ for the Eschenauer-Gligor Randomized KPS in Section 2.1.2. It is easy to see that the randomized scheme also provides a trade-off of $\rho \approx k$. The q-composite scheme from Section 2.1.2 is the same when $q = 1$, but gets progressively better as q grows.

2.2.3 Balanced Incomplete Block Designs

A configuration is a regular, uniform set system where any two blocks intersect at at most a single point. Although this property may be useful in some settings, it may be seen as unnecessarily restrictive in others. For example, recall the q-composite KPS, which achieves a high degree of resilience by only allowing nodes who share multiple keys to communicate. This observation suggests that a more general definition could be useful.

Definition 3. *A (v, b, r, k, λ)-BIBD, or balanced incomplete block design, is a set system $(X, \mathcal{A})$ where*

1. *$|X| = v$ and $|\mathcal{A}| = b$,*
2. *$(X, \mathcal{A})$ is regular of degree r,*
3. *$(X, \mathcal{A})$ is uniform of rank k, and*
4. *every pair of points occurs in exactly λ blocks.*

The definition for a BIBD is very similar to a configuration. In a configuration, two blocks intersect at at most a single point, while in a BIBD two blocks intersect at exactly λ points. In particular, a $(v, b, r, k, 1)$-BIBD is always a configuration, but the reverse need not be true. The example design in Figure 2.2 is a $(7, 7, 3, 3, 1)$-BIBD.

2.2.4 Transversal Designs

The definitions in the previous section were all demonstrated using the same example $(7, 7, 3, 3, 1)$-BIBD, and the relevance of each design with respect to KPSs was hinted at. In order for a design to be useful as a KPS, it is not sufficient for nice theoretical properties to exist. We must also be able to efficiently construct such a design for a wide array of parameters. *Transversal designs* are useful in this regard.

Definition 4. *A transversal design $TD(t, k, n)$ is a triple $(X, \mathcal{H}, \mathcal{A})$ where*

1. *X is a finite set of points with $|X| = kn$,*
2. *$\mathcal{H}$ is a partition of X into k groups of size n, and*
3. *$\mathcal{A}$ is a set of k-subsets of X such that*
   (a) *$|H \cap A| = 1$ for every $H \in \mathcal{H}$ and $A \in \mathcal{A}$, and*
   (b) *Every t elements of X from different groups occur in exactly one block in $\mathcal{A}$.*

Transversal designs are named for the fact that nodes are partitioned into groups, and each block transverses the groups, intersecting each group at a single point. Figure 2.3 shows an example of some blocks in a transversal design.

The definition of a transversal design with $t = 2$ is similar to that of a configuration, as any pair of points in a $\text{TD}(2, k, n)$ occur in exactly 1 block. In fact, it can be shown that a $\text{TD}(2, k, n)$ is a (kn, n^2, n, k)-configuration, allowing the use of Equations 2.1 and 2.2 to describe the performance of a $\text{TD}(2, k, p)$-based KPS.

An explicit and efficient construction for a KPS built from transversal designs is given in Section 2.4. Note that Definition 4 includes the parameter n. The constructions considered in Section 2.4 and the remainder of the thesis make use of algebraic fields, so our attention will be limited to transversal designs $\text{TD}(t, k, p)$ where p is a prime or prime power, such that we can choose points in the design as tuples from $\mathbb{F}_p$.

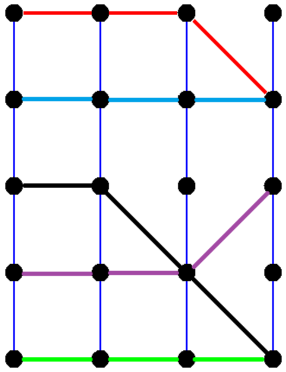


Figure 2.3: Example blocks in a TD(2,4,5). Each vertical line represents a group, and each block is represented by a line that transverses the partition, intersecting each group at exactly one point. Note that two blocks may intersect, but they never overlap at more than one point (because $t = 2$).

2.2.5 Other Designs

Although the focus of this chapter is primarily on regular and uniform designs, such as BIBDs and transversal designs, there exist many other designs that can be utilized for combinatorial key pre-distribution. Comprehensive surveys of such designs include The Handbook of Applied Cryptography [28], and Combinatorial Designs: Constructions and Analysis [101], as well as surveys by Martin focused on the key pre-distribution problem in general [72], and the key pre-distribution problem specifically in sensor networks [73]. Solutions based on generalized quadrangles [20], mutually orthogonal Latin squares [109], inversive planes [34], among others have been proposed. Paterson and Stinson [84] summarize several approaches and introduce *partially balanced t-designs* in an attempt to capture the essential properties of many existing combinatorial design-based KPSs.

2.3 Set Systems as KPSs

The previous sections demonstrated some of the parallels between combinatorial designs and key pre-distribution schemes. The points in a set system naturally map to keys, and the blocks of such a system naturally map to keyrings. This idea was originally explored by Çamtepe and Yener [20, 21], who proposed, among other systems, a KPS based on BIBDs. Since then, a variety of different approaches have been proposed.

To build a KPS from a design we simply associate each node with a distinct block from the set system. In practice, each point is not associated with a key, but rather with a *key identifier.* If a node is associated with a given block, it is issued the key identified by each point in the block. It is assumed that the underlying combinatorial structure, including which block is associated with which node, is public, but that the actual keys remain secret.

Scheme 6. ***The Generic Combinatorial Design KPS.*** *Given a network $\mathcal{N} = \{n_1, \ldots, n_b\}$, choose a set system $(X, \mathcal{A})$ with $X = \{1, \ldots, v\}$ and $\mathcal{A} = \{A_1, \ldots, A_b\}$, and a set of keys $\mathcal{K} = \{K_1, \ldots, K_k\}$. For each $n_i \in \mathcal{N}$, give node n_i the set of keys $\{K_x : x \in A_i\}$.*

Configurations are an ideal candidate for a combinatorial KPS, as they are both regular and uniform. We use the example $(7, 7, 3, 3)$-configuration from the previous section to demonstrate. Recall,

$$\begin{aligned} X &= \{1, 2, 3, 4, 5, 6, 7\} \\ \mathcal{A} &= \{123, 145, 167, 246, 257, 347, 356\}. \end{aligned}$$

We can use this configuration to distribute seven keys to seven nodes using the Generic Combinatorial Design KPS such that each node possesses three keys, and each key is possessed by three nodes.

Node	Block	Keys
n_1	$\{1, 2, 3\}$	$\{K_1, K_2, K_3\}$
n_2	$\{1, 4, 5\}$	$\{K_1, K_4, K_5\}$
n_3	$\{1, 6, 7\}$	$\{K_1, K_6, K_7\}$
n_4	$\{2, 4, 6\}$	$\{K_2, K_4, K_6\}$
n_5	$\{2, 5, 7\}$	$\{K_2, K_5, K_7\}$
n_6	$\{3, 4, 7\}$	$\{K_3, K_4, K_7\}$
n_7	$\{3, 5, 6\}$	$\{K_3, K_5, K_6\}$

While the mapping from a configuration to a KPS is natural, we have not yet specified how two nodes perform shared-key discovery. If there exists an efficient method for computing a block from a node's identifier, then nodes can perform shared-key discovery without exchanging any information beyond their identities. The next sections demonstrate explicit constructions that allow for efficient shared-key discovery.

2.4 The Linear Scheme

In 2005, Lee and Stinson proposed a KPS based on transversal designs [59], which was further studied by the same authors [61]. This scheme is well suited to sensor networks due to the fact that it has a simple construction, allows for two nodes to perform shared-key discovery efficiently, and provides a good balance between connectivity and resilience. In this section we describe a specific instance of Lee and Stinson's KPS that is constructed from a $TD(2, k, p)$, also known as the *linear Lee-Stinson-KPS (LS-KPS)*, or simply the *linear KPS*, or *linear scheme* when the context is clear. We recall that $b = N$ denotes the number of nodes in the network (or blocks in the design), v denotes the total number of keys/points, k denotes the number of keys/points stored by each node, and r denotes the number of nodes (blocks) possessing any given key (point).

Scheme 7. ***The Linear LS-KPS.*** *[**61**] Fix the value k and let p be a prime power such that $2 \leq k \leq p$. Then a linear (v, b, r, k)-LS-KPS can be constructed for $v = kp$, $b = p^2$, and $r = p$. Given a network $\mathcal{N} = \{n_{a,b} : a, b \in \mathbb{F}_p\}$, construct a $TD(2, k, p)$ $(X, \mathcal{A})$ as follows:*

Let $X_1 \subseteq \mathbb{F}_p$ with $|X_1| = k$ and define $X = X_1 \times \mathbb{F}_p$. For each $(a, b) \in \mathbb{F}_p \times \mathbb{F}_p$ define

$$A_{a,b} = \{(x, ax + b) : x \in X_1\}$$

and let $\mathcal{A} = \{A_{a,b} : (a, b) \in \mathbb{F}_p \times \mathbb{F}_p\}$.

Choose a set of keys $\mathcal{K} = \{K_{a,b} : (a, b) \in X\}$. For each node $n_{a,b} \in \mathcal{N}$, give node $n_{a,b}$ the set of keys $\{K_{x,y} : (x, y) \in A_{a,b}\}$.

In the linear LS-KPS, each node is indexed by the tuple (a, b), and the keys possessed by each node are functions of these two values. Conceptually, each node's keys correspond to discrete points along the line $ax + b$ inside some fixed interval. If the slopes of two distinct node's lines are the same, then these lines do not intersect and the nodes do not share a key. If two node's lines have different slopes, then those lines may or may not intersect within a fixed interval.

Shared-key discovery in the linear LS-KPS is straightforward. Let $n_{a,b}$ and $n_{a',b'}$ be two distinct nodes. To determine if they share a key, the following is performed:

1. If $a = a'$ then $n_{a,b}$ and $n_{a',b'}$ do not share a common key.
2. If $a \neq a'$ then compute $x = (b' - b)(a' - a)^{-1} \in \mathbb{F}_p$. If $x \in X_1$ then $n_{a,b}$ and $n_{a',b'}$ share the common key $K_{x,ax+b}$. Otherwise $n_{a,b}$ and $n_{a',b'}$ do not share a common key.

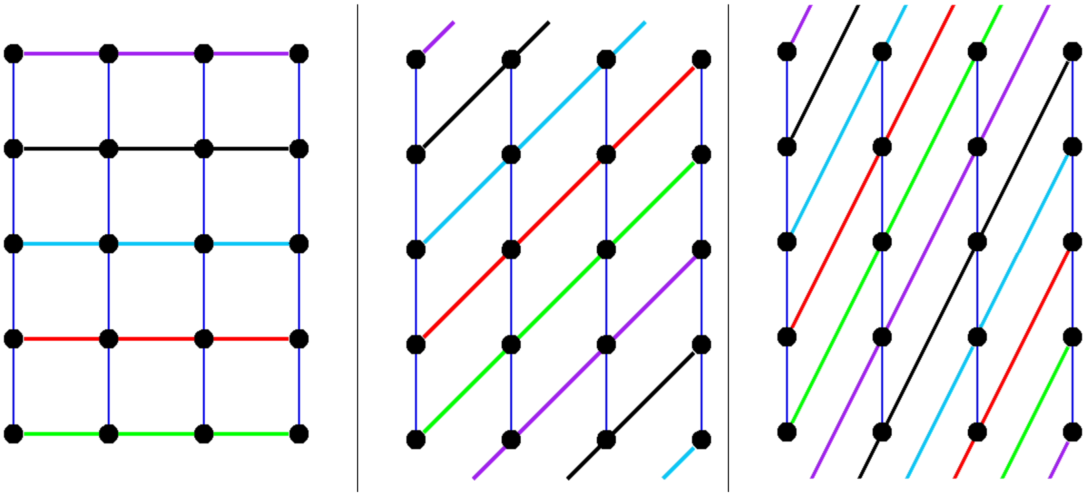


Figure 2.4: Some sample blocks in the linear LS-KPS. Each non-vertical line denotes a block (wrapping from the top to the bottom), and each dot denotes a point. If two nodes are associated with lines that have the same slope, then their lines will never intersect. If two of these diagrams are drawn on top of each other then it is easy to see that many intersections take place.

In the first step, the two nodes are determining whether or the slope of the line specifying their keys is the same. Therefore, in some cases, two nodes require only a single comparison to determine if a key is shared. In the second step, the intersection point is computed, and, if it lies in the correct range, the two nodes have computed the label of their shared key. The computation performed in this step is dominated by a single multiplication of an inverse in $\mathbb{F}_p$, where p is the square root of the total number of nodes. In practice, p will be a small two or three digit prime, making this a very computationally inexpensive operation.

The following theorem is useful for evaluating the performance of the linear KPS.

Theorem 1. *(Lee and Stinson [61, Theorem 4.1]) The design constructed in Scheme 7 is a (v, b, r, k)-configuration (or equivalently, a (kp, p^2, p, k)-configuration).*

Proof. Recall the properties of a configuration from Definition 2.

1. $X = X_1 \times \mathbb{F}_p$ and $|X_1| = k$. Therefore $|X| = kp$. $\mathcal{A} = \{A_{a,b} : (a, b) \in \mathbb{F}_p \times \mathbb{F}_p\}$. Therefore $|\mathcal{A}| = p^2$ and property 1 is satisfied.

2. Choose any point $(x, y) \in X$ and choose any $a \in \mathbb{F}_p$. Then we can compute $b = y - ax \in \mathbb{F}_p$, which shows that the point (x, y) occurs at least once for each $a \in \mathbb{F}_p$. Let $b_2 \in \mathbb{F}_p$ with $(x, y) = (x, ax + b_2)$. Then $ax + b = ax + b_2 \in \mathbb{F}_p$, and it must be true that $b = b_2$. Therefore, each point (x, y) occurs exactly once for each $a \in \mathbb{F}_p$, satisfying property 2.

3. Each block $A_{a,b}$ contains k distinct points, one for each value in X_1. Therefore, property 3 is satisfied.

4. Let (x_1, y_1) and (x_2, y_2) be distinct points in the same block $A_{a,b}$, so $x_1 \neq x_2$. Then $y_1 = ax_1 + b$ and $y_2 = ax_2 + b$. Consider

$$y_2 - y_1 = ax_1 + b - (ax_2 + b) = a(x_2 - x_1).$$

rearranging, we have $a = \frac{y_2 - y_1}{x_2 - x_1}$. Therefore, any two points uniquely determine the value of a. It was demonstrated previously that for any fixed point and value of a, there is exactly one value of b such that $A_{a,b}$ contains that point. Therefore, any pair of distinct points (x_1, y_1) and (x_2, y_2) occur in exactly one block $A_{a,b}$, so satisfying property 4 is satisfied.

□

Furthermore, it can be shown that the construction in Scheme 7 is also a $TD(2, k, p)$.

Theorem 2. *(Lee and Stinson [61, Theorem 4.2]) The design constructed in Scheme 7 is a $TD(2, k, p)$.*

Proof. Let $x \in X_1$, define $H_x = x \times \mathbb{F}_p$, and define $\mathcal{H} = \{H_x : x \in X_1\}$. We show that $(X, \mathcal{H}, \mathcal{A})$ satisfies the properties in Definition 4.

1. By Theorem 1, $|X| = kp$, and property 1 is satisfied.

2. By definition, $\mathcal{H}$ is a partition of X into k groups of size p, where all points in X with the same x-coordinate are part of the same group. Therefore, property 2 is satisfied.

3. Let $A_{a,b} \in \mathcal{A}$. For a given $x \in X_1$, there is exactly one point $(x, y) \in A_{a,b}$ such that $(x, y) = (x, ax + b)$. Therefore, $|A_{a,b} \cap H_x| = 1$, satisfying property 3a. It was shown in Theorem 1 that any two distinct points occur in exactly one block of $\mathcal{A}$, satisfying property 3b.

□

It follows that Scheme 7 is simply a concrete construction for a Generic Combinatorial Design KPS based on a $TD(2,k,p)$. The partition $\mathcal{H}$ of the design can be seen in Figure 2.4, where the points are partitioned evenly as columns. Each block transverses these points as a straight line, with block $A_{a,b}$ containing a set of points arranged along a line of slope a. More specifically, Figure 2.4 demonstrates a subset of blocks from a $TD(2,k,p)$ with $p = 5$ and $k = 4$. Lee and Stinson provide the complete list of blocks in this design [61]. Let $X_1 = \{0,1,2,3\} \subseteq \mathbb{F}_p$. Then $X = \{(x,y) : x \in X_1,\ y \in \mathbb{F}_p\}$. Adopting the shorthand notation $(x,y) \to xy$, the points in X can be partitioned according to their x-coordinate:

H_0	H_1	H_2	H_3
$\{00,01,02,03,04\}$	$\{10,11,12,13,14\}$	$\{20,21,22,23,24\}$	$\{30,31,32,33,34\}$

and the blocks of the underlying KPS are:

	$b=0$	$b=1$	$b=2$	$b=3$	$b=4$
$a=0$	$\{00,10,20,30\}$	$\{01,11,21,31\}$	$\{02,12,22,32\}$	$\{03,13,23,33\}$	$\{04,14,24,34\}$
$a=1$	$\{00,11,22,33\}$	$\{01,12,23,34\}$	$\{02,13,24,30\}$	$\{03,14,20,31\}$	$\{04,10,21,32\}$
$a=2$	$\{00,12,24,31\}$	$\{01,13,20,32\}$	$\{02,14,21,33\}$	$\{03,10,22,34\}$	$\{04,11,23,30\}$
$a=3$	$\{00,13,21,34\}$	$\{01,14,22,30\}$	$\{02,10,23,31\}$	$\{03,11,24,32\}$	$\{04,12,20,33\}$
$a=4$	$\{00,14,23,32\}$	$\{01,10,24,33\}$	$\{02,11,20,34\}$	$\{03,12,21,30\}$	$\{04,13,22,31\}$.

These blocks correspond directly to those in Figure 2.4 for $a = 0,1,2$. It is easy to see that each block intersects each group from the partition exactly once. It can also be observed that no two blocks in the same row intersect, and that any distinct pair of points occurs in exactly one block.

Theorem 1 established that the design used in the linear KPS is in fact a (v,b,r,k)-configuration. Therefore, Equations 2.1 and 2.2 can be used to describe the performance of the scheme:

$$Pr_1 = \frac{k(r-1)}{b-1} = \frac{k(p-1)}{p^2-1} = \frac{k}{p+1} \tag{2.4}$$

$$\text{fail}(1) = \frac{r-2}{b-2} = \frac{p-2}{p^2-2}. \tag{2.5}$$

Because the underlying design is a configuration, the same trade-off between connectivity and resilience exists

$$\rho = \frac{Pr_1}{\text{fail}(1)} = \frac{k(p^2-1)}{(p+1)(p-2)} = \frac{k(p-1)}{p-2} \approx k.$$

2.4.1 A More Thorough Analysis

Although a high degree of connectivity between random nodes is typically considered a good thing, a higher degree of connectivity in configuration-based KPSs also implies a higher value of fail(1). For this reason, it may be desirable to choose parameters such that Pr_1 is low, and rely on path-key establishment to provide two-hop paths between nearby nodes. The structure of the linear KPS allows for a simple analysis of the probability that two nearby nodes that do not share a key are able to establish a two-hop path-key.

Let n_i and n_j be two nodes, and let C denote the number of common neighbors of n_i and n_j. Define Pr_2 to be the probability that n_i and n_j do not share a key, but that one or more of their C common neighbors shares a key with both. In other words, Pr_2 denotes the probability that a secure two-hop path is necessary and exists between nodes n_i and n_j.

Let X, Y be two disjoint blocks in a $TD(2, k, p)$ with $X = \{x_1, \ldots, x_k\}$ and $Y = \{y_1, \ldots, y_k\}$ where $x_i, y_i \in H_i$. We are interested in computing the number of blocks that intersect both X and Y; i.e., the number of blocks containing $\{x_i, y_j\}$ for $1 \leq i, j \leq k$. For $i \neq j$, Property 4(b) of Definition 4, states that there is a unique block that contains x_i and y_j. Therefore, whenever $i \neq j$, there exists a block that contains x_i and y_j. In other words, there are precisely $k^2 - k$ blocks that intersect any pair of disjoint blocks X and Y. A design with this property is called a *common intersection design (CID)*, first defined by Lee and Stinson [60]. Such designs have an additional parameter μ, that represents a lower bound on the number of blocks that intersect any pair of disjoint blocks. This parameter was introduced to allow for the analysis of two-hop paths in a network utilizing a combinatorial KPS. Therefore, the underlying design of the linear KPS, a $TD(2, k, p)$, which is also a (kp, p^2, p, k)-configuration, is also a $(kp, p^2, p; k^2 - k)$-CID.

With a precise bound, $\mu = k^2 - k$, on the number of potential two hop paths, the expected probability of such a path existing among C neighbors can be established. There are $p^2 - 2$ other nodes in the network, and $p^2 - 2 - \mu$ of them are unable to aid in establishing a path key. Therefore, the probability that all C neighbors are unable to aid in establishing a path-key is given by

$$\frac{\binom{p^2-2-\mu}{C}}{\binom{p^2-2}{C}}.$$

The probability that a path-key is necessary and that at least one neighbor is able to aid in establishing a path-key is given by

$$Pr_2 = (1 - Pr_1) \times \left(1 - \frac{\binom{p^2-2-\mu}{C}}{\binom{p^2-2}{C}}\right).$$

A simplified estimate of Pr_2 is given by Lee and Stinson [61], based on the observation that μ is likely to be small with respect to p^2

$$\begin{aligned}
Pr_2 &= (1 - Pr_1) \times \left(1 - \frac{\binom{p^2-2-\mu}{C}}{\binom{p^2-2}{C}}\right) \\
&\approx (1 - Pr_1) \times \left(1 - \left(\frac{p^2-2-\mu}{p^2-2}\right)^C\right) \\
&= (1 - Pr_1) \times \left(1 - \left(1 - \frac{\mu}{p^2-2}\right)\right)^C \\
&= \left(1 - \frac{k}{p+1}\right) \times \left(1 - \left(1 - \frac{k(k-1)}{p^2-2}\right)^C\right).
\end{aligned}$$

Using a similar approach, the effect of multiple node compromises can also be estimated

$$\text{fail}(s) \approx 1 - \left(1 - \frac{p-2}{p^2-2}\right)^s,$$

where fail(s) denotes the probability of any given link in the network being compromised after s nodes are captured. Table 2.1 demonstrates the effect on resilience as more nodes are compromised for some sample parameter choices.

Table 2.1: The effect of node compromise on resilience in the linear scheme.

$p = 71$ (approx. 5000 nodes)	
s	fail(s)
1	0.0137
2	0.0272
3	0.0405
4	0.0536
5	0.0666
10	0.129
20	0.241

$p = 157$ (approx. 24000 nodes)	
s	fail(s)
1	0.00629
2	0.0125
3	0.0187
4	0.0249
5	0.0311
10	0.0611
20	0.119

2.5 The Quadratic and Higher Degree Schemes

A larger family of KPSs can be constructed in the same manner as the linear scheme by considering transversal designs of higher *strength*, i.e., when $t > 2$. We first consider the quadratic scheme, based on a $TD(3, k, p)$.

Scheme 8. ***The Quadratic LS-KPS.*** *[61] Fix the value k and let p be a prime power such that $3 \leq k \leq p$. Then a quadratic (v, b, r, k)-LS-KPS can be constructed for $v = kp$, $b = p^3$, and $r = p^2$. Given a network $\mathcal{N} = \{n_{a,b,c} : a, b, c \in \mathbb{F}_p\}$, construct a $TD(3, k, p)$ $(X, \mathcal{A})$ as follows:*

Let $X_1 \subseteq \mathbb{F}_p$ with $|X_1| = k$ and define $X = X_1 \times \mathbb{F}_p$. For each $(a, b, c) \in \mathbb{F}_p \times \mathbb{F}_p \times \mathbb{F}_p$ define

$$A_{a,b,c} = \{(x, ax^2 + bx + c) : x \in X_1\}$$

and let $\mathcal{A} = \{A_{a,b,c} : (a, b, c) \in \mathbb{F}_p \times \mathbb{F}_p \times \mathbb{F}_p\}$.

Choose a set of keys $\mathcal{K} = \{K_{a,b} : (a, b) \in X\}$. For each node $n_{a,b,c} \in \mathcal{N}$, give node $n_{a,b,c}$ the set of keys $\{K_{x,y} : (x, y) \in A_{a,b,c}\}$.

Like the linear scheme, each node in the quadratic scheme is indexed using a tuple. Similarly, as the name implies, each node is issued keys according to a quadratic polynomial, rather than a straight line as in the linear scheme. Any three distinct keys from different groups are held by exactly one node in the network, and each node may share either one or two keys with a neighbor. For this reason, the underlying design of the quadratic scheme *is not* a configuration, however, it *is* a transversal design.

Theorem 3. *The design constructed in Scheme 8 is a $TD(3, k, p)$.*

Proof. As in the proof of Theorem 2, let $x \in X_1$, define $H_x = x \times \mathbb{F}_p$, and define $\mathcal{H} = \{H_x : x \in X_1\}$. Then $|X| = kp$ and $\mathcal{H}$ is a partition of X into k groups of size p. For a given $x \in X_1$ there is exactly one point $(x, y) \in A_{a,b,c}$ such that $(x, y) = (x, ax^2 + bx + c)$. Therefore, $|A_{a,b,c} \cap H_x| = 1$. It remains to be seen that any triple of points occurs in exactly one block.

Let (x_1, y_1), (x_2, y_2), and (x_3, y_3) be points such that x_1, x_2, and x_3 are distinct. If these three points occur in the same block, then there exists $a, b, c \in \mathbb{F}_p$ such that

$$\begin{aligned} y_1 &= ax_1^2 + bx_1 + c \\ y_2 &= ax_2^2 + bx_2 + c \\ y_3 &= ax_3^2 + bx_3 + c. \end{aligned}$$

In $\mathbb{F}_p$, these equations have a unique solution. Therefore, any triple of points occurs in exactly one block and the design is a $TD(3, k, p)$. □

It follows that the quadratic scheme is simply the Generic Combinatorial KPS based on a concrete construction of a $TD(3, k, p)$. Since the quadratic scheme is not a configuration, the previously established equations for Pr_1 and fail(1) do not apply. The following theorem aids in establishing performance metrics.

Theorem 4. *(Lee and Stinson [61, Theorem 5.2]) Suppose $(X, \mathcal{H}, \mathcal{A})$ is a $TD(3, k, p)$. Then every point $x \in X$ occurs in exactly p^2 blocks, and every pair of points from different groups occurs in exactly p blocks. Furthermore, any block $A \in \mathcal{A}$ intersects exactly $a_1 = k(p-1)(p-k+2)$ blocks in one point, exactly $a_2 = \binom{k}{2}(p-1)$ blocks in two points, and is disjoint from exactly $a_0 = p^3 - 1 - a_1 - a_2$ blocks.*

From Theorem 4, we see that each block in the design intersects a_2 other blocks at exactly two points. The probability that a pair of nodes share two keys is

$$Pr_1 = \frac{a_2}{p^3 - 1} = \frac{\binom{k}{2}(p-1)}{p^3 - 1} = \frac{k(k-1)}{2(p^2 + p + 1)}. \tag{2.6}$$

From Theorem 4 we also have that any pair of points from different groups occur in exactly p blocks. If a given pair of points secures a link between two nodes, then $p-2$ of the p^3-2 other blocks in the network contain both of these keys, allowing us to compute

$$\text{fail}(1) = \frac{p-2}{p^3-2}. \tag{2.7}$$

A more general estimate of resilience is given by Lee and Stinson [61]. The probability that a link is remains secure after s node compromises is the probability one or both keys from a pair remain secure. This probability can be estimated by counting the number of ways of choosing s nodes that do not contain the first key plus the ways of choosing s nodes that do not contain the second key, and subtracting the overlap. A given key is contained in p^2 blocks, and the number of blocks containing one or both of a pair of keys is $2p^2 - p$, so

$$\begin{aligned} \text{fail}(s) &= 1 - \frac{2\binom{p^3-p^2}{s} - \binom{p^3-2p^2+p}{s}}{\binom{p^3-2}{s}} && (2.8) \\ &\approx 1 - 2\left(1 - \frac{p^2-2}{p^3-2}\right)^s + \left(1 - \frac{2p^2-p-2}{p^3-2}\right)^s, && (2.9) \end{aligned}$$

Let A_1 be a block with points distributed according to $f(x)$ and A_2 be a block with points distributed according to $g(x)$. Shared-key discovery is performed by computing the points where $f(x) = g(x)$, or equivalently, the roots of the polynomial $f(x) - g(x) = 0$. If two such roots r_1 and r_2 exist, and $r_1, r_2 \in X_1$, then $(r_1, f(r_1) = g(r_1))$ and $(r_2, f(r_2) = g(r_2))$ are the common points in each block, and are labels for the shared keys in the quadratic KPS.

In general, the quadratic scheme provides a slightly lower degree of connectivity when two keys are required to form a secure link, but achieves a higher degree of resilience when only a small number of nodes are compromised, as both keys must revealed to an adversary to compromise a link. The quadratic scheme also incurs a higher computation cost, as nodes must find the roots of a quadratic polynomial in order to determine shared keys. Table 2.2 demonstrates the effect on resilience as more nodes are compromised, and can be compared directly to Table 2.1 in the previous section.

Table 2.2: The effect of node compromise on resilience in the quadratic scheme.

$p = 17$ (approx. 5000 nodes)	
s	fail(s)
1	0.0031
2	0.0122
3	0.0265
4	0.0448
5	0.0665
10	0.2034
20	0.4894

$p = 29$ (approx. 24000 nodes)	
s	fail(s)
1	0.0011
2	0.0044
3	0.00979
4	0.0168
5	0.0255
10	0.0868
20	0.2531

2.5.1 Higher Degree Constructions

The construction of the linear and quadratic scheme generalizes in the obvious way to higher degree polynomials. The underlying design is a $TD(t, k, p)$ and keys are blocks computed using a degree $t - 1$ polynomial. Higher strength designs allow for a greater degree of resilience, as nodes may have $\eta = 1, 2, \ldots, t - 1$ keys in common, but with the drawback of more complicated performance metrics, and a greater computational cost to perform key discovery. For completeness, the general construction appears below; however, attention will be limited to the linear and quadratic schemes for practical reasons. The next chapter demonstrates that the linear and quadratic schemes can be made flexible enough to apply to nearly any practical application.

Scheme 9. ***General Transversal KPS.*** *Fix values t and k and let p be a prime power such that $t \leq k \leq p$. Then a t-(v, b, r, k)-LS-KPS can be constructed for $v = kp$, $b = p^t$, and $r = p$. Given a network $\mathcal{N} = \{n_c : c \in (\mathbb{F}_p)^t\}$, construct a $TD(t, k, p)$ $(X, \mathcal{A})$ as follows:*

Let $X_1 \subseteq \mathbb{F}_p$ with $|X_1| = k$ and define $X = X_1 \times \mathbb{F}_p$. For each $c = \{c_0, c_1, \ldots, c_{t-1}\} \in (\mathbb{F}_p)^t$ define

$$A_c = \left\{ \left(x, \sum_{i=0}^{t-1} c_i x^i \right) : x \in X_1 \right\}$$

and let $\mathcal{A} = \{A_c : c \in (\mathbb{F}_p)^t\}$.

Choose a set of keys $\mathcal{K} = \{K_{a,b} : (a, b) \in X\}$. For each node $n_c \in \mathcal{N}$, give node n_c the set of keys $\{K_{x,y} : (x, y) \in A_c\}$.

As with the early transversal design-based schemes, this is simply the Generic Combinatorial Design KPS on a specific construction of a $TD(t, k, p)$. Shared-key discovery between two nodes is performed by evaluating their polynomials at each point intersection point. If at least $1 \leq \eta \leq t - 1$ keys are available, a secure link can be formed.

2.6 A Useful Generalization

Thus far we have considered a hierarchy of combinatorial designs by iteratively introducing necessary properties to a basic set system. Configurations are a popular choice for generic KPS constructions because of several nice properties:

1. Uniformity ensures that each node has the same number of keys,
2. regularity ensures that keys are evenly distributed across the nodes, and,
3. pairs of points occurring in at most one block allows for the maximal number of connections in the network.

The structure of the design also allows for simple formulas to express the connectivity and resilience of the KPS. In order to efficiently construct such a KPS, the linear and quadratic schemes utilized a subset of configurations called transversal designs. In the linear case, because a TD$(2, k, p)$ is a configuration, the formulas for performance metrics still hold. Analysis of the quadratic scheme is more complicated, as the underlying design is not a configuration, and the block intersection properties of the design require more detail to characterize.

Paterson and Stinson [84] have observed that a variety of different combinatorial constructions have appeared in the literature for use as KPSs. They propose a generalization that encapsulates nearly all of them and derive performance metrics for this general class of designs. The schemes proposed in the next chapter will utilize their approach when analyzing performance.

Definition 5. ***Partially Balanced t-Design.*** *[84] Let v, k, t, be positive integers and let λ_i be a positive integer for $0 \leq i \leq t-1$. A t-$(v, k, \lambda_0, \ldots, \lambda_{t-1})$-partially balanced t-design (PBtD) is a pair $(X, \mathcal{A})$ that satisfies:*

1. *$|X| = v$ and $\mathcal{A}$ is a set of k-subsets of X.*
2. *There are exactly λ_0 blocks.*
3. *For $1 \leq i \leq t-1$, every i-subset of points occurs in either 0 or λ_i blocks.*
4. *For $t \leq i \leq k$, every i-subset of points occurs in either 0 or 1 blocks.*

It follows from this definition that the number of blocks is $b = \lambda_0$, and, if it assumed that every point occurs in at least 1 block, then the design is also of degree $r = \lambda_1$.

It is easy to see that a transversal design is also a PBtD. It holds that in a TD(t, k, p), any set of $0 \leq i \leq t$ points from different groups occur in exactly n^{t-i} blocks. In the case of the linear scheme, we have a TD$(2, k, p)$ where each pair of points from different groups occurs in exactly 1 block. In the case of the quadratic scheme we have that each pair of points from different groups occurs in exactly p blocks, and each triple of points occurs in exactly 1 block. It follows that a TD$(2, k, p)$ is a 2-(kp, k, p^2, p)-PBtD, and a TD$(3, k, p)$ is a 3-(kp, k, p^3, p^2, p)-PBtD.

The values λ_i characterize the block intersection properties of the design. In particular, we are interested in cases when the underlying design is of strength $t = 2, 3$. Two blocks $\{A_1, A_2\} \in \mathcal{A}$ form a *link* if $|A_1 \cap A_2| \geq \eta$. In the linear case, $\eta = 1$, and in the quadratic case we may consider $\eta = 1$ or $\eta = 2$, depending on the application.

If $|A_1 \cap A_2| = i \geq \eta$, then the link is referred to as an *i-link*. A block A is said to *break* a link if $A \notin \{A_1, A_2\}$ and $|A_1 \cap A_2| \subseteq A$. In other words, A breaks the link between A_1

and A_2 if A contains all points shared by A_1 and A_2. For a fixed block A, let α_i denote the number of i-links that block A is contained in, and let β_i denote the number of i-links that block A breaks. In the linear case, we define

$$\alpha = \alpha_1$$

$$\beta = \beta_1,$$

while in the quadratic case we define

$$\alpha = \alpha_1 + \alpha_2$$

$$\beta = \beta_1 + \beta_2.$$

Similarly, let L_i be the total number of i-links and define $L = L_1$ in the linear case and $L = L_1 + L_2$ in the quadratic case. Finally, let B be a block and C a subset of B of size i and define

$$\mu'(i) = |\{A \in \mathcal{A} \,:\, A \cap B = C\}|.$$

Then, for a given set of i points in B, $\mu'(i)$ represents the number of other blocks in the design that also contain those i points. A recursive formula for $\mu'(i)$ is given by Paterson and Stinson [84]. The relevant value for the linear scheme is

$$\mu'(1) = p - 1,$$

and for the quadratic scheme,

$$\begin{aligned} \mu'(2) &= p - 1 \\ \mu'(1) &= p^2 - 1 - (k-1)(p-1). \end{aligned}$$

We can now state the following results:

Lemma 1. *(Paterson and Stinson [84, Lemma 4.1]) For $\eta \leq i \leq t-1$, it holds that*

$$\alpha_i = \binom{k}{i}\mu'(i).$$

Lemma 2. *(Paterson and Stinson [84, Lemma 4.2]) For $\eta \leq i \leq t-1$, it holds that*

$$\beta_i = \mu'(i)\left(\frac{\lambda_i}{2} - i\right)\binom{k}{i}.$$

Lemma 3. *(Paterson and Stinson [84, Lemma 4.3]) For $\eta \leq i \leq t-1$ it holds that*

$$\begin{aligned} L_i &= \frac{b\alpha_i}{2} \\ L &= \frac{b\alpha}{2}. \end{aligned}$$

Theorem 5. *(Paterson and Stinson [84, Theorem 4.8]) It holds that*

$$Pr_1 = \frac{\alpha}{b-1}.$$

Theorem 6. *(Paterson and Stinson [84, Theorem 4.7]) It holds that*

$$\mathit{fail}(1) = \frac{\beta}{L-\alpha}.$$

Applying the formulas above to a TD$(2, k, p)$ we have:

$$\begin{aligned} (v, b, r, k) &= (kp, p^2p, k) \\ \mu'(1) &= p-1 \\ \alpha_1 = \alpha &= k(p-1) \\ \beta_1 = \beta &= k(p-1)\left(\frac{p}{2}-1\right) \\ L_1 = L &= kp^2(p-1) \\ Pr_1 &= \frac{k}{p+1} \\ \text{fail}(1) &= \frac{p-2}{p^2-2}. \end{aligned}$$

The formulas for Pr_1 and fail(1) are identical to Equations 2.1 and 2.2. The analysis is slightly more interesting for a TD$(3, k, p)$, as we can examine the performance for $\eta = 1, 2$. Equations 2.6 and 2.7 presented previously only considered the $\eta = 2$ case.

$$\begin{aligned} (v, b, r, k) &= (kp, p^3, p^2, k) \\ \mu'(2) &= p-1 \\ \mu'(1) &= (p^2-1) - (k-1)(p-1) \\ \alpha_1 &= k(p-1)(p-k+2) \end{aligned}$$

$$\begin{aligned}
\alpha_2 &= \binom{k}{2}(p-1) \\
\beta_1 &= \frac{k(p-1)(p-k+2)(p^2-2)}{2} \\
\beta_2 &= \binom{k}{2}(p-1)\frac{p-2}{2} = \binom{k}{2}\binom{p-1}{2} \\
L_1 &= \frac{p^3 k(p-1)(p-k+2)}{2} \\
L_2 &= \frac{p^3\binom{k}{2}(p-1)}{2}.
\end{aligned}$$

If we fix $\eta = 2$ then we obtain the same results as Equations 2.6 and 2.7:

$$Pr_1 = \frac{\alpha}{b-1} = \frac{\binom{k}{2}(p-1)}{p^3-1} = \frac{k(k-1)}{2(p^2+p+1)}$$

$$\text{fail}(1) = \frac{\beta}{L-\alpha} = \frac{\binom{k}{2}\binom{p-1}{2}}{\frac{p^3\binom{k}{2}(p-1)}{2} - \binom{k}{2}(p-1)} = \frac{p-2}{p^3-2}.$$

Next, fix $\eta = 1$. Then

$$\begin{aligned}
\alpha &= k(p-1)(p-k+2) + \binom{k}{2}(p-1) \\
&= k(p-1)\frac{(2p-k+3)}{2}.
\end{aligned}$$

$$\beta = \frac{k(p-1)(p-k+2)(p^2-2)}{2} + \binom{k}{2}\binom{p-1}{2},$$

and

$$L = \frac{p^3 k(p-1)(p-k+2)}{2} + \frac{p^3\binom{k}{2}(p-1)}{2}.$$

Thus, we can compute

$$Pr_1 = \frac{\alpha}{b-1} = \frac{k(2p-k+3)}{2(p^2+p+1)}$$

and

$$\text{fail}(1) = \frac{\beta}{L-\alpha} = \frac{2p^3+(4-2k)p^2+(k-5)p+2k-6}{(2p-k+3)(p^3-2)}.$$

When $\eta = 1$, the formulas for Pr_1 and fail(1) are much more complicated than the $\eta = 2$ case. Nevertheless, it is still useful to have a precise measure of performance.

2.7 Multiple Space Schemes

Recall that the value ρ was introduced to demonstrate the trade-off between connectivity and resilience in a KPS. For configuration-based schemes, it was shown in Section 2.2.2 that $\rho \approx k$. In other words, Pr_1 cannot be greater than approximately $k \cdot \text{fail}(1)$. The linear scheme is a configuration-based scheme, and therefore subject to this constraint. The quadratic scheme *is not* a configuration-based scheme, and achieves a better ratio between connectivity and resilience, as shown in Table 2.2. This improvement follows from the fact that when $\eta = 2$, a given link is only broken when two specific keys are compromised, which occurs with much lower probability than just a single key being compromised.

Lee and Stinson [61] observe that the resilience of the LS-KPS can be increased by nesting multiple KPSs inside of each other. Rather than using a linear scheme to distribute k keys to each node, it can be used to distribute membership in k different KPSs. In order to compromise a single key, an attacker must compromise both the linear scheme, and the secondary scheme, which would typically require the compromise of multiple additional nodes. A good candidate for the secondary scheme is a *Blom Scheme* [11].

In Blom's key pre-distribution scheme, a symmetric bivariate polynomial $f(x, y)$ is used to define the key shared by a pair of nodes. Each node is given a single partial evaluation of this polynomial (i.e, a univariate polynomial), and the degree of this polynomial determines the resilience of the system. In the simplest case, knowledge of two points is sufficient to reconstruct the original polynomial. Therefore, an adversary that learns the key information from two nodes in a Blom scheme can compute all keys in the system.

Scheme 10. ***Blom's Scheme.*** *Let q be a prime number, and let $\mathcal{N} = \{n_1, \ldots, n_N\}$ be a network of $N < q$ nodes.*

1. *For each $n_i \in \mathcal{N}$, choose a unique element $u_i \in \mathbb{Z}_q$. The value q and each u_i are made public.*

2. *Choose random values $a, b, c \in \mathbb{Z}_q$ and define*

$$f(x, y) = a + b(x + y) + cxy.$$

3. *For each u_i, compute*

$$g_i(x) = f(x, u_i) \mod q$$

and securely transmit this value to n_i as its secret key.

4. *Nodes n_i and n_j can compute their shared key $K_{i,j} = K_{j,i}$, by computing one of the two following values:*

$$K_{i,j} = g_i(u_j) \textit{ or}$$

$$K_{j,i} = g_j(u_i).$$

The scheme generalizes naturally to higher degree symmetric bivariate polynomials, which introduces a straightforward increase in storage cost as the degree of $f(x, u_i)$ increases, and in computational cost when evaluating $g_i(u_j)$ for the same reason, but allows for a larger number of node compromises before the scheme is broken. Blom's scheme is also well-defined if q is a prime power, with computations taking place in $\mathbb{F}_q$.

When the linear scheme is used to distribute membership in a set of Blom Schemes, it is referred to as a *Multiple Space KPS (MS-KPS)*. The storage requirement of each node is doubled, and the cost of key computation is increased by a constant number of additions and multiplications to compute the underlying key from the Blom scheme. The benefit to such an approach is a significant increase in resilience to key compromise. As shown by Lee and Stinson [61], the multiple space LS-KPS using the simple variant of Blom's Scheme decreases the probability of a compromised affecting other links in the network by an order of magnitude, and often more when only a small fraction of nodes are compromised. Note that Pr_1 is unaffected by switching to the MS-KPS, as any two nodes that share a point in the outer linear scheme are always able to derive a shared key in the inner Blom scheme.

To instantiate the MS-KPS, one would create kp instances of a Blom scheme, one for each point in the underlying transversal design of the linear scheme. Each of these Blom schemes must be large enough to accommodate p distinct nodes, as each point in the underlying TD occurs in p blocks. Each node is then issued membership in k of these Blom schemes according to its block A in the TD. For each point in A, the node is given a unique ID u_A in the Blom scheme associated with this point (which can be derived from it's label in the linear scheme), and is issued an evaluation of $f(x, u_A)$ as described above. If two distinct nodes share a key in the underlying TD, then they use their shared point to determine which Blom scheme to use to compute their shared key.

Despite the fact that no two nodes ever have more than one key in common, the MS-KPS can achieve very high levels of connectivity and resilience. This fact is demonstrated in Table 2.3. Setting $k = 50$ and $p = 149$, the left table shows the probability that a one- or two-hop path exists between two nodes n_i and n_j when C nodes lie in their common neighborhood (i.e., are candidates to route messages between them), and the right table shows the probability that the link between them is compromised given that the adversary has learned keys from s nodes. In the left table, $Pr_1 = \frac{1}{3}$ for all values of C.

Table 2.3: The two-hop connectivity (Pr_2) and resilience of the linear LS-KPS and MS-KPS for $k = 50$ and $p = 149$. The value $Pr_1 = \frac{1}{3}$ is constant for all values of C.

C	Pr_2	$Pr_1 + Pr_2$
1	0.074	0.407
2	0.139	0.472
3	0.197	0.530
4	0.249	0.582
5	0.295	0.628
10	0.460	0.793
15	0.551	0.885
20	0.602	0.936

s	LS-KPS	MS-KPS
2	0.0132	0.0001
5	0.0327	0.0017
10	0.0642	0.0073
20	0.1244	0.0281
30	0.1807	0.0590
40	0.2333	0.0972
50	0.2827	0.1404

The idea of nesting multiple KPSs to boost resilience can be applied to virtually any KPS setting, including the new schemes presented in the next chapter. The Blom scheme is an ideal candidate, especially in conjunction with the linear scheme, as it boosts the resilience of the network significantly, without incurring much computational overhead. Martin [73] summarizes several combinatorial approaches, including the MS-KPS considered here, and their applicability to constructing "nested" or "layered" KPSs. He concludes that this is a promising direction for providing a greater degree of flexibility in combinatorial KPSs. The next chapter also addresses the problem of flexibility in combinatorial KPSs, using a different approach, but the resulting schemes are good candidates for use in a nested KPS, much like the basic linear scheme considered here.

2.8 Summary and Remarks

This chapter presented an overview of combinatorial designs and established their use for key pre-distribution. Of these designs, transversal designs were presented in detail, as they are particularly useful for sensor network key pre-distribution. Transversal designs can be constructed efficiently, and allow for efficient neighbor discovery when the strength of the design is low, which led to the definition of the linear scheme, and the quadratic scheme, corresponding to transversal designs of strength two and three, respectively.

Paterson and Stinson observed that a number of combinatorial KPSs, including the transversal design-based schemes considered here, shared common properties, and introduced partially balanced t-designs as a means of unifying the analysis of each of these approaches. The analysis of PBtDs led to a thorough analysis of the quadratic scheme with intersection threshold $\eta = 1$. The idea of nesting multiple KPSs to build so-called multiple space schemes was also considered.

The next chapter presents some new families of combinatorial KPSs based on the linear and quadratic schemes considered here, and utilizes the theory of partially balanced t-designs to analyze their performance. The goal of these new schemes is to address a fundamental limitation in many combinatorial design-based KPSs. Namely, that such designs only exist for restricted sets of parameters, which places undesirable constraints on the size of the network.

Chapter 3

Constructing Combinatorial KPSs With Flexible Network Parameters

By nature, sensor networks can be considered fragile in several ways. Battery-powered sensors will eventually fail when their battery dies, potentially leading to a loss of connectivity in the network. When adversarial settings are considered, nodes are expected to be compromised by an adversary and could be taken offline, or logically removed from the network by neighbors when caught misbehaving. The random nature of deployment may reveal that certain regions do not have sufficient sensor coverage once the network is up and running. To combat this, it may be necessary to prolong the life of a network by adding additional nodes at a later time to compensate for nodes that are no longer available.

The key pre-distribution schemes considered in the previous chapter have nice theoretical properties, but may not be perfectly suited to actual sensor network needs. This problem is elaborated on in the next section. In the remainder of this chapter we consider techniques for adding flexibility to a family of combinatorial key pre-distribution schemes that allow them to be used in a wide variety of deployment scenarios. In particular, constructions are provided that can accommodate a much wider range of network size and storage requirements than the schemes in the previous chapter, including the ability to deterministically add nodes to the network while still retaining the precise theoretical analysis and benefits of using a deterministic combinatorial-based approach. Performance metrics for these new approaches are derived, and a generalized technique for computing performance metrics for less structured designs is presented. We conclude with a performance analysis that compares the approaches considered for a few concrete sets of parameters, and demonstrate that the more flexible techniques considered here are able to match, or nearly match, the performance of the schemes considered in the previous chapter.

This chapter contains joint work with Maura Paterson, and has been published previously [47].

3.1 Problems with Parameter Choice

The linear and quadratic schemes presented in the previous chapter are able to achieve a reasonable balance between connectivity and resilience, along with providing simple formulas to compute or estimate these values for any valid parameter set. For a fixed network size, (determined by p), the parameter k, or key storage, can be controlled to achieve a variety of connectivity/resilience bounds. Despite this, transversal design-based schemes have been criticized [12] due to restrictions on parameter choices.

Our constructions of transversal design-based schemes are defined by three parameters, (t, k, p), and, in the case of linear and quadratic schemes, t is fixed and only p and k are chosen. From this, a (v, b, r, k)-LS-KPS can be constructed where

$$\begin{aligned} v &= kp && \text{(total keys)} \\ b &= p^t && \text{(total nodes)} \\ r &= p^{t-1} && \text{(nodes per key)} \\ k &= k && \text{(keys per node)} \end{aligned}$$

and $t \leq k \leq p$. These parameters imply that the size of the network must be p^2 or p^3, where p is a prime or prime power. Furthermore, k is bounded above by p, so a small value of p forces the maximal number of keys to be small as well. The formulas for Pr_1 and fail(s) for the linear and quadratic schemes were computed based on the assumption that all blocks in the underlying design were present in the network.

Consider a network of 1000 nodes. If we wish to use the linear scheme to distribute keys, then $p = 31$ only accommodates a network size of $b = 961$ nodes, while $p = 37$ accommodates $b = 1367$ nodes. If we attempt to use a quadratic scheme, the best choice is $p = 11$, which gives us network size of $b = 1331$, with the added constraint that each node can have a maximum of $k = 11$ keys. The connectivity of such a system may not be high enough for some applications.

These properties have been criticized in the literature for being too restrictive, and the solution of simply choosing larger parameters and deploying a subset of the network was dismissed with the claim that network performance becomes unpredictable [12]. Alternate schemes have been proposed; however, unlike the straightforward parameter choice of a TD-based KPS, the alternate suggestions are much more complicated. For example, parameter selection for the schemes of Bose et al. [12] is as follows:

1. $b = af(2g+1)$, $k = (a-1)f + g$, $v = \binom{a}{2}f^2 + \frac{1}{3}(2g+1)g$, where a, $f(\geq 2)$ are any integers such that $g \geq 3$ satisfies $g = 0$ or $1 \mod 3$.
2. $b = \binom{m}{2}(2g+1)$, $k = \binom{m-2}{2} + g$, $v = 3\binom{m}{4} + \frac{1}{3}(2g+1)g$, where $m(\geq 4)$ is any integer and g is as above.
3. $b = p^2(2g+1)$, $k = \tilde{k} + g$, $v = \tilde{k}p + \frac{1}{3}(2g+1)g$, where $p(\geq 3)$ and $\tilde{k}(< p+1)$ are integers such that $\tilde{k} - 2$ mutually orthogonal Latin squares of order p exist, and g is as above.

While these schemes certainly provide a wide variety of parameter choices, the complexity of the constructions makes choosing a concrete set of parameters for a fixed network size or storage requirement difficult.

There is clear motivation for KPSs that allow for straightforward parameter selection, as the linear and quadratic schemes provide, but that also allow for a great degree of flexibility in choosing network size and storage requirements, as provided by Bose et al. [12]. To this end, the remainder of this chapter is dedicated to describing a new family of KPSs based on the linear and quadratic schemes that can be used in a much wider variety of applications, while still retaining straightforward parameter choice and performance metric evaluation. Furthermore, we demonstrate that randomly deploying a subset of the linear and quadratic schemes does not lead to any significant loss of connectivity or resilience, thereby allowing the linear and quadratic schemes to be utilized for networks of arbitrary size without introducing any additional complexity.

3.2 Approaches to Varying Network Size

The analysis of the connectivity and resilience of transversal design-based KPSs considered thus far has relied on the assumption that all blocks of the design are present in the network. In practice, it is usually expected that a small number of nodes will lose power, be compromised, or otherwise fail during deployment and operation. Therefore, even if the parameters of the KPS perfectly match the desired network, it may be impossible to make exact claims about the performance of the network under a realistic operating scenario. We now consider two approaches that can be used to address this problem, and by extension, allow for the schemes considered in the previous chapter to be used for a wide array of applications without sacrificing performance.

3.2.1 Randomized Subset Schemes

We can establish a family of key pre-distribution schemes based on selecting a random subset of a design with the idea that a sufficiently large random subset of nodes will still retain the performance of the entire network. Such a result is expected due to the regularity and uniformity of the underlying design used to distribute keys. Randomized subset schemes are formally presented in this section, and are evaluated with respect to other approaches in Section 3.4

Scheme 11. ***Random Subset KPS.*** *Let $(X, \mathcal{A})$ be a combinatorial KPS for a network of N nodes. For a network $\mathcal{N}$ of $m < N$ nodes, construct a new KPS by selecting m distinct blocks from $\mathcal{A}$ uniformly at random.*

This definition can be applied specifically to the linear and quadratic schemes presented earlier.

Scheme 12. ***Random Linear KPS.*** *Given a network of size m, fix the value k and let p be the smallest prime power such that $2 \leq k \leq p$ and $m < p^2$. Construct a linear LS-KPS as in Scheme 7, and choose m distinct blocks uniformly at random.*

Scheme 13. ***Random Quadratic KPS.*** *Given a network of size m, fix the value k and let p be the smallest prime power such that $3 \leq k \leq p$ and $m < p^3$. Construct a quadratic LS-KPS as in Scheme 8, and choose m distinct blocks uniformly at random.*

Although the existing formulas for connectivity and resilience do not apply to random subset schemes, we expect the performance of random subset-based schemes to match the performance with exceptionally high probability as long as the network contains a reasonably large proportion of the original KPS. We demonstrate later in this chapter that even a very small subset of the original KPS is chosen, the performance metrics are virtually identical with very high probability. Therefore, while the definitions for the random linear and quadratic KPSs suggest choosing p to be the smallest prime or prime power that will accommodate the network, this is not strictly necessary. Parameters can be chosen optimistically to, for example, allow additional nodes to be added to the network in the future.

On top of the lack of concrete analysis, random subset schemes do suffer from some of the drawbacks of the randomized KPS schemes considered earlier (Schemes 4 and 5). Namely, a good source of randomness is needed, and additional bookkeeping is necessary to keep track of which blocks are included in the network.

3.2.2 Resolvable and Decomposable Designs

The previous section discussed choosing subsets of a design at random in order to build a KPS for a wider variety of network sizes. Rather than relying on random subsets, we can utilize some additional structure of certain combinatorial designs in order to deterministically select a subset of the network in such a way that a concrete analysis is still possible.

Definition 6. *A design $(X, \mathcal{A})$ is said to be resolvable if there exists a partition R of its set of blocks $\mathcal{A}$ into parallel classes, each of which in turn partitions the set X. The partition R is called a resolution.*

Consider a $\mathrm{TD}(2, k, p) = (X, \mathcal{H}, \mathcal{A})$. Definition 6 states that such a design is resolvable if it possible to partition the blocks of the system into distinct sets $B_1, B_2, \ldots B_p$ such that each $x \in X$ belongs to precisely one block in each B_i. Each B_i is known as a *parallel class*, and the set $R = \{B_1, \ldots, B_p\}$ is a *resolution* of the design. Figure 3.1 demonstrates this for a $\mathrm{TD}(2, 5, 5)$. Note that, because each parallel class partitions the set of points X, no two blocks in the same parallel class intersect.

The resolution of a $\mathrm{TD}(2, k, p)$ is useful because it allows for the selection of subsets of the design in such a way that there is still a regular and uniform structure to each subset. Each parallel class of a $\mathrm{TD}(2, k, p)$ is, in fact, a $\mathrm{TD}(1, k, p)$, and can be analyzed using any general result for transversal designs. The ability to decompose a $\mathrm{TD}(2, k, p)$ into p copies of a lower degree design is not dependent on the value $t = 2$. For example, the underlying $\mathrm{TD}(3, k, p)$ used in Scheme 8 can be partitioned into p copies of a $\mathrm{TD}(2, k, p)$, each of which can be further decomposed into p copies of a $\mathrm{TD}(1, k, p)$, or parallel classes. This property holds in general for the family transversal designs established earlier. The blocks of a $\mathrm{TD}(t, k, p)$ can be partitioned into disjoint sets, each of which is a $\mathrm{TD}(t - 1, k, p)$.

A necessary property in the linear case is that each B_i in the resolution partitions the set X, which implies that no two blocks in a given parallel class intersect. The resolution of a $\mathrm{TD}(3, k, p)$ into p copies of a $\mathrm{TD}(2, k, p)$ does not have this property, as multiple blocks in each B_i will intersect. Despite this, the ability to partition a higher degree transversal design into p copies of a lower degree design is extremely useful. For higher degree designs,

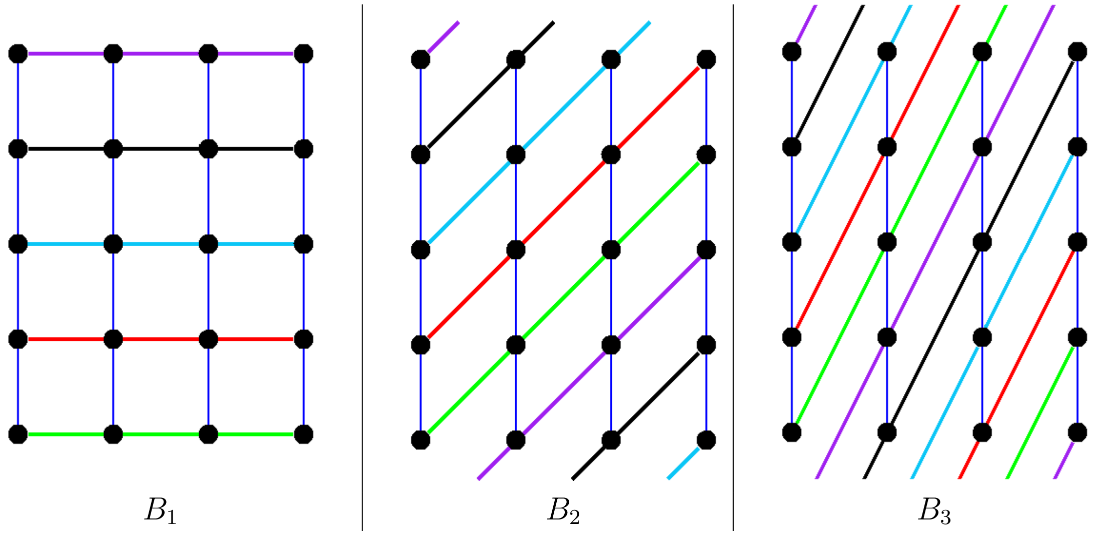


Figure 3.1: (Previously presented as Figure 2.4.) A partial resolution of a TD$(2, 5, 5)$. In each parallel class, it is easy to see that each point is contained in exactly one block (or line in the diagram). The parallel classes form a partition of the design, and the blocks within each parallel class form a partition of the points within the design. The two parallel classes not shown consist of all lines of slope $a = 3$ and all lines of slope $a = 4$.

we refer to this process as a *decomposition*. Although each B_i in the resolution of a higher degree design is not necessarily a partition of X, and therefore not necessarily a parallel class, a similar property does exist. Namely, although two blocks of a TD(t, k, p) may intersect in $t - 1$ points, no two blocks in the same B_i intersect at $t - 1$ blocks. This property follows from the fact that each B_i is a TD$(t - 1, k, p)$.

3.2.3 Decomposable Schemes

In the same manner that the random subset KPS builds a smaller KPS from a larger one using random subsets, we can define a family of KPSs based on decomposable designs which adjust the network size by choosing a subset of the decomposition.

Scheme 14. ***Decomposable KPS.*** *Let $(X, \mathcal{A})$ be a combinatorial KPS for a network of N nodes, where $(X, \mathcal{A})$ is a decomposable design with resolution $B_1, \ldots, B_p$. For a network $\mathcal{N}$ of $m < N$ nodes, construct a new KPS by selecting all blocks in $B_1, \ldots, B_j$ where $|B_1 \cup \ldots \cup B_j| = m$.*

The decomposable KPS deterministically chooses a subset of a larger KPS by adding groups one at a time until the desired network size is met. Note that this still imposes a constraint on the size of the network. For example, the TD$(2,5,5)$ considered in Figure 3.1 contains 25 blocks, but the decomposable KPS can be used for networks of size $m = 5, 10, 15, 20, 25$.

The transversal designs used to construct the linear and quadratic schemes are also decomposable designs. Therefore, we can construct decomposable versions of the corresponding KPSs. We begin by demonstrating that underlying designs are decomposable.

Theorem 7. *The TD$(2,k,p)$ constructed in Scheme 7 is a resolvable design with resolution $R = \{B_1, \ldots, B_p\}$, where $B_i = \{A_{i,j} \mid j \in \mathbb{F}_p\}$. Furthermore, each B_i is a TD$(1,k,p)$.*

Proof. R is clearly a partition of $\mathcal{A}$. Fix B_i and assume there exists a point (x, y) contained in two distinct blocks $A_{i,j}$ and $A_{i,h}$ in B_i. This implies $y = ix + j$ and $y = ix + h$ in $\mathbb{F}_p$, which implies $i = h$, contradicting the fact that $A_{i,j}$ and $A_{i,h}$ are distinct. Therefore, B_i is a partition of X. □

Theorem 8. *The TD$(3,k,p)$ constructed in Scheme 8 is a decomposable design with resolution $R = \{B_1, \ldots, B_p\}$, where $B_i = \{A_{i,j,k} \mid j,k \in \mathbb{F}_p\}$. Furthermore, each (X, B_i) is a TD$(2,k,p)$.*

Proof. We show that each B_a is a TD$(2,k,p)$. In particular, we show that, using the same groups as the TD$(3,k,p)$, any pair of points occurs in at most one block. If this is not the case, then there exist two points (x_1, y_1) and (x_2, y_2) with $x_1 \neq x_2$ that occur in two blocks in B_a, which implies that there exists b, b', c, c' with $b \neq b'$ and $c \neq c'$ such that

$$\begin{aligned} y_1 &= ax_1^2 + bx_1 + c \\ y_1 &= ax_1^2 + b'x_1 + c', \end{aligned}$$

and

$$\begin{aligned} y_2 &= ax_2^2 + bx_2 + c \\ y_2 &= ax_2^2 + b'x_2 + c'. \end{aligned}$$

Therefore,

$$\begin{aligned} bx_1 + c - (b'x_1 + c') &= 0 \\ bx_2 + c - (b'x_2 + c') &= 0, \end{aligned}$$

and,

$$\begin{aligned}
bx_1 + c - (b'x_1 + c') &= bx_2 + c - (b'x_2 + c') \\
bx_1 - b'x_1 &= bx_2 - b'x_2 \\
bx_1 - bx_2 &= b'x_1 - b'x_2 \\
b(x_1 - x_2) &= b'(x_1 - x_2).
\end{aligned}$$

Because $x_1 \neq x_2$, we have $b = b'$ (and therefore $c = c'$), contradicting our assumption. Therefore, any pair of points in B_a occur in at most one block, and, since there are p^2 points, it follows that B_a is a TD$(2, k, p)$. □

The fact that each B_i in the resolution of the quadratic scheme is a TD$(2, k, p)$ is intuitive. Each block $A_{a,b,c}$ in the quadratic scheme is determined by a polynomial of the form $y = ax^2 + bx + c \in \mathbb{F}_p$. Each B_i fixes the value of a. Therefore, the blocks within a given B_i are all of the form $y - ax^2 = bx + c$, which is identical to the polynomial used to generate blocks in the linear scheme, shifted by a constant value.

Blocks within the linear and quadratic scheme are indexed using tuples from $(\mathbb{F}_p)^t$, and each B_i in the resolution is defined simply by fixing the first index in the tuple. If we restrict our attention to $\mathbb{Z}_p$ where p is prime, then this observation implies a simple and natural ordering on the decomposition of each design. In this case, we can refer to the first ℓ blocks of these designs without ambiguity.

Scheme 15. ***Parallel Class (Decomposable) Linear KPS.*** *Given a linear LS-KPS as in Scheme 7 using a TD$(2, k, p)$ and a network of size $m = \ell p$, construct a Decomposable Linear KPS by combining all blocks contained in the first ℓ parallel classes of the TD$(2, k, p)$.*

Scheme 16. ***Decomposable Quadratic KPS.*** *Given a quadratic LS-KPS as in Scheme 8 using a TD$(3, k, p)$ and a network of size $m = \ell p^2$, construct a Decomposable Quadratic KPS by combining all blocks contained in $B_1, \ldots, B_\ell$ in the resolution of the TD$(3, k, p)$.*

As with the random linear and quadratic schemes presented earlier, it is not strictly necessary to choose p to be the smallest prime power that satisfies the requirements. Instead, p can be chosen somewhat optimistically to allow for future network growth. However, unlike the random schemes, the fact that the smaller subnetwork is chosen based on the structure of the underlying design allows us to derive concrete performance metrics. Additionally, no randomness is required to select which blocks are included in the subnetwork, and no bookkeeping is necessary to keep track of which blocks have been assigned to nodes. Because the resolution is naturally ordered, the size of the subnetwork determines precisely which subset of blocks are in use. Although the decomposable schemes have several benefits over random schemes, they are less flexible with respect to network size. In their presented form, the decomposable schemes can only accommodate network sizes that are a multiple of p or p^2, as nodes must be added to the network one B_i at a time for the performance evaluation presented in the next section to be correct.

3.2.4 Finer Control Over Network Size

If a finer degree of control over network size is required, there are different approaches that can be considered. Theorem 16 established that the $\mathrm{TD}(3, k, p)$ used in the quadratic scheme is not only decomposable, but is decomposable into p instances of a $\mathrm{TD}(2, k, p)$, each of which are themselves decomposable. Therefore, one could utilize the decomposable quadratic scheme to select a subnetwork of size ℓp^2, consisting of $B_1, \ldots, B_\ell$ from the resolution, and then add $\ell' p$ additional nodes from $B_{\ell+1}$ using the linear parallel class scheme. In this manner, networks of size $p, 2p, \ldots, p^3$ can be accommodated, while still retaining the ability to derive precise performance metrics.

A second approach is to combine the decomposable schemes and random schemes. To reach a specific network size m, we choose ℓ such that $B_1, \ldots, B_\ell$ do not contain enough blocks, but $B_1, \ldots, B_{\ell+1}$contain too many blocks. We then use the random scheme to select $m - \ell p^{t-1}$ nodes from $B_{\ell+1}$. This approach balances the benefits and drawbacks of both schemes. The performance analysis applies only to blocks in $B_1, \ldots, B_\ell$, which are completely deployed, and randomness/bookkeeping are only necessary for the partially deployed blocks in $B_{\ell+1}$.

Although these and other approaches are possible, the remainder of this chapter is restricted to considering the random and decomposable schemes on their own. The performance results later in this chapter demonstrate that even relatively small subnetworks can still achieve good performance, which allows for significant flexibility when choosing p, which in turn allows for more flexibility in subnetwork size.

3.3 Deriving Performance Metrics

The previous sections in this chapter have built a family of key pre-distribution schemes based on transversal designs of strength 2 (linear) and 3 (quadratic) that allow for efficient construction, efficient key discovery, flexible parameter choice, and allow for thorough analysis of performance metrics. Weaknesses in these schemes were identified, and more flexible generalizations based on both random subsets and deterministic subsets, by exploiting the resolvability of the underlying transversal designs, were proposed. In this section we more closely examine the performance of these schemes, and demonstrate that the flexible generalizations are amenable to concrete analysis, much like the original schemes they were based on.

3.3.1 Analysis of the Decomposable Linear KPS

The decomposable linear KPS provides a simple method for deterministically selecting a subset of blocks from a $\mathrm{TD}(2, k, p)$. The underlying set system resulting from selecting $B_1, \ldots, B_\ell$ in the resolution of a $\mathrm{TD}(2, k, p)$ turns out to be a partially balanced t-design, allowing the formulas from Section 2.6 to be used for performance metrics.

Definition 7. *A $\overline{TD}(2, k, p, \ell)$ is a set system consisting of $1 \leq \ell \leq p$ parallel classes from a $TD(2, k, p)$.*

Theorem 9. *A $\overline{TD}(2, k, p, \ell)$ is a 2-$(kp, k, \ell p, \ell)$-PBtD.*

Proof. Given ℓ parallel classes of a $\mathrm{TD}(2, k, p)$, let $\mathcal{A}$ be the set of blocks contained within these classes, and let X be the set of points contained within these classes. Note that the set X contains the same kp points as the original $\mathrm{TD}(2, k, p)$, as every point in the original design is contained in each parallel class. The set $\mathcal{A}$ contains ℓp blocks, as each of the ℓ parallel classes contain p distinct blocks, and we have $\lambda_0 = \ell p$. Because each point occurs exactly once in each parallel class, we have that each point occurs in exactly $\lambda_1 = l$ blocks. Finally, each point in the original design occurs in at most 1 block, and therefore occurs in at most 1 block in $\mathcal{A}$. Therefore, all necessary properties from Definition 5 are satisfied. □

Recall from Section 2.6 that for a given point and block containing that point, $\mu'(1)$ denotes the number of other blocks in the design that contain that point. In this case, the symmetry of the underlying PBtD means that the value of $\mu'(1)$ is independent of the choice of point and block. The value α represents the number of links in the network that a given block is contained in, the value β represents the number of links that a given block can break, and the value L represents the total number of links in the network. Using the previously derived formulas yields:

$$\begin{aligned}
\mu'(1) &= \lambda_1 - 1 = \ell - 1, \\
\alpha &= k\mu'(1) = k(\ell - 1), \\
\beta &= \mu'(1)\left(\frac{\lambda_1}{2} - 1\right)k = (\ell - 1)\left(\frac{\ell}{2} - 1\right)k, \\
L &= \frac{b\alpha}{2} = \frac{\ell p k(\ell - 1)}{2}.
\end{aligned}$$

Applying Theorems 5 and 6 gives us

$$Pr_1 = \frac{\alpha}{b-1} = \frac{k(\ell-1)}{\ell p - 1}, \tag{3.1}$$

$$\text{fail}(1) = \frac{\beta}{L-\alpha} = \frac{\ell-2}{\ell p - 2}. \tag{3.2}$$

Setting $\ell = p$ yields the corresponding formulas for the complete linear scheme (Equations 2.4 and 2.5). Therefore, the decomposable linear scheme can be seen as a generalization of the linear scheme.

3.3.2 Analysis of the Decomposable Quadratic KPS

In a similar manner to the previous section, the quadratic scheme can be analyzed by using results for partially balanced t-designs.

Definition 8. *A $\overline{TD}(3, k, p, \ell)$, with $\ell \leq p$, is a set system consisting of $B_1, \ldots, B_\ell$ from the resolution of a $TD(3, k, p)$.*

Theorem 10. *A $\overline{TD}(3, k, p, \ell)$ is a 3-$(kp, k, \ell p^2, \ell p, \ell)$-PBtD.*

Proof. Given $B_1, \ldots, B_\ell$ from the resolution of a $\text{TD}(3, k, p)$, let $\mathcal{A}$ be the set of blocks contained within $B_1, \ldots, B_\ell$, and let X be the set of points contained within these blocks. Note that X contains the same kp points of the original $\text{TD}(3, k, p)$, as every point in the design is contained in each B_i. The set $\mathcal{A}$ contains ℓp^2 blocks, as each B_i contains p^2 distinct blocks, and so $\lambda_0 = \ell p^2$. Each point in X is contained in exactly p blocks in each B_i, and is therefore contained in exactly $\lambda_1 = \ell p$ blocks total. Next, consider any pair of points from the original $\text{TD}(3, k, p)$. If this pair of points occurs in the same group $\mathcal{H}$ of the $\text{TD}(3, k, p)$, then they occur in precisely 0 blocks of the design. Otherwise, in each of

$B_1, \ldots, B_\ell$ there is exactly 1 block that contains this pair of points, and $\lambda_2 = \ell$. Finally, any set of 3 points occurs in at most 1 block of the TD$(3, k, p)$, and therefore occurs in either 0 or 1 blocks of the $\overline{\text{TD}}(3, k, p, \ell)$. Thus, all necessary properties from Definition 5 are satisfied. □

As before, we can apply the formulas from section 2.6 to the decomposable quadratic scheme. If a pair of points occur in a block B, and there are λ_2 blocks total containing this pair, then

$$\mu'(2) = \lambda_2 - 1$$

other blocks intersect B at this pair of points. Paterson and Stinson [84] provide a formula to recursively compute $\mu'(1)$:

$$\begin{aligned} \mu'(t-1) &= \lambda_{t-1} - 1, \\ \mu'(i) &= \lambda_i - 1 - \sum_{j=1}^{t-1-i} \binom{k-i}{j} \mu'(i+j). \end{aligned}$$

In the case of a 3-$(kp, k, \ell p^2, \ell p, \ell)$-PB$t$D, we have

$$\begin{aligned} \mu'(1) &= \lambda_1 - 1 - (k-1)\mu'(2) \\ &= \ell p - 1 - (k-1)(\ell - 1). \end{aligned}$$

As with the analysis of the quadratic scheme using PBtDs, we can consider the performance of the parallel class variant with intersection thresholds of both $\eta = 1$ and $\eta = 2$. We begin with the $\eta = 2$ case, where nodes must have a pair of keys in common to communicate. Utilizing the formulas from Section 2.6 gives us:

$$\begin{aligned} \alpha &= \binom{k}{2}\mu'(2) \\ &= \binom{k}{2}(\ell - 1), \\ \\ \beta &= \mu'(2)\left(\frac{\lambda_2}{2} - 1\right)\binom{k}{2} \\ &= (\ell - 1)\left(\frac{\ell}{2} - 1\right)\binom{k}{2}, \\ \\ L &= \frac{b\alpha}{2} \\ &= \frac{\ell p^2(\ell - 1)}{2}\binom{k}{2}, \end{aligned}$$

$$
\begin{aligned}
\text{fail}(1) &= \frac{\beta}{L-\alpha} = \frac{\ell-2}{\ell p^2-2} && (3.3)\\
Pr_1 &= \frac{\alpha}{b-1} = \frac{k(k-1)(\ell-1)}{2(\ell p^2-1)}. && (3.4)
\end{aligned}
$$

Setting $\ell = p$ yields identical results to those derived for the original quadratic scheme.

We can now set $\eta = 1$ and repeat this process, albeit with much more complicated formulas:

$$
\begin{aligned}
\alpha &= k\mu'(1) + \binom{k}{2}\mu'(2)\\
&= k(\ell p-1) - \binom{k}{2}(\ell-1)\\
\beta &= \mu'(1)\left(\frac{\lambda_1}{2}-1\right)k + \mu'(2)\left(\frac{\lambda_2}{2}-1\right)\binom{k}{2}\\
&= (\ell p-1-(k-1)(\ell-1))k\left(\frac{\ell n}{2}-1\right) + (\ell-1)\left(\frac{\ell}{2}-1\right)\binom{k}{2}\\
L &= \frac{b\alpha}{2},\\
&= \frac{\ell p^2\left(k(\ell p-1)-\binom{k}{2}(\ell-1)\right)}{2},
\end{aligned}
$$

$$
\begin{aligned}
\text{fail}(1) &= \frac{\beta}{L-\alpha} = \frac{2(\ell p-1)(\ell p-2)-(k-1)(\ell-1)(2\ell p-\ell-2)}{(\ell p^2-2)(2\ell p-2-(k-1)(\ell-1))} && (3.5)\\
Pr_1 &= \frac{\alpha}{b-1} = \frac{k(2\ell p-2-(k-1)(\ell-1))}{2(\ell p^2-1)}. && (3.6)
\end{aligned}
$$

As expected, these formulas agree with the earlier derived formulas when $\ell = p$. It is worth noting that while the above formulas are quite complicated, they simply contain polynomials in ℓ, p, and k, which are either direct network parameters, or closely related to direct network parameters. In comparison to the schemes by Bose et al. [12], presented earlier in Section 3.1, the above formulas are more intuitive. One can fix a direct network parameter, such as at the number of keys per node k, and easily examine the expected outcome on performance for various network sizes using the above formulas.

3.3.3 Computing Performance Metrics for Arbitrary Set Systems

The performance metrics derived earlier rely heavily on the structure of the underlying design. In the case of random subset schemes, and for other less structured set systems, there may not be enough structure to compute these metrics directly from known network parameters. This section describes a method that can be used to efficiently compute performance metrics for less structured set systems.

Let $(X, \mathcal{A})$ be a set system with b blocks of size k, and suppose the maximum intersection size of any two blocks is $t-1$. For each set of points C of size i, where $\eta \leq i \leq t-1$, define λ_C to be the number of blocks $A \in \mathcal{A}$ containing all points in C. For the linear scheme, λ_C represents the number of times a given single key occurs in the network. In the quadratic scheme with $\eta = 2$, each C is a two-element set, and represents the number of times each pair of points occurs in the network. The λ_C values are sufficient to compute performance metrics for the resulting KPS. Such an approach is beneficial as the values of λ_C are readily available for many types of designs, such as partially balanced t-designs. In the case that values for λ_C are unknown, they can be computed in $\Theta(bk)$ time simply by iterating over each block. By comparison, a naive approach would require each pair of blocks to be compared in order to discover shared sets of keys, resulting in a cost of $\Theta((bk)^2)$.

Computing Connectivity

The λ_C values are straightforward to compute simply by iterating through each block and counting occurrences of keys. We now consider how these values can be used to compute the connectivity Pr_1 for any uniform set system where the maximum intersection size of two blocks is known.

Given a set of points C with $\eta \leq |C| \leq t-1$, define a *C-link* to be a set of two blocks $\{A, B\}$ such that $A \cap B = C$. In other words, A and B form a C-link if they share the set of keys contained in C. The number of C-links is denoted by $\lambda'(C)$, where

$$\lambda'(C) = \left|\{\{A, B\} \,:\, A, B \in \mathcal{A},\, A \cap B = C\}\right|.$$

In the case of the linear KPS, $\lambda'(C)$ represents the number of pairs of nodes that rely on a given single key for communication. In the quadratic KPS, $\lambda'(C)$ is defined for $\eta = 1, 2$, and represents the pairs of nodes that rely on a given key, or a given pair of keys for communication. In all cases, if $|C| = t-1$, then $\lambda'(C) = \binom{\lambda_C}{2}$. The next lemma follows directly from the principle of inclusion-exclusion.

Lemma 4. *If $|C| = i \leq t-1$, then*

$$\lambda'(C) = \sum_{D \subseteq X \setminus C, |D| \leq t-1-i} (-1)^{|D|} \binom{\lambda_{C \cup D}}{2}.$$

In particular, $\lambda'(C) = \binom{\lambda_C}{2}$ if $|C| = t-1$.

Define an *i-link* to be any C-link where $|C| = i$. For $\eta \leq i \leq t-1$, let L_i denote the number of i-links. Note that if $i \geq t$ then there do not exist any i-links, and if $i < \eta$, then the number of shared keys is below the intersection threshold and a secure link does not exist. For any $\eta \leq i \leq t-1$, it follows that

$$L_i = \sum_{|C|=i} \lambda'(C), \tag{3.7}$$

and we can denote the total number of links in the network by

$$L = \sum_{i=\eta}^{t-1} L_i. \tag{3.8}$$

Given the total number of links in the network, the probability that any pair of nodes share a key is given by

$$Pr_1 = \frac{L}{\binom{b}{2}}. \tag{3.9}$$

We now demonstrate how to compute L_i using the λ_C values. Define

$$q_i = \sum_{|C|=i} \binom{\lambda_C}{2}. \tag{3.10}$$

Lemma 5. *For $\eta \leq i \leq t-1$, it holds that*

$$L_i = \sum_{j=i}^{t-1} (-1)^{j-i} \binom{j}{i} q_j. \tag{3.11}$$

Proof. The formula from Lemma 4 must be summed over all C such that $|C| = i$. In doing so, the term $(-1)^{|D|}\binom{\lambda_{C \cup D}}{2}$ is included in the sum $\binom{|C \cup D|}{|C|} = \binom{|D|+i}{i}$ times. □

For $\eta \leq i \leq t-1$, let

$$a_i = \sum_{j=\eta}^{i} (-1)^{i-j} \binom{i}{j}. \tag{3.12}$$

Then we can prove the following.

Theorem 11.

$$L = \sum_{i=\eta}^{t-1} a_i q_i, \tag{3.13}$$

where a_i and q_i are as defined in equations 3.12 and 3.10, respectively.

Proof. When summing Equation 3.11, each q_i is included precisely a_i times. □

Recalling our earlier formula for Pr_1, we have the following.

Corollary 1.

$$Pr_1 = \frac{\sum_{i=\eta}^{t-1} a_i q_i}{\binom{b}{2}}. \tag{3.14}$$

Table 3.1 demonstrates some sample values for L as computed using the formulas in this section. The relevant values for the linear scheme occur when $t = 2$, and the relevant values for the quadratic scheme occur when $t = 3$ and $\eta = 1, 2$. When $t = 3$ and $\eta = 1$, the result $q_1 - q_2$ can be interpreted as the total number of times two blocks intersect at a single point, minus the number of blocks that were counted twice because they intersect at two points.

Computing Resilience

The previous section defined a C-link as a set of two nodes that intersect at all points contained in C, and $\lambda'(C)$ is the number of links that rely on C. Therefore, the number of nodes that break a given C-link $\{A, B\}$ is given by $\lambda_C - 2$. Because there are $b - 2$ other nodes in the network, the probability that a random node breaks a given link is given by

$$\frac{\lambda_C - 2}{b - 2}.$$

Averaging this over all L links in the network yields

$$\text{fail}(1) = \frac{1}{L} \sum_{\{C\,:\,\eta \leq |C| \leq t-1\}} \frac{(\lambda_C - 2)\lambda'(C)}{b - 2}. \tag{3.15}$$

Table 3.1: Sample values for L for several small intersection thresholds.

t	η	L
2	1	q_1
3	2	q_2
3	1	$q_1 - q_2$
4	3	q_3
4	2	$q_2 - 2q_3$
4	1	$q_1 - q_2 + q_3$
5	4	q_1
5	3	$q_3 - 3q_4$
5	2	$q_2 - 2q_3 + 3q_4$
5	1	$q_1 - q_2 + q_3 - q_4$

As with the analysis of connectivity in the previous section, our goal is to find an equivalent formula which relies only on the λ_C values, and not the $\lambda'(C)$ values. Recalling Lemma 4, we note the following when $|C| \leq t-1$:

$$\sum_{\{C\,:\,\eta \leq |C| \leq t-1\}} \lambda_C \lambda'(C) = \sum_{\{C\,:\,\eta \leq |C| \leq t-1\}} \left(\lambda_C \sum_{D \subseteq X \setminus X, |D| \leq t-1-i} (-1)^{|D|} \binom{\lambda_{C \cup D}}{2} \right).$$

Let $E = C \cup D$, then the above is equal to

$$\sum_{\{E\,:\,\eta \leq |E| \leq t-1\}} \left(\binom{\lambda_E}{2} \sum_{\{C\,:\,\eta \leq |C|, C \subseteq E\}} (-1)^{|E|-|C|} \lambda_C \right).$$

Separating out the constant terms yields the following lemma.

Lemma 6.

$$\sum_{\{C\,:\,\eta \leq |C| \leq t-1\}} \lambda_C \lambda'(C) = \sum_{\{E\,:\,\eta \leq |E| \leq t-1\}} \mu_E \binom{\lambda_E}{2}, \tag{3.16}$$

where

$$\mu_E = \sum_{\{C\,:\,\eta \leq |C|, C \subseteq E\}} (-1)^{|E|-|C|} \lambda_C. \tag{3.17}$$

Some special cases of this lemma are useful

$$\mu_E = \begin{cases} \lambda_E & \text{if } |E| = \eta \\ \lambda_E - \sum_{x \in E} \lambda_{E \setminus \{x\}} & \text{if } |E| = \eta + 1. \end{cases} \tag{3.18}$$

Applying Lemma 6 to Equation 3.15 allows us to state the following theorem.

Theorem 12.

$$\mathit{fail}(1) = \frac{1}{L(b-2)} \left(\sum_{\{E\,:\,\eta \le |E| \le t-1\}} \mu_E \binom{\lambda_E}{2} \right) - \frac{2}{b-2}. \tag{3.19}$$

Proof. The result follows directly from applying Lemma 6 to Equation 3.15, with the observation that

$$\sum_{\{C\,:\,\eta \le |C| \le t-1\}} \lambda'(C) = L,$$

which follows from Equations 3.7 and 3.8. □

Efficiently Computing Connectivity and Resilience

Given a set system $(X, \mathcal{A})$ with $b = |\mathcal{A}|$, with maximum block intersection $t-1$, the following method can be used to efficiently compute Pr_1 and fail(1) for a KPS built from this set system without any additional knowledge of the structure of the design.

1. For $\eta \le |C| \le t-1$, compute λ_C for each set C as follows:
    (a) Set $\lambda_C \leftarrow 0$ for all C.
    (b) For each block $A \in \mathcal{A}$ and for every $C \subseteq A$ such that $\eta \le |C| \le t-1$, set $\lambda_C \leftarrow \lambda_C + 1$. Node that when η and t are fixed, known values, this requires $\Theta(b)$ time.
2. Compute μ_C for $\eta \le |C| \le t-1$ using Equation 3.17.
3. Compute q_i for $\eta \le i \le t-1$ using Equation 3.10.
4. Compute L using Equation 3.13.
5. Compute Pr_1 using Equation 3.14, and compute fail(1) using Equation 3.19.

The approach outlined here is used in the next section to compute performance metrics for the random linear KPS (Scheme 12) and random quadratic KPS (Scheme 13) for a variety of network sizes.

Examples

The algorithm in the previous section can be demonstrated through some simple examples.

Example 1. Recall the configuration considered in Figure 2.2. The complete design is defined by

$$\begin{aligned} X &= \{1,2,3,4,5,6,7\} \\ \mathcal{A} &= \{123,145,167,247,256,346,357\} \end{aligned}$$

with $Pr_1 = 1$ and $\text{fail}(1) = 0.2$. Each pair of points occurs in at most a single block, so $t = 2$. If we change the last two blocks of the set system by introducing a new key:

$$\mathcal{A}' = \{123,145,167,247,256,348,358\},$$

then the previously established formulas for a configuration are no longer valid. The resulting λ values are

$$\lambda_1 = 3 \quad \lambda_2 = 3 \quad \lambda_3 = 3 \quad \lambda_4 = 3 \quad \lambda_5 = 3 \quad \lambda_6 = 2 \quad \lambda_7 = 2 \quad \lambda_8 = 2.$$

From this it is easy to compute $q_1 = 18$. Because $t = 2$ and $\eta = 1$, we know from Table 3.1 that $L = q_1 = 18$, and, by Equation 3.14,

$$Pr_1 = \frac{L}{\binom{b}{2}} = \frac{18}{21} \approx 0.86.$$

Because the maximum intersection threshold in this scheme is $t - 1 = 1$, it holds that $\mu_C = \lambda_C$, as $|C| = 1$ in all cases. Using Equation 3.19 we can compute

$$\text{fail}(1) = \frac{1}{18 \times 5}\left(5 \times 3\binom{3}{2} + 3 \times 2\binom{2}{2}\right) - \frac{2}{5} \approx 0.17.$$

Unsurprisingly, a small modification to the underlying set system has only a small impact on the performance metrics. We now consider a more complicated example.

Example 2. Consider the following set system

$$\begin{aligned} X &= \{1,2,3,4,5,6\} \\ \mathcal{A} &= \{123,124,125,456,136\}. \end{aligned}$$

By simple inspection it can seen that some pairs of keys occur in multiple blocks, but any triple is unique, therefore $t = 3$ and we can compute metrics for intersection thresholds of $\eta = 1, 2$. The λ_C values are as follows

$$\lambda_1 = 4 \quad \lambda_2 = 3 \quad \lambda_3 = 2 \quad \lambda_4 = 2 \quad \lambda_5 = 2 \quad \lambda_6 = 2$$

when $|C| = 1$, and

$$\begin{array}{ccccc}
\lambda_{1,2} = 3 & \lambda_{1,3} = 2 & \lambda_{1,4} = 1 & \lambda_{1,5} = 1 & \lambda_{1,6} = 1 \\
 & \lambda_{2,3} = 1 & \lambda_{2,4} = 1 & \lambda_{2,5} = 1 & \lambda_{2,6} = 0 \\
 & & \lambda_{3,4} = 0 & \lambda_{3,5} = 0 & \lambda_{3,6} = 1 \\
 & & & \lambda_{4,5} = 1 & \lambda_{4,6} = 1 \\
 & & & & \lambda_{5,6} = 1
\end{array}$$

when $|C| = 2$. From these, we compute $q_1 = 13$ and $q_2 = 4$. When $\eta = 1$, we have $L = q_1 - q_2 = 9$ and

$$Pr_1 = \frac{L}{\binom{b}{2}} = \frac{9}{\binom{5}{2}} = \frac{9}{10}.$$

When $\eta = 2$, we have $L = q_2 = 4$ and

$$Pr_1 = \frac{4}{10}.$$

In order to compute fail(1), we must first compute each μ_C. When $|C| = \eta = 2$ it holds that $\mu_C = \lambda_C$, and Equation 3.19 gives us

$$\text{fail}(1) = \frac{1}{4 \times 3}\left(3\binom{3}{2} + 2\binom{2}{2}\right) - \frac{2}{3} = \frac{1}{4}.$$

Note that only $\lambda_{1,3} = 3$ and $\lambda_{1,4} = 2$ produce non-zero terms for $\binom{\lambda_C}{2}$. When $\eta = 1$, we use Equation 3.18 to compute

$$\begin{array}{ccccc}
\mu_{1,2} = -4 & \mu_{1,3} = -4 & \mu_{1,4} = -5 & \mu_{1,5} = -5 & \mu_{1,6} = -5 \\
 & \mu_{2,3} = -4 & \mu_{2,4} = -4 & \mu_{2,5} = -4 & \mu_{2,6} = -5 \\
 & & \mu_{3,4} = -4 & \mu_{3,5} = -4 & \mu_{3,6} = -3 \\
 & & & \mu_{4,5} = -3 & \mu_{4,6} = -3 \\
 & & & & \mu_{5,6} = -3.
\end{array}$$

Again, using Equation 3.19, we have

$$\text{fail}(1) = \frac{1}{9 \times 3}\left(4\binom{4}{2} + 3\binom{3}{2} + 4 \times 2\binom{2}{2} - 4\binom{3}{2} - 4\binom{2}{2}\right) - \frac{2}{3} = \frac{7}{27}.$$

More results using this approach, but applied to much larger families of set systems, are given in the next section.

3.4 Comparison of Decomposable and Randomized Approaches

This chapter has thus far presented several flexible constructions for combinatorial design-based KPSs based on a family of transversal designs. In this section, we compare the performance metrics Pr_1 and fail(1) for a variety of network parameters. Nine different schemes are considered:

A: The Linear KPS (Scheme 7), based on a complete $\mathrm{TD}(2,k,p)$.

B: The Quadratic KPS (Scheme 8), based on a complete $TD(3,k,p)$ with intersection threshold $\eta = 2$.

C: The Quadratic KPS (Scheme 8), based on a complete $\mathrm{TD}(3,k,p)$ with intersection threshold $\eta = 1$.

D: The Decomposable Linear KPS (Scheme 15), based on a $\overline{\mathrm{TD}}(3,k,p,\ell)$ with $\ell \leq p$.

E: The Decomposable Quadratic KPS (Scheme 16), based on a $\overline{\mathrm{TD}}(3,k,p,\ell)$ with $\ell \leq p$ and intersection threshold $\eta = 2$.

F: The Decomposable Quadratic KPS (Scheme 16), based on a $\overline{\mathrm{TD}}(3,k,p,\ell)$ with $\ell \leq p$ and intersection threshold $\eta = 1$.

G: The Random Linear KPS (Scheme 12), based on a random subset of a $\mathrm{TD}(2,k,p)$.

H: The Random Quadratic KPS (Scheme 13), based on a random subset of a $\mathrm{TD}(3,k,p)$ with intersection threshold $\eta = 2$.

I: The Random Quadratic KPS (Scheme 13), based on a random subset of a $\mathrm{TD}(3,k,p)$ with intersection threshold $\eta = 1$.

Performance metrics for these schemes are summarized in Table 3.2. Note that schemes **A**, **D**, and **G** are identical when $\ell = p$ in the decomposable scheme, or the entire network is included in the random scheme. Similarly, schemes **B**, **E**, and **H** are identical when the entire design is used, as are schemes **C**, **F**, and **I**.

For scheme **A**, the total number of nodes in the underlying $\mathrm{TD}(2,k,p)$ is p^2, and for schemes **B** and **C**, the total number of nodes in the underlying $\mathrm{TD}(3,k,p)$ is p^3. In the case of schemes **D**, **E**, and **F**, the total number of nodes in the underlying $\overline{\mathrm{TD}}(t,k,p,\ell))$

Table 3.2: Comparison of performance metrics for each transversal design-based scheme. The parameters L, λ_E, and μ_E are defined in Section 3.3.3, and correspond to the total number of links in the network, the number of occurrences of each set of keys in the network, and the number of links relying on a given set of keys, respectively.

Scheme	Pr_1	fail(1)
A:	$\frac{k}{p+1}$	$\frac{p-2}{p^2-2}$
B:	$\frac{k(k-1)}{2(p^2+p+1)}$	$\frac{p-2}{p^2-2}$
C:	$\frac{k(2p-k+3)}{2(p^2+p+1)}$	$\frac{2p^3+(4-2k)p^2+(k-5)p+2k-6}{(2p-k+3)(p^3-2)}$
D:	$\frac{k(\ell-1)}{\ell p-1}$	$\frac{\ell-2}{\ell p-2}$
E:	$\frac{k(k-1)(\ell-1)}{2(\ell p^2-1)}$	$\frac{\ell-2}{\ell p^2-2}$
F:	$\frac{k(2\ell p-2-(k-1)(\ell-1))}{2(\ell p^2-1)}$	$\frac{2(\ell p-1)(\ell p-2)-(k-1)(\ell-1)(2\ell p-\ell-2)}{(\ell p^2-2)(2\ell p-2-(k-1)(\ell-1))}$
G,H,I:	$\frac{L}{\binom{b}{2}}$	$\frac{1}{L(b-2)}\left(\sum_{\{E\,:\,\eta\leq\|E\|\leq t-1\}}\mu_E\binom{\lambda_E}{2}\right)-\frac{2}{b-2}$

is $\ell \cdot p^{t-1}$. For schemes **G**, **H**, and **I**, the underlying design is a $\mathrm{TD}(t,k,p)$, but any number of blocks $1 \leq b \leq p^t$ may be selected. In order to accurately compare the relative performance of these schemes, parameters must be chosen such that the resulting network sizes are approximately equal. The following sets of parameters achieve this:

1. Let $N \approx 5000$. Then schemes based on a $\mathrm{TD}(2,15,71)$ can be compared to schemes based on a $\mathrm{TD}(3,15,17)$. In this case, $71^2 = 5041$ and $17^3 = 4913$, and the size of the resulting networks differ by less than 3%. Each block in these designs contains 15 points, which corresponds directly to the number of keys each node stores.

2. Let $N \approx 24000$. Then schemes based on a $\mathrm{TD}(2,25,157)$ can be compared to schemes based on a $\mathrm{TD}(3,25,29)$. In this case, $157^2 = 24649$ and $29^3 = 24387$, and the size of the resulting networks differ by approximately 1%. Each block in these designs contains 25 points, and each node stores 25 keys.

Because schemes **A**, **B**, and **C** form the basis for the flexible schemes proposed in this chapter, but do not allow for flexible selection of parameters themselves, we use them as a baseline for comparison. The decomposable schemes (**D**, **E**, and **F**) allow for network sizes of $m = \ell \cdot p^{t-1}$ for $0 \leq \ell \leq p$, however, only results for $\ell \geq 2$ are of interest. This restriction is due to the fact that a single parallel class from a $\mathrm{TD}(2,k,p)$ does not contain any intersecting blocks, and a single group from the decomposition of a $\mathrm{TD}(3,k,p)$ does

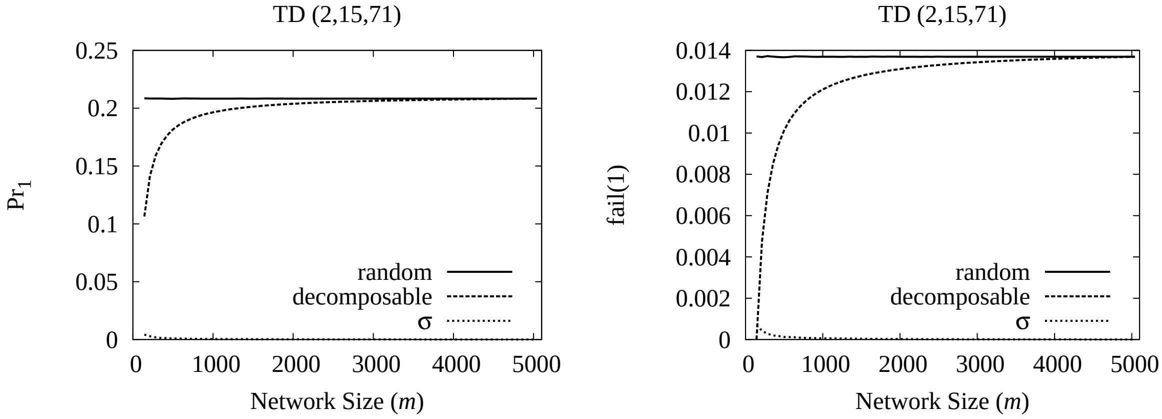


Figure 3.2: Comparison of linear schemes based on a $\mathrm{TD}(2, 15, 17)$ with maximal network size $N \approx 5000$.

not contain any blocks that intersect in two points, so the resulting connectivity is $Pr_1 = 0$. For the random schemes (**G**, **H**, and **I**) we compute average results over multiple random subsets of size $\ell \cdot p^{t-1}$ to match the network sizes of the decomposable schemes. For each datapoint we distribute keys according to a $\mathrm{TD}(t, k, p)$, select a random subset of nodes of size $m = \ell \cdot p^{t-1}$ for $1 \leq \ell \leq p$, and use the process described in Section 3.3.3 to compute Pr_1 and fail(1). This process is repeated over 100 trials, and the average and standard deviation for each datapoint are computed. As seen in the graphs and in Appendix B, 100 trials was sufficient to produce extremely low deviation across trials.

Each set of results in this section is presented in a graph which includes a line demonstrating the relevant metric for the decomposable schemes, computed using the formulas in Table 3.2, and average results for the randomized scheme as described above. The standard deviation is also included for the random case, but is too small to be distinguished for most datapoints. Tables with exact data values are included in Appendix B for reference. Both the decomposable and random schemes converge to the performance of the complete schemes when the entire network is included.

We first consider schemes **A**, **D**, and **G** based on a $\mathrm{TD}(2, 15, 71)$, with results presented in Figure 3.2. Observe that the performance of the random scheme is essentially flat, with only small deviations when the subnetwork size is small. The decomposable schemes provide poor connectivity and high resilience, but quickly approach the expected performance of the complete network. Once approximately a third of the nodes are included, the performance of the decomposable scheme is nearly the same as the original

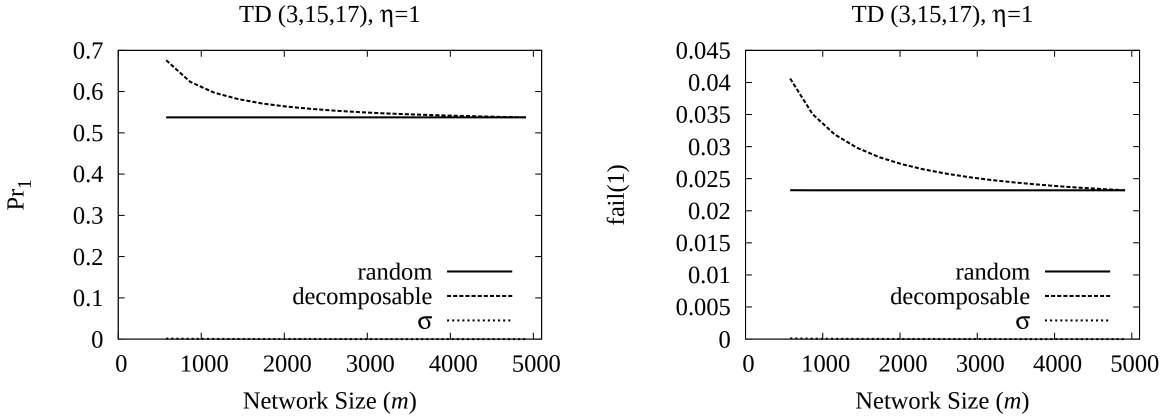


Figure 3.3: Comparison of quadratic schemes with intersection threshold $\eta = 1$ based on a TD$(3, 15, 17)$ with maximal network size $N \approx 5000$.

scheme.

Schemes **A**, **D**, and **G** based on a TD$(2, 15, 71)$ can be compared directly to Schemes **C**, **F**, and **I** based on a TD$(3, 15, 17)$ with intersection threshold $\eta = 1$. This comparison is the most direct, as only a single key is required for secure communication, even though a pair of nodes may share two keys. Results are presented in Figure 3.3. As with the $t = 2$ case, we see that the random scheme tends to approximate performance of the network from the beginning, and the decomposable scheme quickly approaches the performance of the complete, becoming a good approximation once about a third of the nodes are included. The resulting connectivity is higher, accompanied by a loss of resilience (larger fail(1)), due to the fact that more nodes intersect at a single point in the quadratic scheme. The main observable difference is the shape of the curves. The decomposable scheme begins with a much higher connectivity (and lower resilience), and converges to the expected performance from above. This behavior follows from the fact that each group in the decomposition of a TD$(3, 15, 17)$ is a TD$(2, 15, 17)$, whereas each group in the decomposition of a TD$(2, 15, 71)$ is a parallel class with no intersecting blocks. The connectivity and resilience of a TD$(2, 15, 17)$-based KPS are

$$\begin{aligned} Pr_1 &= \frac{k}{p+1} = \frac{15}{18} \approx .833 \\ \text{fail}(1) &= \frac{p-2}{p^2-2} = \frac{13}{15^2-2} \approx .0523. \end{aligned}$$

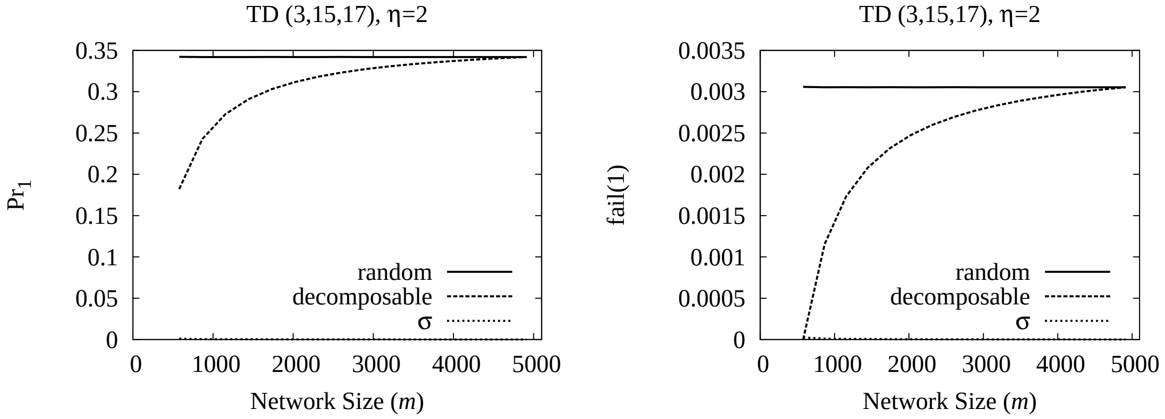


Figure 3.4: Comparison of quadratic schemes with intersection threshold $\eta = 2$ based on a $\mathrm{TD}(3, 15, 17)$ with maximal network size $N \approx 5000$

As ℓ grows, more groups are added to the set of included blocks. Within each group the expected connectivity and resilience are as above, but the connectivity of nodes between groups is lower. As the probability that two random nodes fall within the same group drops, the expected performance of the network quickly converges to the expected performance of the complete scheme.

Schemes **B**, **E**, and **H** based on a $\mathrm{TD}(3, 15, 17)$ with intersection threshold $\eta = 2$ can be compared to the $\eta = 1$ case to observe the effect of requiring a pair of nodes to posses two shared keys in order to communicate. Results are presented in Figure 3.4. The shape of the graph resembles the performance of the $\mathrm{TD}(2, 15, 71)$-based scheme, which is expected. Each B_i in the resolution is a $\mathrm{TD}(2, 15, 17)$, and therefore no two blocks from a given B_i will ever intersect at two points. As more blocks from the resolution are added, more pairs of nodes with two shared keys exist and the performance approaches that of the complete scheme. In general, the $\eta = 2$ is slower to converge to the expected performance, with closer to half the network being required to achieve a good approximation, and provides a lower connectivity than the $\eta = 1$ case. To balance this, the quadratic scheme achieves a significant boost to resilience (approximately an order of magnitude), which follows from the fact that two specific keys must be compromised for a given link to be broken. As in the other cases, the random scheme tends to approximate performance quite closely even for small network sizes.

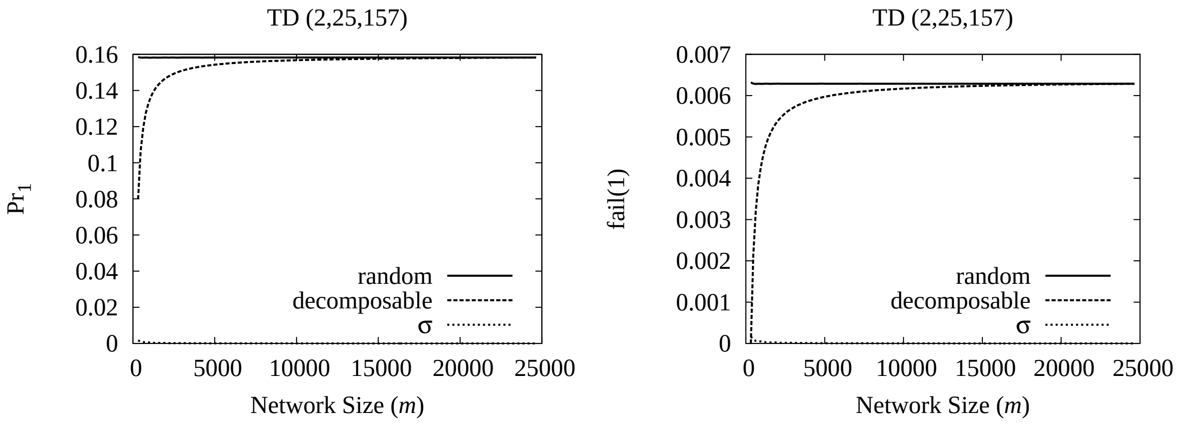


Figure 3.5: Comparison of linear schemes based on a TD$(2, 25, 157)$ with maximal network size $N \approx 24000$.

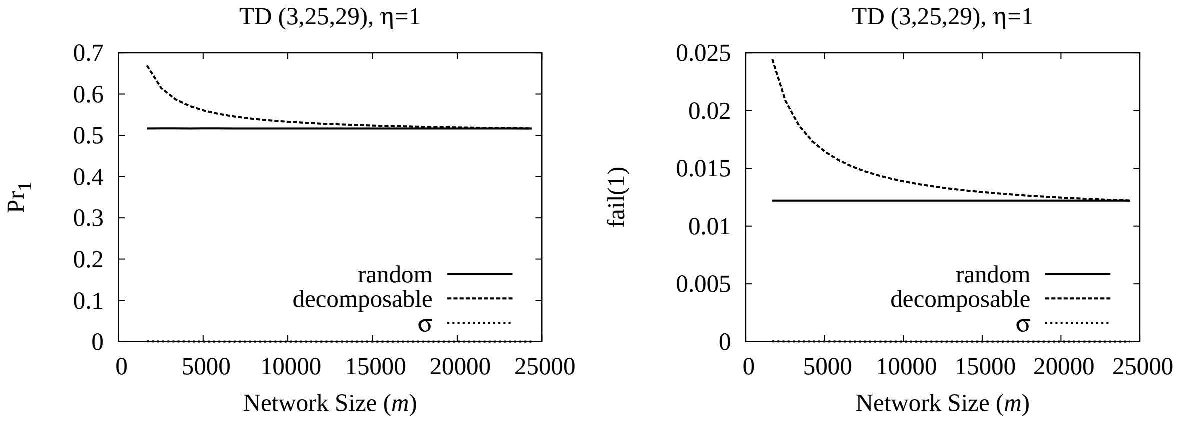


Figure 3.6: Comparison of quadratic schemes with intersection threshold $\eta = 1$ based on a TD$(3, 25, 29)$ with maximal network size $N \approx 24000$.

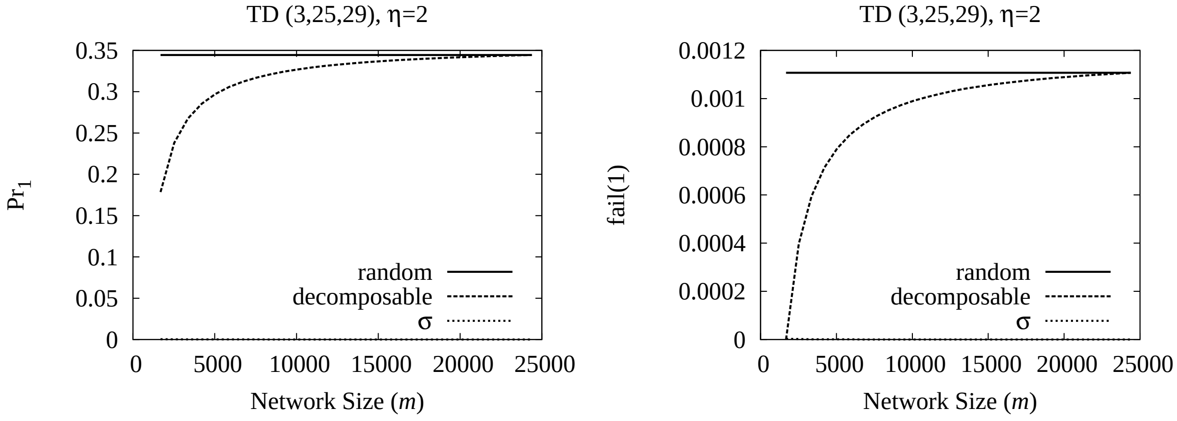


Figure 3.7: Comparison of quadratic schemes with intersection threshold $\eta = 2$ based on a TD$(3, 25, 29)$ with maximal network size $N \approx 24000$

Figures 3.5, 3.6, and 3.7 contain corresponding results for a larger network of $N \approx 24000$ nodes. The observed patterns of performance are the same as the $N \approx 5000$ case. Further trials for several other network sizes were computed, but have not been included as the observed performance patterns were consistent with the included results.

3.5 Summary and Remarks

This chapter established the problem of key distribution and demonstrated that combinatorial designs provide a convenient mathematical tool for solving the key pre-distribution problem. Through a series of stricter definitions, transversal designs were identified as an ideal candidate for building KPSs, as they are amenable to concrete analysis of performance metrics, can be constructed simply, and incur a low computational cost for nodes to determine shared keys. The main drawback of these designs lies in the restrictions on network size and its relationship to the maximum number of keys per node.

Attempts to provide more flexible approaches have been proposed, but the simplicity of TD-based approaches was lost. This chapter presented two approaches, along with thorough analysis, that allow transversal designs to be used for a much wider variety of parameters. These approaches allow for the construction of a TD-based KPS using only two parameters, p and k, to distribute keys to a network of size $m = \ell \cdot p \leq p^2$ or $m = \ell \cdot p^2 \leq p^3$

as long as $k \leq p$, and provide concrete formulas for performance metrics. An alternate approach based on random subsets was also presented that allows for arbitrary network sizes of $m \leq p^2$ or $m \leq p^3$, with an efficient algorithm for computing performance metrics. Although not considered in this thesis, the results generalize to higher-strength designs.

Performance results demonstrate that in our deterministic schemes, for the parameter sets considered, as long as at least half of the blocks in the underlying design are utilized for key pre-distribution, then performance closely matches that of the entire design. This result allows for the use of the simple (and more intuitive) formulas for the entire design to be used as an accurate estimate for performance. If less than half of the blocks are included, the changes in connectivity and resilience are predictable according to our analysis. In general, if only a small fraction of the network is to be deployed, then better performance can be achieved by choosing different parameters (i.e., a smaller value of p) and deploying a larger fraction of the network.

Performance results for the randomized subset schemes demonstrate that even when a very small subset of blocks are included from the underlying design, the resulting performance matches that of the entire design extremely closely. This result suggests that networks of arbitrary size can be easily accommodated using the randomized subset schemes, with virtually no loss of performance.

Chapter 4

Secure Network Discovery Using Combinatorial Key Pre-distribution

4.1 Introduction

The previous two chapters presented an introduction to combinatorial key pre-distribution, and then derived a family of flexible combinatorial key pre-distribution schemes for wireless sensor networks. These schemes are applicable at the pre-deployment phase of a sensor network's operational lifetime, and allow nodes to engage in secure communication even when the post-deployment topology is unknown. This chapter is concerned with the deployment and setup phases of a sensor network, and utilizes specific properties of linear KPSs to facilitate network discovery in the presence of active malicious nodes.

Recall that, according to Martin and Paterson's framework [74], sensor networks can be broadly categorized according to their relative capabilities, how they are deployed, and their ideal communication structure. In a *homogeneous* sensor network all nodes are identical, whereas a *hierarchical* network contains nodes of varying capabilities. In practice, this is often in the form of a two- or three-level tree, with many low power nodes forwarding information to a single higher power node (often called a base station). A large network may have several base stations, all forwarding their data to some final central location.

In terms of deployment, sensor nodes can either be *fixed* or *mobile*. In the latter case, a distinction can be made between nodes that are free to travel throughout the entire network (*fully mobile*), and those nodes may only move within in fixed region (*locally mobile*). When considering fixed networks, distinctions are made between having *full control*, *partial control*, or *no control* over the location of nodes after deployment.

This chapter is concerned with a simple homogeneous, fixed, no-control network with pre-distributed cryptographic keys. The goal is to investigate what assumptions are necessary to allow a node to accurately determine the local network topology in the presence of an active adversary. In particular, we are concerned with establishing multiple node-disjoint paths between a source node and its destination. Node-disjoint paths allows for the use of the perfectly secure message transmission protocol of Dolev et al. [32], as well as other protocols built upon the same idea, such as the multi-path key establishment protocols of Wu and Stinson [108], or the multi-path key reinforcement protocol by Chan et al. [24]. These protocols assume that multiple node-disjoint paths exist between a source and a destination, but do not specify how such a set of paths can be discovered. When the topology is unknown prior to deployment, then nodes must be able to determine the local network topology, even in the presence of one or more malicious nodes attempting to inject false routing information.

We present a protocol that proceeds in multiple steps, each of which allows a node to expand its view of the network by one hop. This protocol is based on voting and the assumption that an honest majority of nodes are present at each point in the network in which a decision must be made about the identity of a node. We accomplish this goal through the use of a linear key pre-distribution scheme that allows a node to prove its identity by proving it possesses two specific keys. In order to evaluate the practicality of this approach, simulation results are provided that demonstrate that, with proper parameter choices, we can still determine network topology even when a large number of malicious nodes are present during network discovery.

The work in this chapter has been published previously [48].

4.1.1 Related Work

In addition to providing a framework for categorizing sensor networks, Martin and Paterson [74] provide a comprehensive survey of key establishment protocols for sensor networks, while Karlof and Wagner [54] have surveyed a number of attacks against routing protocols in sensor networks.

Many secure routing protocols for ad-hoc wireless networks have been presented, such as ARAN [97], S-AODV [114], and DV-SRP [83], but they have not focused on solving the problem of establishing several node-disjoint paths to a destination. The protocol presented here addresses this problem.

Poturalski et al. [88] have given a formal treatment of neighborhood discovery in wireless networks and give an impossibility result for protocols that rely solely on message transmission time or distance between nodes. However, it was shown that secure protocols utilizing both distance and location are possible. Their model specifically excluded protocols that rely on cooperation with other nodes, and hence their result does not directly apply to the protocol presented here.

Recall that Eschenauer and Gligor [37] presented a simple key pre-distribution scheme based on randomly assigning a small set of keys (drawn from a larger pool) to each node, and Chan et al. [24] have extended this approach. Others have considered combinatorial key pre-distribution, such as Çamtepe and Yener [20] or Lee and Stinson [61], with the latter being utilized in this thesis to construct a new family of flexible key pre-distribution schemes. In these deterministic schemes, a node's identifier determines which keys it holds via a public function, although the keys themselves are still chosen randomly. More specifically, the combinatorial design specifies the labels of the keys each node holds, but not the keys themselves. These approaches were described in much greater detail in Chapter 2. In particular, the scheme of Lee and Stinson is utilized here to allow for the authentication of a node's identity (and therefore its entire keylist), by proving it possesses any two keys consistent with its claimed identity.

Distance bounding protocols [14] utilize different characteristics of the communication medium to put an upper bound on the distance between two nodes. Rasmussen and Čapkun [92] have demonstrated that extremely accurate distance bounding is possible over radio frequency. Their solution is practical for sensor networks.

A recent direction in sensor network research is to consider the use of directional antennas instead of omnidirectional antennas. Boudour et al. [13] have investigated how modern protocols can be changed to accommodate the use of directional antennas, and Ash and Potter [6] have demonstrated a method for sensor network localization that in some cases can estimate the location of a node to within one or two meters using directional antennas.

4.2 Problem Setting

Let $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$ be a sensor network. Our focus is limited to *homogeneous* sensor networks, in which the storage, computation, and communication capabilities of sensor nodes are identical. The method of communication used by all nodes is omnidirectional wireless broadcasts. More specifically, all nodes have a fixed communication radius d such that any message sent by a given node will be overheard by any node within distance d and no others. The set of nodes within distance d of node n_i is referred to as n_i's *neighborhood*, with the nodes in n_i's neighborhood being referred to as n_i's *neighbors*.

In order to facilitate secure communication between sensor nodes, a *key pre-distribution scheme (KPS)* is utilized to issue each node with a subset of cryptographic keys selected from a master key list. The set of keys issued to each node is determined by a unique identity associated with each node n_i. Although nodes are free to perform both unencrypted or encrypted communication, transmission of sensor readings should only be done if a secure path exists all the way from the source to the destination. By secure, we mean that each pair of nodes communicating along the path must be able to communicate using a shared key.

Upon deployment, nodes are distributed geographically randomly (i.e. a *fixed, no control network*). Thus, upon activation, no node has any knowledge of its neighbors. The goal of each node is to discover the local network topology to the extent that it can establish multiple disjoint paths between itself and a base station or other desired destination. We require multiple disjoint paths rather than the shortest path so that multi-path protocols, such as perfectly secure message transmission, can be utilized. These protocols ensure that a node can always report its readings, even in the presence of an active adversary, when the assumptions of the protocol are satisfied.

The process of deploying the sensor network (say, for example, by dropping them out of an airplane) may cause nodes to become damaged or to malfunction. Thus, each node will only route messages through those nodes it believes are functioning properly and are capable of secure communication with other nodes. Furthermore, if nodes are deployed in a hostile environment, then an adversary may attempt to reprogram a subset of the nodes, jam communication between nodes, or insert fake nodes into the network, thus providing additional incentives for each node to verify the correct functioning of those nodes it communicates with.

We stress that the model considered here is extremely restrictive. Nodes are deployed with no prior knowledge of the topology, and are pre-loaded with only a small number of symmetric keys to facilitate secure communication. We assume an adversary is present and that a random subset of nodes in the network are compromised. The lack of information about the network and the presence of an adversary from the very beginning necessitates several strong assumptions in order to limit the ability of compromised nodes and to accurately discover the network topology. These assumptions are considered in the next section, alongside a discussion of how they can be realized in practice.

4.3 Tools and Assumptions

4.3.1 Authentication

Consider the simple problem of two nodes wishing to verify whether or not they both possess a shared key, and hence, are capable of secure communication with each other. One may attempt to use a simple challenge/response protocol to solve this problem. n_i sends an encrypted message to n_j. If n_j is in possession of the key used to encrypt the message, then it can respond with the correct plaintext message. Unfortunately, such a simple solution is not sufficient. Consider a node n_i attempting to verify a shared key K with a malicious node m_j who does not actually possess the key. A malicious node m_k who does possess the key could reply in place of m_j, thereby spoofing m_j's identity. For this reason, we introduce the following assumption:

Assumption 1. *(**Identity**) We assume that each node has the ability to distinguish between two messages from the same source, and two messages from different sources.*

This assumption allows n_i to verify that the response to its challenge came from m_j and no one else, and in practice, can be realized through wireless fingerprinting [31, 91], which provides a mechanism for obtaining a fingerprint unique to each node from a received RF signal. Moreover, a mutual authentication protocol utilizing fingerprinting is given by Rasmussen and Čapkun [91], which is utilized here for network discovery. This protocol is presented in Figure 4.1. No formal proof of security is provided, but the protocol is similar in spirit to the secure SKID3 protocol [77, Section 10.17] with additional checks added to make sure the fingerprints are as expected. We will henceforth refer to this as the *Fingerprinted Mutual Authentication Protocol (FMAP)*.

The FMAP protocol, as presented, assumes that each node is pre-loaded with the fingerprint of every other node. In order to avoid the overhead this would cause (and the inability to add nodes to the network at a later time), we have each node commit to its identity by broadcasting a simple `HELLO` message containing its claimed identity (and hence its key list). Each node that overhears such a message can verify that the fingerprint has not been seen before and store it locally, which allows us to make the following assumption:

Assumption 2. *(**Uniqueness**) We assume that each node can commit to at most one unique identity.*

The node identity and uniqueness assumptions allow us to protect against many spoofing attacks and the Sybil attack [35], where a single node pretends to be many nodes within

$$\begin{array}{ccc}
\text{(A)lice} & & \text{(B)ob} \\
\text{Pick } N_A \in_U \{0,1\}^k & & \\
 & \xrightarrow{A,N_A} & \textbf{if } fp_{sig} \approx fp_A \textbf{ continue} \\
 & & \text{Pick } N_B \in_U \{0,1\}^k \\
\textbf{if } fp_{sig} \approx fp_B \textbf{ continue} & \overset{\substack{B,N_A,N_B \\ MAC_K(B,N_A,N_B)}}{\longleftarrow} & \\
A \text{ accepts } B & & \\
 & \overset{\substack{A,N_A,N_B \\ MAC_K(A,N_A,N_B)}}{\longrightarrow} & \textbf{if } fp_{sig} \approx fp_A \textbf{ continue} \\
 & & B \text{ accepts } A
\end{array}$$

Figure 4.1: The Fingerprinted Mutual Authentication Protocol (FMAP) [91]. Here fp_{sig} denotes the fingerprint of the received message, fp_A and fp_B the fingerprints of Alice and Bob respectively, and K is a key shared by Alice and Bob.

the network. Otherwise, a successful Sybil attack would allow an adversary to cast several votes in a consensus-based protocol in order to alter the outcome.

Rasmussen and Čapkun's experimental results on fingerprinting were initially carried out with a single powerful receiver in a network that computed all fingerprints at a single location. The intention was to demonstrate that RF fingerprinting was possible for sensor nodes, and to later port the technique to individual nodes. Others have continued in this direction, such as Knox and Kunz [57], who also investigate authentication via RF fingerprinting, but they limit their attention to what can be achieved using embedded processors on the nodes themselves. Their approach assumes no pre-shared fingerprint information, and tolerates inaccuracies by pooling results from all nearby nodes and allows a group of connected nodes to establish a "conference key" shared by everyone in the group, ultimately leading to the ability to authenticate direct neighbors. Although we use the FMAP protocol in this paper, the correctness of our approach relies only on the assumptions outlined in this section, and other approaches, such as Knox and Kunz, could be substituted.

It is important to note that the FMAP protocol only demonstrates that node m_j has the ability to compute a MAC under a given key K. This setting does not exclude the case where a node m_j may use another node m_k as an encryption or decryption oracle. More specifically, m_j may forward the challenge message to a colluding node m_k, who then responds to m_j with the appropriate response, who then replies to n_i. Because n_i receives the response from the "correct" node, the fingerprints will match and n_i will be falsely convinced that m_j possesses a shared key. In order to combat this, we reiterate an assumption from Section 4.2:

Assumption 3. ***(Uniformity)*** *We assume that all nodes have uniform communication range and all messages sent by a node are received by all of its neighbors.*

By this assumption, if m_j attempts to forward the challenge to a colluding node, then n_i will overhear it. Furthermore, any malicious neighbor of n_i cannot send a message to m_j without n_i overhearing it. In practice, this requires silence from other nodes in n_i's neighborhood until the mutual authentication protocol completes. Should this be violated, the protocol will have to be re-run.

The Uniformity assumption is extremely limiting, and unlikely to hold even in ideal circumstances in a real world deployment. The purpose of making such a strong assumption is to perfectly limit the ability of adversarial nodes to collaborate without detection. In practice, it is sufficient to detect and respond to this behavior in a manner that detects most instances of misbehavior. To realize this in a more practical setting, we could rely on received signal strength (RSS) measurements. Rather than assume that each node communicates with a consistent, pre-defined radius, we can detect and respond to many cases of nodes significantly altering their communication strength, much in the same way we record fingerprints for each node. Along with recording a fingerprint, nodes can record the expected RSS for each neighbor. If at any time a node is observed to be broadcasting a message at a significantly different rate, it can be assumed that a malfunction or malicious behavior is occurring. In order for two malicious nodes to lower their communication range and "whisper" to each other without detection, they must be closer to each other than to any other node in the network, otherwise a low RSS will be observed by an honest node. We assume random node compromise, so the probability that malicious nodes are frequently near each other is low. In the case that it does occur, we stress that our goal is to establish multiple node-disjoint paths for use in multi-path protocols, and such protocols tolerate a certain threshold of compromised paths.

Theorem 13. *If two nodes successfully complete the FMAP protocol using key K, then both nodes must be in possession of K and be within each other's neighborhoods, except with very small probability.*

Proof. The identity, uniqueness, and uniformity assumptions combined ensure that the protocol can only be successfully completed if both nodes know K, or can guess it, and if the fingerprints on each message are as expected. Assuming the underlying mutual authentication protocol is secure, an adversary is only successful if it can guess K and can successfully spoof fingerprints. Danev and Čapkun [31] claim an error rate as low as 0.24% on fingerprint recognition. □

Some problem settings consider nodes that can lower their communication range in an effort to reduce power consumption, thus violating the uniformity assumption. If nodes were allowed to selectively reduce their communication range then two nearby malicious nodes may be able to exchange messages without detection by others. This ability would prevent us from proving Theorem 13.

4.3.2 Key Pre-distribution

We established in the previous section that a node can recognize two messages from the same source, that a node can commit to at most one identity, and that two nodes can verify a shared key. These assumptions do not, however, allow us to actually verify the identity (and key list) of a node. Because a given node will be relying on its shared-key neighbors to forward messages to those nodes with which it does not share a key, and also to nodes outside of its communication range, it is desirable for a node to be able to prove its identity (and therefore its key list), thus proving who it can communicate with securely. To aid in solving this problem, we use a key pre-distribution scheme (KPS) that allows for a node to be uniquely identified by the keys that it possesses. For example, the linear LS-KPS scheme of Lee and Stinson [61] allows a node to be uniquely identified by knowledge of any two keys that it possesses, while maintaining practical system parameters. This system was presented in detail in Section 2.4, and was generalized in the following chapter to allow for additional flexibility in parameter choices. The flexible variants considered in Chapter 3 are compatible with the assumptions used here, but for ease of presentation we use the basic linear scheme.

The Linear LS-KPS

For a network of size $N = b$, let k be the number of keys issued to each node, let r be the number of nodes possessing a given key, and let v be the total size of the key pool. A linear (v, b, r, k)-LS-KPS can be derived for certain values of v, b, r, and k. More specifically, let p be a prime power such that $2 \leq k \leq p$. Then Lee and Stinson [61] show that a (kp, p^2, p, k)-LS-KPS can be constructed. To construct such a scheme, let $X_1 \subseteq \mathbb{F}_p$ such that $|X_1| = k$. Each node will be issued a label (identifier) $(a, b) \in \mathbb{F}_p \times \mathbb{F}_p$ which determines the set of key identifiers $A_{a,b} = \{(x, ax + b) : x \in X_1\}$.

Two nodes $n_{a,b}$ and $n_{a',b'}$ can determine if they share a key through the following:

1. If $a = a'$, then $n_{a,b}$ and $n_{a',b'}$ do not share a common key.
2. If $a \neq a'$, then compute $x = (b' - b)(a' - a)^{-1} \in \mathbb{Z}_p$. If $x \in X_1$, then $n_{a,b}$ and $n_{a',b'}$ share the common key $K_{x,ax+b}$. Otherwise $n_{a,b}$ and $n_{a',b'}$ do not share a common key.

As an example of the linear LS-KPS, let $p = 47$ and $k = 30$. Then $v = kp = 1410$, $n = p^2 = 2209$ and a linear $(1410, 2209, 47, 30)$-LS-KPS can be constructed. Thus, if each node stores just 30 keys, then we can accommodate up to 2209 nodes while maintaining the property that a node is uniquely identified by any two keys that it possesses.

The LS-KPS is well suited to sensor networks as two nodes can efficiently determine whether or not they share a key. Shared-key discovery can be performed using only two integer subtractions and one inverse in $\mathbb{F}_p$, where p is very small. Furthermore, the multiple-space variant of the LS-KPS, presented earlier as the MS-KPS in Section 2.7, can be utilized here for a significant boost in resilience.

4.3.3 Localization and Directional Antennas

Most sensor networks feature nodes that broadcast and receive messages using omnidirectional antennas, with their communication region being a sphere of radius d centered around the node. A more recent area of study has been the investigation of sensor networks utilizing directional antennas. Although we assume omnidirectional broadcast for communication, there are tools that have been developed using directional antennas that are of interest, the most relevant being the localization methods by Ash and Potter [6]. Utilizing received signal strength and angle of arrival at multiple directional antennas on a single node, a node is able to estimate the position of a neighbor to an accuracy of less than one meter in some cases, with the ability to estimate the angle of arrival of an incoming signal to within 3 degrees. Ash and Potter's methods are of particular interest as they do not rely on any trusted *anchor nodes* possessing additional hardware, such as a GPS device. Thus, their localization method is referred to as *self-localization*, as it does not require any additional trust between nodes.

The later phases of our network discovery protocol assume that nodes have the ability to learn basic location information about their neighbors to within an error of ϵ, where ϵ is significantly smaller than the communication range d of each node. This information is utilized to ensure that nodes are physically located in a way that is consistent with their claimed view of the network. Directional antennas are one method of achieving this, but our protocols do not require that nodes have the ability to send directional messages. Nodes could, for example, be equipped with multiple small antennas such that differences in received signal strength at each antenna is enough to infer the approximate incoming angle and distance of an incoming message.

4.4 Proposed Solution

4.4.1 Phase 1: Identifying Neighbors

The first goal of each node is to discover which nodes are within its neighborhood and to verify their identities. In order to achieve this, each node will commit to an identity and then perform the FMAP protocol described in Section 4.3.1 with each neighbor it is supposed to share a key with. Each node then broadcasts the result of the FMAP protocol to all other nodes in their neighborhood so that they may decide whether or not to trust each node. For simplicity, we assume all nodes come online at the same time. Adding nodes into an existing network will be discussed later. Let n_i be the node running the protocol.

1. Broadcast a `HELLO` message containing the identity of node n_i.
2. For each `HELLO` message received, record the fingerprint and identity, making sure that the fingerprint has not already been seen. From this point on, fingerprints are verified for each received message.
3. For each neighboring node n_j with which a key is supposed to be shared with node n_i, broadcast (`BEGIN FMAP`, n_i, n_j) and run the FMAP protocol to verify the key.
4. (a) If FMAP is successful, broadcast `ACCEPT KEY` n_j.
   (b) If FMAP is not successful, broadcast `REJECT KEY` n_j.
5. (a) If a neighbor n_j of n_i begins the FMAP protocol with n_k, record the tuple (n_j, n_k).
   (b) If n_j broadcasts `ACCEPT KEY` n_k, verify that the tuple (n_j, n_k) has been recorded, and if so, add `ACCEPT KEY` to the tuple.
   (c) If n_j broadcasts `REJECT KEY` n_k, verify that the tuple (n_j, n_k) has been recorded, and if so, add `REJECT KEY` to the tuple.
6. Wait until no new nodes have come online and no new `ACCEPT KEY` or `REJECT KEY` votes have been heard for a period of time t_0.

7. Mark any run of FMAP that has yet to complete as `REJECT KEY`.

8. Apply the "accept rule" (discussed below) to each n_j and broadcast the result as `ACCEPT ID` n_j or `REJECT ID` n_j.

At the end of this process, each node n_i has a list of those nodes in its neighborhood (both identity and fingerprint), has verified those nodes with which it shares a key, and knows the result of all other FMAP protocols run by its neighbors. It remains to be decided when a node should accept the identity of each of its neighbors (i.e., what the "accept rule" is). The parameter t_0 represents a timeout value, meant to determine whether or not nodes in the neighborhood are still in the process of completing their runs of the FMAP protocol. Figures 4.2 and 4.3 demonstrate the process of neighbor discovery.

Recall that the linear LS-KPS allows one node n_i to uniquely identify any other node n_j if it can be convinced of any two keys that n_j possesses. A consequence of this is that each node can verify at most a single key with any other node, for if the nodes had two keys in common this would imply that they were the same node. Similarly, if a malicious node attempts to lie about its identity then it possesses at most one key associated with the false identity it has assumed.

Definition 9. *A set of nodes casting a vote for n_j is said to form a t-Local Honest Majority (t-LHM) if:*

1. *There are more `ACCEPT KEY` votes than `REJECT KEY` votes;*

2. *at least t distinct keys were verified by nodes casting votes; and*

3. *if $\mathcal{K} = \{k_1, \ldots, k_m\}$ are the different keys verified with n_j using FMAP, then the previous two properties are satisfied for the set $\mathcal{K} \setminus k_i$ for all $k_i \in \mathcal{K}$.*

In other words, a set of voting nodes forms a t-LHM if a majority of the nodes have cast an `ACCEPT KEY` vote over t different keys, even when all nodes possessing any single key are removed from the vote. This property is intended to capture the fact that, if a malicious node attempts to lie about its identity, then it can have at most one key consistent with that identity. Therefore, it can falsely convince all honest nodes possessing that one key to accept its false identity.

Theorem 14. *A node can cast at most one `ACCEPT KEY` or `REJECT KEY` vote for another node. Furthermore, a node can only vote for another node with which it supposed to share a key and with which it has run the FMAP protocol.*

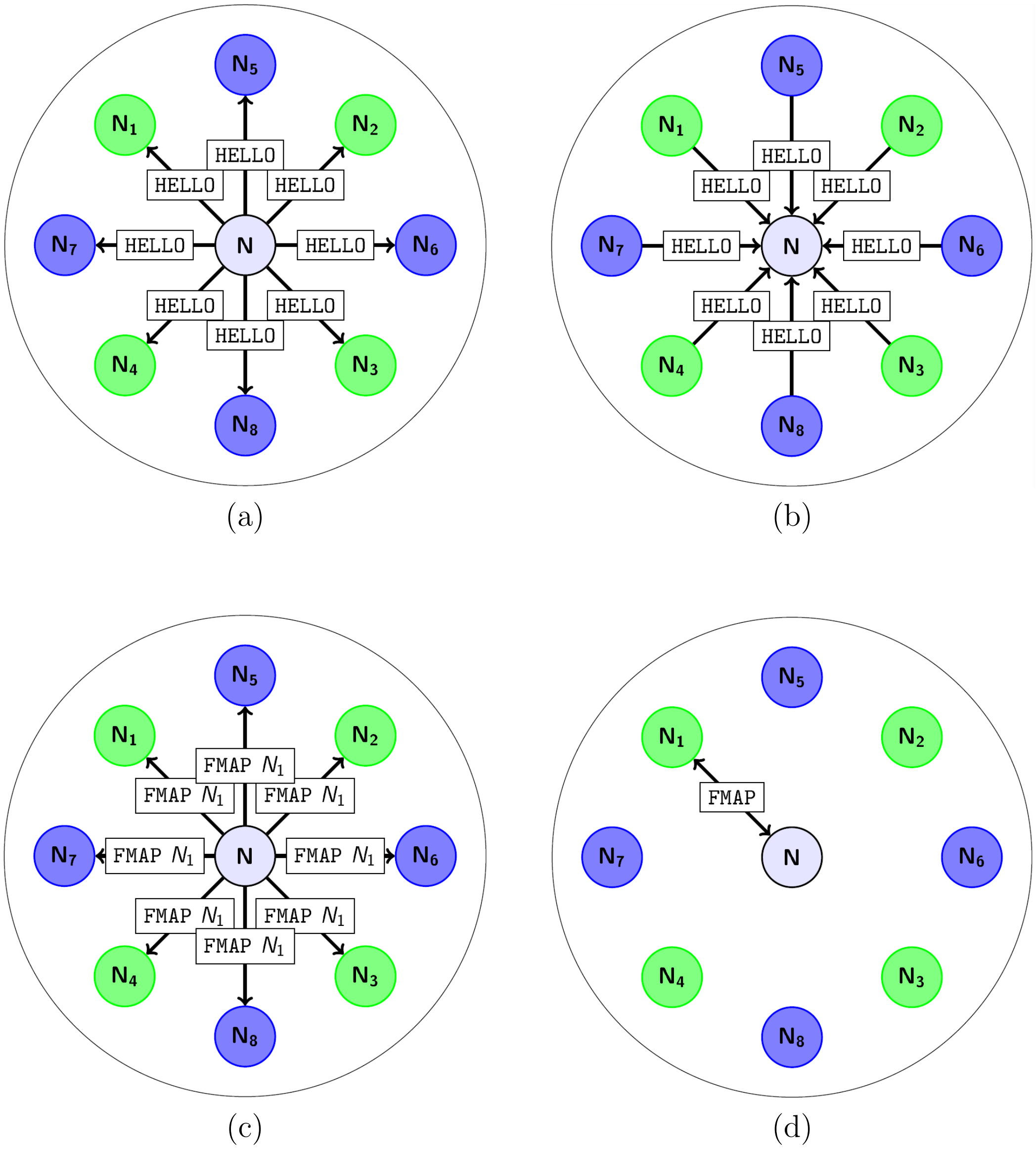
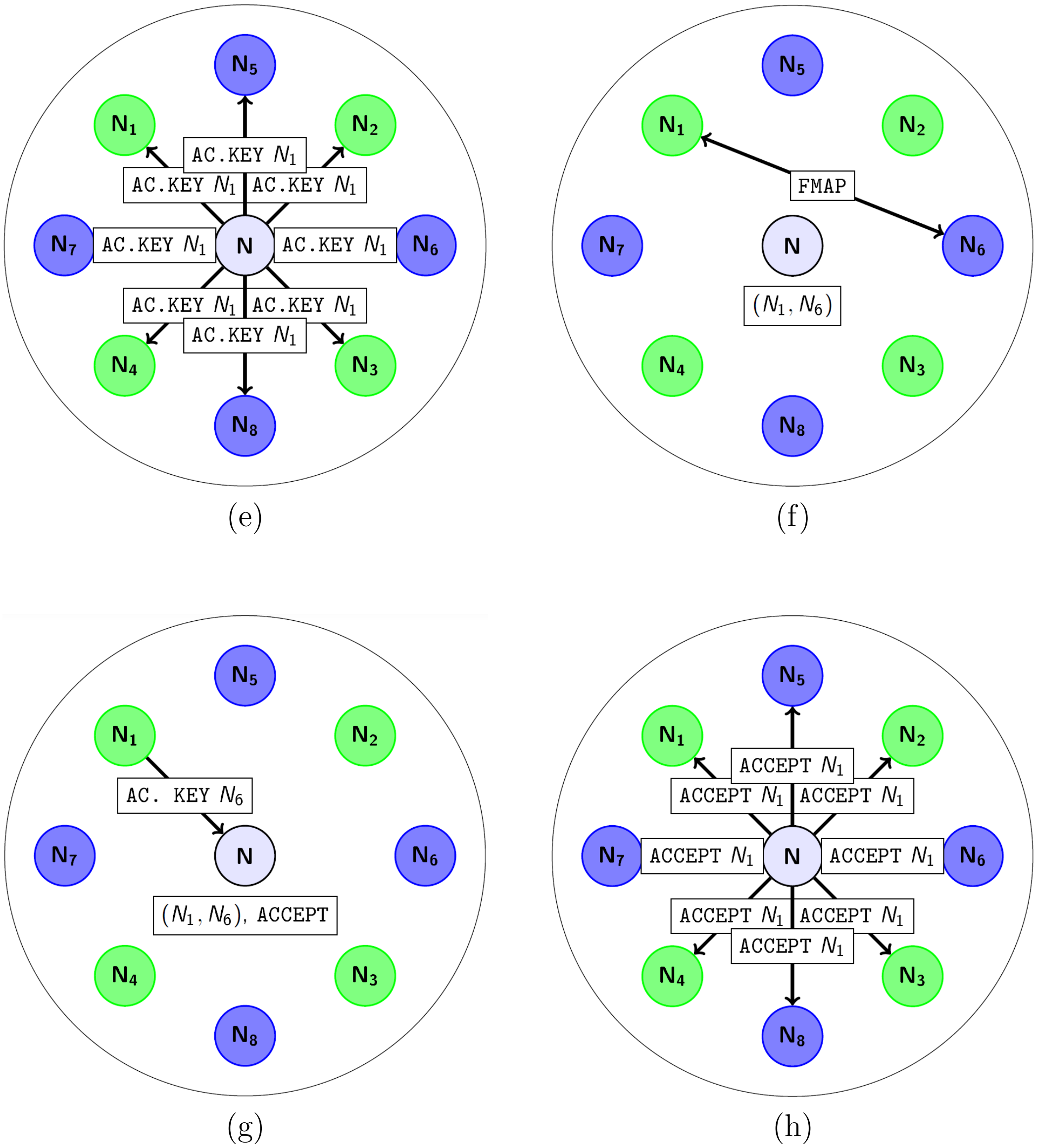


Figure 4.2: A pictorial demonstration of neighbor discovery. (a) The central node N sends a HELLO message to identify itself to its neighbors. (b) It overhears a similar message from each neighbor. (c) Node N then announces that it is authenticating node N_1, and (d) performs the FMAP protocol. (Continued in Figure 4.3.)



(e)

(f)

(g)

(h)

Figure 4.3: (Continued from Figure 4.2.) A pictorial demonstration of neighbor discovery. (e) Node N informs its neighbors that it successfully completed the FMAP protocol with node N_1. (f) Node N records each instance of the FMAP protocol it overhears, and (g) appends the record with the announced result. (h) Finally, once each FMAP run is complete, node N announces whether or not it accepts the identity of node N_1, thus informing its neighbors that it can provide a route to node N_1.

Proof. In Step 3 of the protocol, each node broadcasts the fact that it is beginning the FMAP protocol with another node. Each node that overhears a run of the protocol records this fact in the tuple (n_j, n_k). Because the list of keys given to each node is derived from its identity, it can easily be verified that n_j and n_k are supposed to share a key. By the uniformity assumption, if both n_j and n_k are in a node's neighborhood, then it will overhear each step of the FMAP protocol between them. Thus, only those tuples/votes that are valid with respect to the previous observations will be used to establish a node's identity. The fact that a node may cast at most one vote follows directly from the uniqueness and identity assumptions. □

Note that Theorem 14 does not guarantee that the FMAP protocol was successful (or not) between nodes n_j and n_k, only that n_i can observe that the appropriate messages were exchanged and that only one vote was cast by each. It is possible that two malicious nodes, one or both lying about their identities, could fake the FMAP protocol and cast a false `ACCEPT KEY` vote for each other. Similarly, a malicious node may successfully complete the FMAP protocol with another but vote `REJECT KEY` regardless. A malicious node could also announce it was beginning the protocol with a node that it cannot talk to, and then announce the result. This behavior is undetectable by n_i, but will be addressed in the next phase of the protocol.

Definition 10. *The t-LHM-Accept Rule:*

1. *If n_i is supposed to share a key with n_j and it does not complete the FMAP protocol with n_j, then reject n_j.*

2. *If the set of nodes casting a vote for n_j form a t-LHM, then accept the identity of n_j as valid;*

3. *otherwise, reject the identity of n_j.*

The t-LHM-Accept Rule relies on the fact that if there are enough honest nodes in n_i's neighborhood, then they can always achieve a majority when voting on n_j's honesty. In order to prove the security of the protocol, we must put a bound on the number of malicious nodes participating in a given vote.

Assumption 4. ***(LHM)*** *Let $\mathcal{K}$ be the set of keys possessed by n_j that are also possessed by common neighbors of n_i and n_j (including the shared key between n_i and n_j if it exists). We assume that $|\mathcal{K}| \geq 3$ and there are more honest nodes in the common neighborhood possessing keys from the set $\mathcal{K} \setminus k_i$ for all $k_i \in \mathcal{K}$ than malicious nodes.*

This assumption is intended to capture a necessary condition for the 2-LHM-Accept Rule to be a sufficient condition for determining a neighbor's honesty.

Theorem 15. *If a node n_j is accepted by 2-LHM-Accept and the LHM assumption holds, then n_j is not lying about its identity.*

Proof. We first note that, if n_j fails the FMAP protocol with n_i, then it is immediately rejected by n_i. Assume n_j is lying about its identity; then n_j has at most one key K consistent with its claimed identity. Also, recall that, by Theorem 14, each node can cast at most one vote. The 2-LHM property will only be satisfied if the number of `ACCEPT KEY` votes from nodes not possessing K is greater than the number of `REJECT KEY` votes, and at least two other keys are verified. Each honest node that does not possess K will cast one `REJECT KEY` vote for n_j, and in the worst case each malicious node not possessing K will cast at most one `ACCEPT KEY` vote. By the LHM assumption, the number of `REJECT KEY` votes is greater than the number of `ACCEPT KEY` votes among neighbors not possessing K. Hence, the 2-LHM-Accept rule is not satisfied and n_j's identity will not be accepted. □

The LHM assumption also places a limit on the capability of malicious nodes to prevent honest nodes from being accepted.

Theorem 16. *If the LHM assumption holds, then a coalition of malicious nodes cannot force an honest node n_j to be rejected by n_i with the 2-LHM-Accept Rule.*

Proof. If an honest node n_j is rejected, then it must have received more `REJECT KEY` votes than `ACCEPT KEY` votes. Recall that, by Theorem 14, each node can cast at most one vote. Because n_j is honest, each `REJECT KEY` vote must have come from a distinct malicious node. The LHM assumption guarantees that there are always more honest nodes than malicious nodes in situations considered by the 2-LHM-Accept Rule, and hence, there can never be more `REJECT KEY` votes than `ACCEPT KEY` votes for an honest n_j. □

Theorems 15 and 16 together prove the correctness of our neighbor discovery protocol. It is worth noting that, if n_i completes the FMAP protocol with n_j, then n_i has directly verified that n_i possesses a single key K consistent with its claimed identity. This fact means the 1-LHM-Accept Rule is sufficient in the case where n_i and n_j share a key, with the caveat that 1-LHM must verify a key different from K.

Phase 1 does not rely on any sort of location information to authenticate neighbors, but Phase 2 requires a modification to include location information to ensure nodes claiming to have a route to a 2-hop neighbor are not lying. This is accomplished by assuming each node can compute an approximate location for each of its neighbors. An interesting direction for future work is to determine if nodes can estimate each other's position by including received signal strength (RSS) measurements along with their public `accept` during Phase 1. With a complete set of relative RSS measurements between neighbors there may be sufficient location information to verify a node's location for Phase 2.

4.4.2 Phase 2: Identifying 2-Hop Paths

At the conclusion of Phase 1, each node has a list of its neighbors and has decided whether or not to accept their claimed identities. From this point on, we assume that an honest node n_i will ignore the existence of any node n_j it could not verify during Phase 1. To simplify presentation we also assume that n_i can establish a path to any neighbor it does not share a key with via its local neighbors. Table 2.3 from Chapter 2 shows the probability that 2-hop paths exist to any neighbor for some example parameters, and longer length paths would be possible to calculate from n_i's verified neighbors, thereby increasing the likelihood a local path can be established. Our goal in this section is to establish 2-hop neighbors that exist outside of n_i's neighborhood (i.e., paths of the form $n_i \to n_j \to n_k$).

Verifying the identity of n_i's 2-hop neighbors is different than identifying nodes in n_i's neighborhood, as we lose the ability to verify fingerprints on n_k, as well as the ability to overhear whether or not the FMAP protocol was actually executed. Furthermore, we cannot assume that an intermediate node n_j is honest, only that it is not lying about its identity. Recall that in Step 5a of Phase 1, node n_i records the tuple (n_j, n_k) when the FMAP protocol is initiated by a neighbor. If n_k is also in n_i's neighborhood, then the corresponding tuple (n_k, n_j) is also recorded. Thus, during Phase 1, node n_i also learns which nodes n_k lie outside of its neighborhood such that n_j has successfully completed the FMAP protocol with n_k, and whether or not they were accepted in Step 8. Such nodes are referred to as *2-hop neighbors*. We can use this information to apply a voting protocol similar to Phase 1.

Our main problem during this phase is identifying those nodes that are attempting to vote on the identity of n_k, but who are not actually within n_k's communication range. Let $\mathcal{N}_i$ denote n_i's communication range (i.e., the nodes in n_i's neighborhood). Any node in $\mathcal{N}_i$ can cast a vote for n_k, but only those nodes that lie in the region $\mathcal{N}_i \cap \mathcal{N}_k$ are capable of routing a message between n_i and n_k. Nodes in the region $\mathcal{N}_j \setminus \mathcal{N}_k$ are unable to route a message to n_k, but learn of n_k via n_j during Phase 1. Nodes in the region $\mathcal{N}_i \setminus \mathcal{N}_j$ are unable to route a message to n_k, and do not overhear n_j during Phase 1 (but could overhear a node in $\mathcal{N}_j \setminus \mathcal{N}_k$ that repeats the fact). A malicious node in $\mathcal{N}_j \setminus \mathcal{N}_k$ could falsely

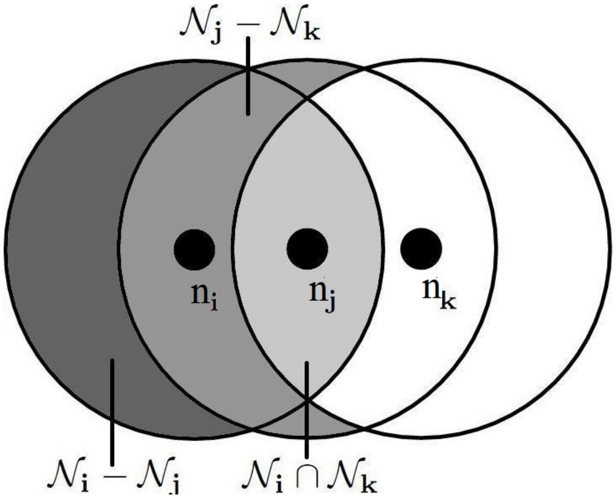

$\mathcal{N}_\mathbf{j} - \mathcal{N}_\mathbf{k}$
$\mathrm{n_i}$
$\mathrm{n_j}$
$\mathrm{n_k}$
$\mathcal{N}_\mathbf{i} - \mathcal{N}_\mathbf{j}$
$\mathcal{N}_\mathbf{i} \cap \mathcal{N}_\mathbf{k}$

Figure 4.4: Only nodes in $\mathcal{N}_i \cap \mathcal{N}_k$ are capable of routing a message, but malicious nodes in $\mathcal{N}_j - \mathcal{N}_k$ learn about n_k from n_j, and nodes in $\mathcal{N}_i - \mathcal{N}_j$ learn about n_k from malicious nodes in $\mathcal{N}_j - \mathcal{N}_k$, thus enabling them to falsely claim they can route to n_k.

broadcast that it is beginning the FMAP protocol with n_k and then cast a vote. Similarly, if a node in $\mathcal{N}_j \setminus \mathcal{N}_k$ falsely begins the protocol, then a malicious node in $\mathcal{N}_i \setminus \mathcal{N}_j$ learns of n_k and can do the same. The fact that any of n_i's neighbors is capable of learning of n_k's existence means that, in the worst case, every malicious node in n_i's neighborhood that shares a key with n_k will cast a false vote.

Clearly, if there are more honest nodes in $\mathcal{N}_i \cap \mathcal{N}_k$ that can route messages between n_i and n_k than malicious nodes in $\mathcal{N}_i$, then the malicious nodes cannot cast enough `REJECT ID` votes to prevent n_k from being accepted. Unfortunately, the malicious nodes can still falsely accept n_k's identity, thus suggesting a secure path exists where it actually does not. For this reason, we have each n_j include localization information about n_k, allowing malicious nodes to be detected in many cases.

Assumption 5. *The* Honest 2-Hop Majority (H2M) *assumption states that there are more honest nodes in $\mathcal{N}_i \cap \mathcal{N}_k$ that can route messages between n_i and n_k than malicious nodes in $\mathcal{N}_i$.*

Let the protocol in Phase 1 be altered such that each node includes information on the location of n_j (i.e., the relative angle and distance) when it broadcasts its `ACCEPT ID` vote. Furthermore, we assume that n_i can compute similar localization information on n_j as needed. Then the protocol for identifying 2-hop neighbors from n_i's point of view is:

For each 2-hop neighbor n_k:

1. For each accepted neighbor n_j casting an `ACCEPT ID` vote for n_k:
   (a) Compute (n_j, n_k, `ACCEPT ID`, loc(n_k)), where loc(n_k) is the location of n_k relative to n_i.
2. For each neighbor n_j casting a `REJECT ID` vote for n_k:
   (a) Compute (n_j, n_k, `REJECT ID`, loc(n_k)), where loc(n_k) is the location of n_k relative to n_i.
3. Wait until no new votes for n_k have been received for a pre-specified period of time t_1.
4. Let $\mathcal{S}_k$ be the largest subset of tuples from Steps 1 and 2 such that all estimates of loc(n_k) are within 2ϵ of each other and $n_j \in \mathcal{N}_i \cap \mathcal{N}_k$ (according to the consensus on n_k's location), where ϵ is the maximum error of localization.
5. (a) If there are more `ACCEPT ID` votes in $\mathcal{S}_k$ than `REJECT ID` votes, then accept all n_j in $\mathcal{S}_k$ who cast an `ACCEPT ID` vote as being able to route messages to n_k. Broadcast the message `ACCEPT ROUTE` n_j`,`n_k.
   (b) If there are more `REJECT ID` votes in $\mathcal{S}_k$ than `REJECT ID` votes, then do not accept n_k as being an honest node and do not include them in any path.

Figure 4.5 demonstrates the protocol. As with the protocol in Phase 1, this protocol relies on the fact that a majority of the nodes voting on n_k are honest. If this assumption holds, then there will exist a majority of nodes that agree on n_k's position, who also lie within communication range of n_i and n_j. The parameter t_1 is a timeout value used to ensure all votes for n_k have been received before proceeding.

Theorem 17. *If the H2M assumption holds, and if $n_j \notin \mathcal{N}_i \cap \mathcal{N}_k$, then $n_i \to n_j \to n_k$ will not be accepted as a valid 2-hop path, except with small probability.*

Proof. In Step 4 of the protocol, $\mathcal{S}_k$ is chosen to be the largest set of n_j such that each n_j agrees on n_k's position, and lies in the intersection of n_i and n_k's communication range. By the H2M assumption, the honest n_j's will agree on n_k's position with accuracy 2ϵ, thus defining the common intersection of n_i and n_k's communication range to within 2ϵ. Hence, any node lying about its ability to communicate with n_k must be within 3ϵ of n_k's communication range, as we can only estimate n_j's position with accuracy ϵ. □

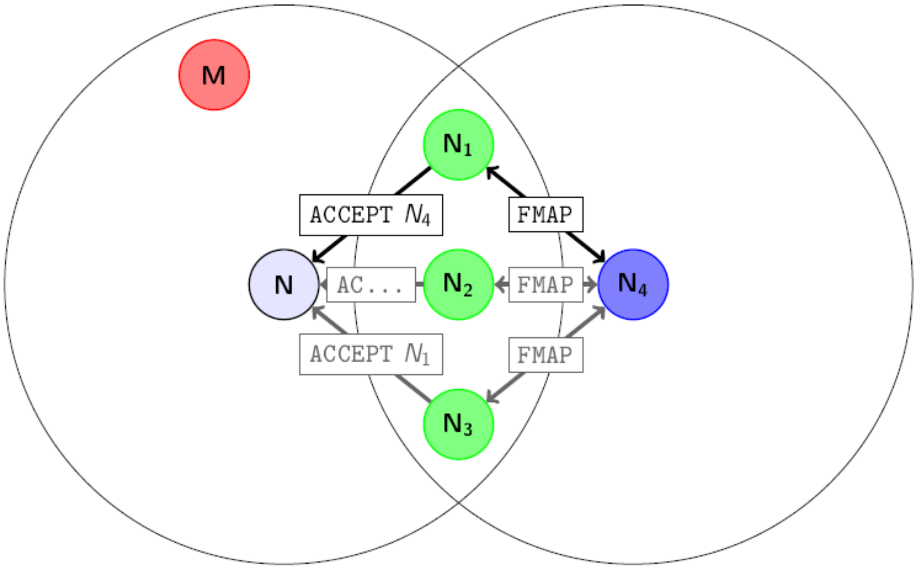


Figure 4.5: An example of two-hop neighbor discovery. Node N learns about node N_4 (and its location) via nodes N_1, N_2, N_3. The honest nodes form a majority, and prevent malicious node M from claiming it can route to node N_4.

Theorem 17 allows us to conclude with high probability that only votes from those nodes that are actually capable of communication with n_k will be included in the set $\mathcal{S}_k$.

Theorem 18. *If the LHM and 2HM assumptions are satisfied, and n_k is accepted by the protocol, then n_k exists and is not lying about its identity.*

Proof. This follows from Theorem 15 and the fact that by the 2HM assumption a majority of the n_j in $\mathcal{S}_k$ are honest. If n_k is lying about its identity, then each honest n_j will have detected this in Phase 1 and broadcasted a reject vote. Hence, n_k has proven its identity to each honest n_j in $\mathcal{S}_k$. □

Note that, during Phase 2, a node discovers if any malicious node attempted to claim it could route messages to n_k when it was not in communication range. Thus, we can mark such nodes as untrusted and not consider any path containing them in the future.

4.4.3 Phase 3: Beyond 2-Hop Paths

We now consider how a node can extend its knowledge of the network to longer paths. At the end of Phase 2, each node has determined its 2-hop neighbors and broadcasts that fact publicly. Additionally, any malicious node that has claimed a false identity, or the ability to communicate with a 2-hop neighbor when it cannot, has been discovered, except with small probability. We continue from this point assuming that each node has established a path to each of its neighbors, as well as to its 2-hop neighbors, and that each of these paths are valid.

In Phase 1, each n_i learns paths of the form $n_i \rightarrow n_j$, and in Phase 2 each n_i learns paths of the form $n_i \rightarrow n_j \rightarrow n_k$. Just as nodes informed their neighbors of the results of Phase 1 so that the information could be utilized to construct 2-hop paths, each node broadcasts the results of Phase 2 so that nodes if their neighborhood learn which 3-hop paths exist. More specifically, each n_j will broadcast all paths it has discovered of the form $n_j \rightarrow n_k \rightarrow n_l$.

Recall that each n_j accepted in Phase 2 must have proven they are capable of communication with n_k (by proving their identity in Phase 1), that they lie within n_k's communication range, and also that the n_j's form an honest majority. Therefore, if a majority of the nodes n_j that are capable of communicating with n_k broadcast knowledge of a route n_l via n_k, then n_i can conclude that said route does in fact exist. n_i's view of the network can be updated accordingly, and n_i can inform its neighbors of the fact that a secure path to n_l has been determined.

From n_i's point of view, the process is as follows:

For each 3-hop path $n_i \rightarrow n_j \rightarrow n_k \rightarrow n_l$

1. Wait until no new 3-hop paths to n_l via n_k have been received for a period of time t_2.
2. Let $\mathcal{S}_k$ be the set of n_j capable of routing a message to n_k.
3. Let $\mathcal{S}_l$ be the set of n_j capable of routing a message to n_l via n_k.
4. If $|\mathcal{S}_k \cap \mathcal{S}_l| > \frac{1}{2}|\mathcal{S}_k|$ broadcast `ACCEPT ROUTE` n_j,n_k,n_l for each route involving n_j and n_k;
5. otherwise, broadcast `REJECT ROUTE` n_j,n_k,n_l.

The same process can be followed for routes of arbitrary length. During each subsequent phase, node n_i increases its knowledge of the network by one hop by relying on the nodes that were verified during Phases 1 and 2 of the protocol. In practice, this process would be repeated until each sensor node has discovered a base station, the desired destination node, or for a fixed maximum number of hops to ensure it eventually terminates. Therefore, the total cost of network discovery is dependent on the depth to which a nodes wish to learn the topology of the network.

4.4.4 Adding New Nodes

In order to accommodate new nodes being added to the network, we observe that once network discovery has completed, each node has established its own view of the network. When a new node comes online, it can announce its presence through a `HELLO` message and engage in the FMAP protocol with its neighbors to establish its identity. This process is essentially re-running Phase 1 of the protocol, where, upon completion, each node in the neighborhood will accept or reject the new node's identity. By broadcasting these votes, surrounding nodes are informed of the new node and its status, and the information is propagated throughout the network as each node repeats whether or not it accepts or rejects the new node, as described in Phase 3. The new node can learn about the rest of the network by querying its neighbors. Because an honest majority is assumed, the new node will accept a path through the network as valid only if a majority of its neighbors agree on its existence.

4.5 Performance Analysis

Our protocol relies heavily on the assumption that there always exists an honest majority that can be trusted. In order to justify that such an assumption is practical, we investigate how likely it is that our LHM and 2HM assumptions are satisfied as more malicious nodes are added to a network. To do so, we run a Java simulation where 50 keys are assigned to 2000 randomly labeled nodes with communication radius 1 using a $(50 \cdot 149, 149^2, 149, 50)$-LS-KPS as described in Section 4.3.2, which are then distributed randomly over a square region according to a density parameter δ (i.e., we vary the size of the network to produce the desired density), such that we expect $\pi\delta$ nodes to lie in any circle of radius 1. Note that nodes around the perimeter of this area will have fewer nodes in their neighborhood than interior nodes on average. Our simulation includes results from all nodes in the network, including perimeter nodes.

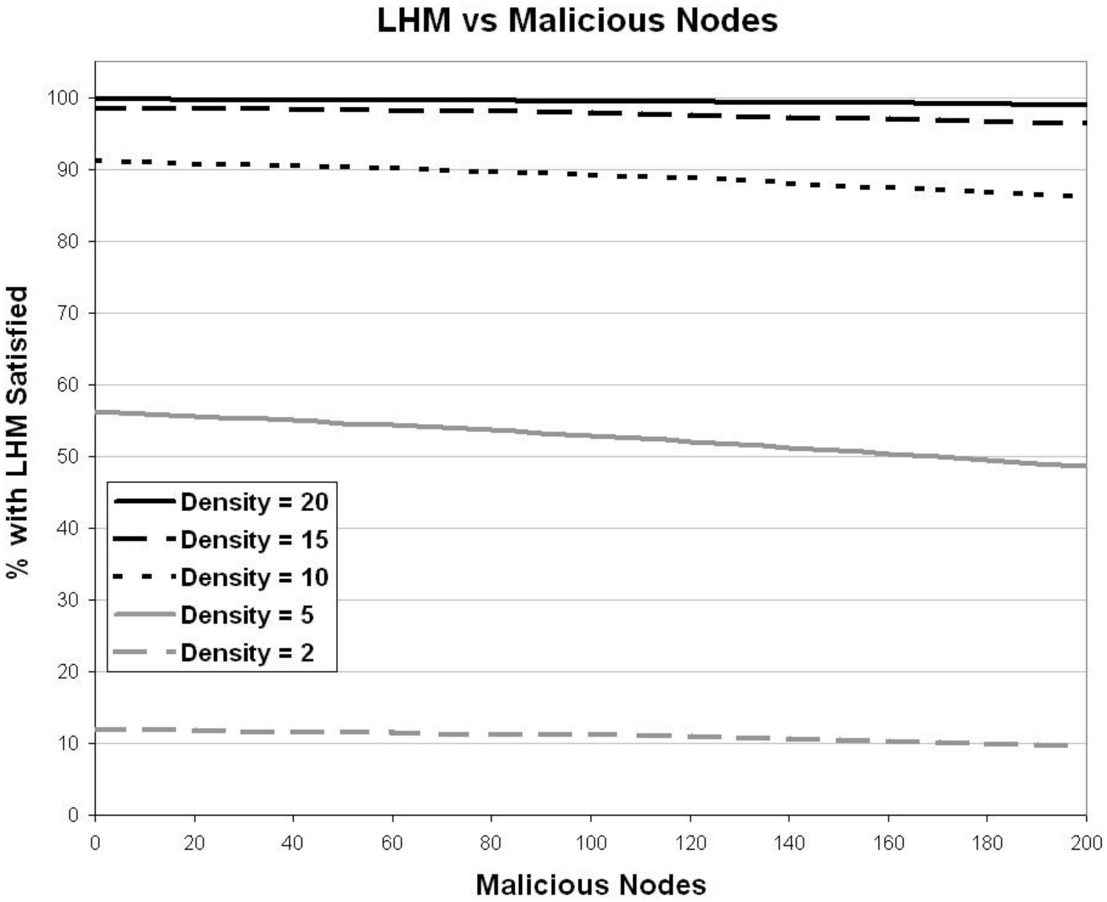


Figure 4.6: The probability that the LHM assumption is satisfied for a random network of 2000 nodes, each possessing 50 keys, for varying numbers of malicious nodes.

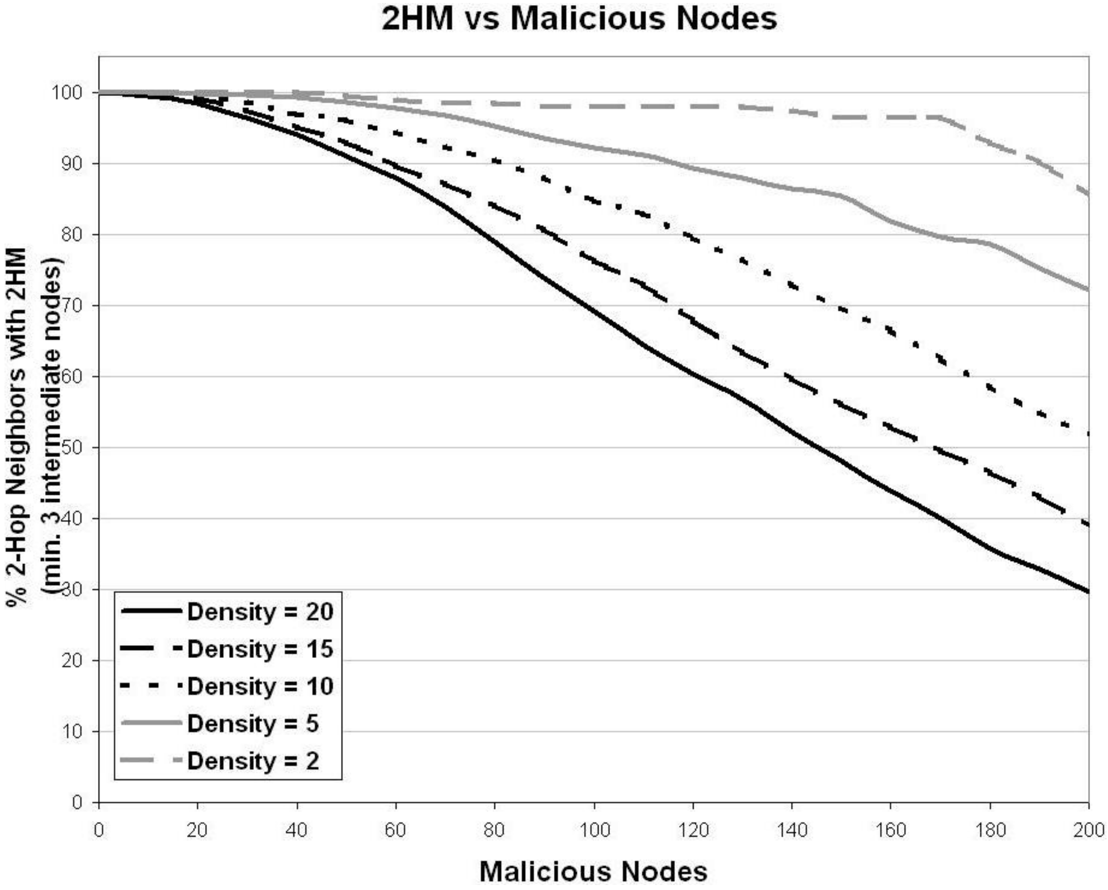


Figure 4.7: The probability 2HM assumption is satisfied for a random network of 2000 nodes, each possessing 50 keys, for varying numbers of malicious nodes. The graph only considers pairs of 2-hop neighbors such that at least three nodes exist that can route messages between them.

To test the LHM assumption we take every pair of neighbors (n_i, n_j) and test whether or not the nodes in n_i's neighborhood satisfy the LHM assumption with respect to n_j as a growing number of malicious nodes are added to the network. The graph in Figure 4.5 shows our results. We observe that, at low network densities, the LHM assumption is difficult to satisfy due to the fact that among n_i and its neighbors there must be at least three distinct keys shared with n_j. Once network density is sufficiently high, we see that the LHM assumption is satisfied nearly 100% of the time, even when more than 100 nodes present during network discovery are malicious.

To test the 2HM assumption we take every pair of 2-hop neighbors (n_i, n_k) such that there exist at least three different n_j capable of routing a message between n_i and n_k, and test whether or not the 2HM assumption is satisfied with respect to n_i and the set of n_j nodes as a growing number of malicious nodes are added to the network. We limit our analysis to situations where three or mode intermediate nodes exist because voting on the validity of a link does not make sense until at least three nodes are involved. The graph in Figure 4.7 shows our results. We observe that at low node density the 2HM assumption is satisfied with higher probability. This observation is due to the fact that situations where three intermediate nodes exist are somewhat rare, and a larger number of malicious nodes are required before it is expected that one lies within any given node's communication range. At higher densities we observe that the 2HM is still satisfied with high probability even when more than 50 nodes are malicious.

In order to study the effect of parameter choice on the probability that the LHM and 2HM assumptions are satisfied, we repeated the previous experiments for a variety of different parameter sets. The results are summarized in Figures 4.8 and 4.9 for the fixed density $\delta = 20$. We observe that performance is closely related to the ratio between k and p, which is reflected in the network itself by the probability of two nodes sharing a key increasing as the ratio of k to p approaches 1. When connectivity is high, more malicious nodes must be present before the LHM or 2HM assumptions are violated.

4.6 Summary and Remarks

We have considered a solution to the problem of discovering disjoint paths in a sensor network that assumes an active adversary is present in the network from the very beginning. Although several assumptions are required to ensure the protocol is secure, we demonstrate that practical tools, such as combinatorial key pre-distribution, fingerprinting, and self-localization, can be used to realize these assumptions. We also show that the resulting

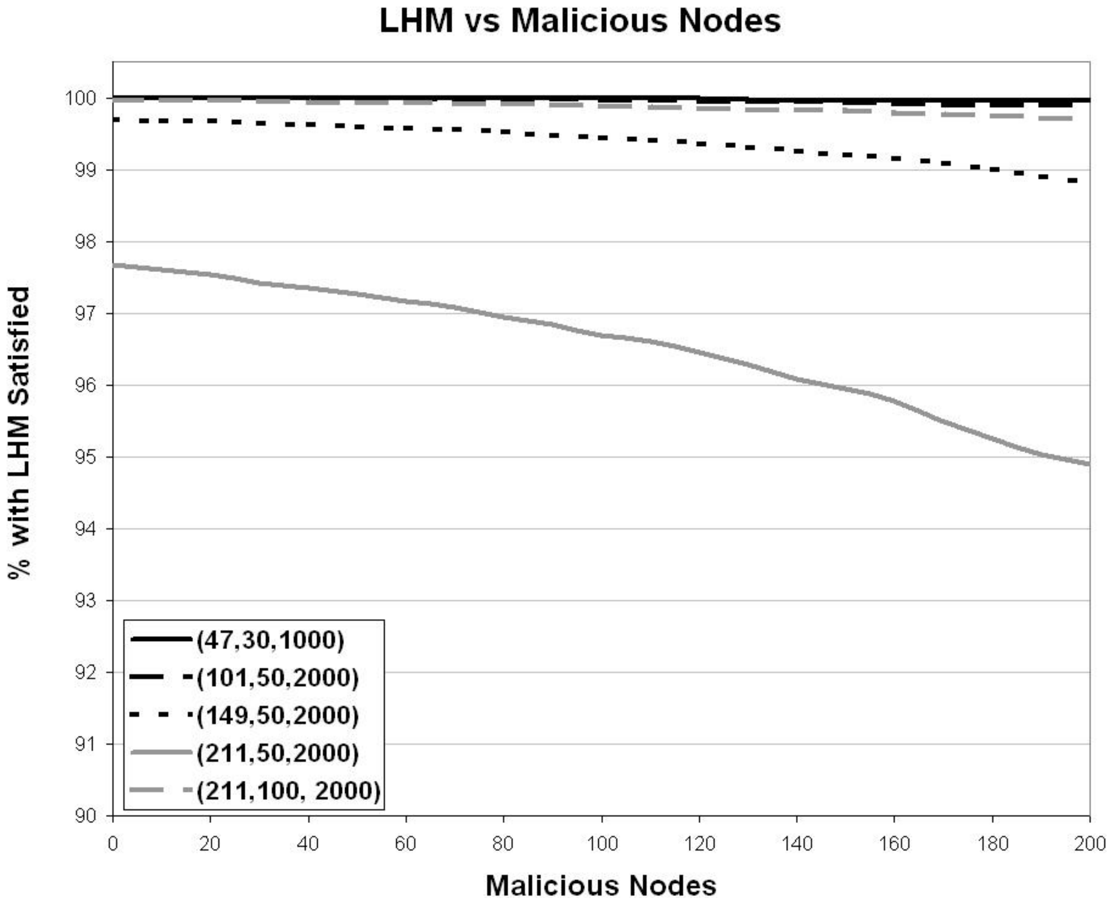


Figure 4.8: The probability that the LHM assumption is satisfied for a variety of different parameter sets using a fixed density of $\delta = 20$. Here, each data set is marked with the tuple (p, k, n), where n is the number of nodes.

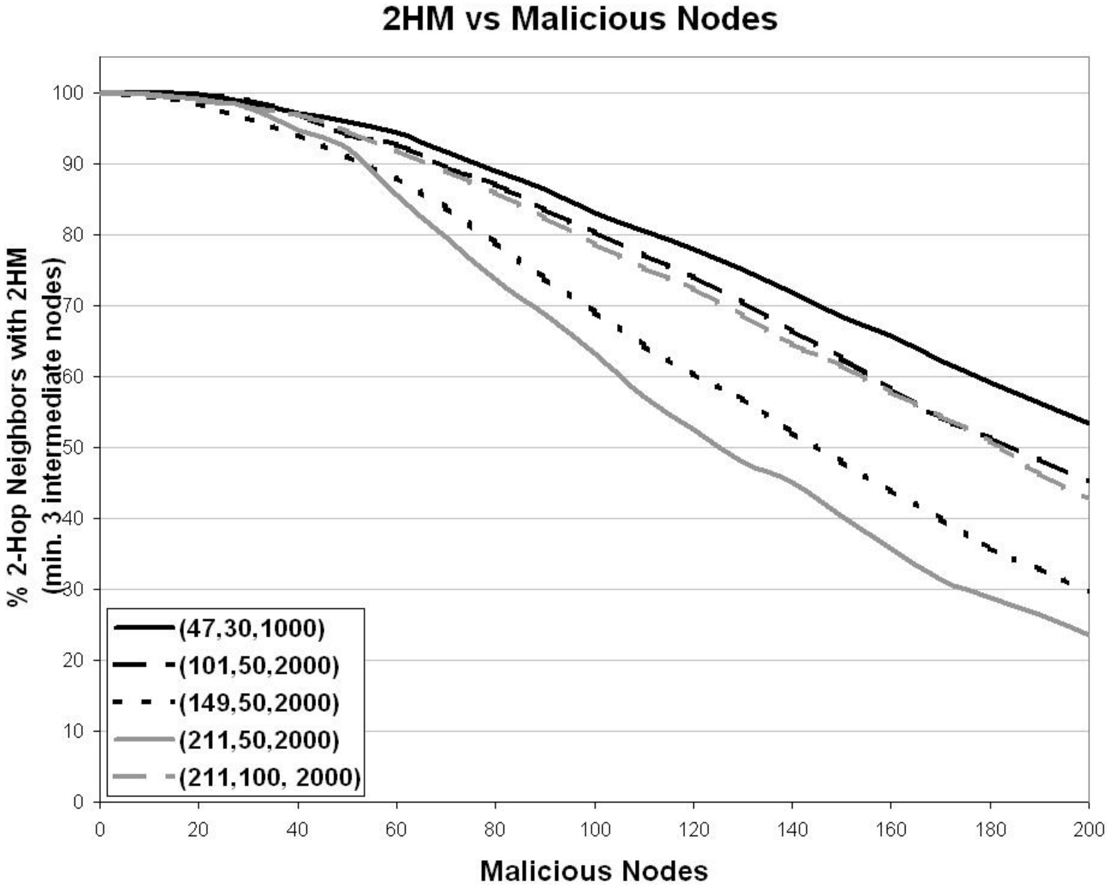


Figure 4.9: The probability that the 2HM assumptions are satisfied for a variety of different parameter sets using a fixed density of $\delta = 20$. Here, each data set is marked with the tuple (p, k, n), where n is the number of nodes. The graph only considers pairs of 2-hop neighbors such that at least three nodes exist that can route messages between them.

protocol is resilient in the presence of many adversarial nodes. Because we rely on fingerprinting and localization, there always exists a small probability that a malicious node could exploit the inherent error in these techniques, however, we emphasize that our goal was to establish disjoint paths so that perfectly secure message transmission protocols can be utilized. Such protocols are designed assuming an adversary controls up to one third of the paths between nodes.

It may be possible to strengthen this protocol by performing additional analysis on the votes received in Phases 1 and 2. A `REJECT` vote implies that at least one of the voters is malicious, but determining which node(s) solely from the votes overheard among nodes would allow for the detection of malicious nodes in some situations where they are not lying about their identity. It may also be possible to relax our need of localization in Phase 2 and instead rely only on distance bounds or angle of arrival information. These problems motivate future work in secure network discovery.

Chapter 5

Resilient Aggregation in Sensor Networks

With regards to the life cycle of a sensor network, the previous chapters have examined issues at the pre-deployment and deployment/setup phases of a network's lifetime. This chapter continues by considering the operational phase of a sensor network. In particular, the problem of *secure data aggregation* is considered.

At a high level, the goal of a sensor network is to collect sensor readings from some number of sensor nodes. This problem setting is distinguished from other ad-hoc networks by the fact that sensors are resource limited with respect to computational speed, memory, energy, and communication range. Typically, the most energy-intensive task a sensor networks performs is transmitting a message. For this reason, minimizing the total number of messages a node transmits is extremely important. Sensor networks have been proposed for a variety of applications, but each deployment tends to have the same goal of each sensor node measuring some aspect of its environment and forwarding this information to a *sink* or *base station.*

A common technique to preserve energy in a sensor network is to *aggregate* sensor readings as they travel from a node to the base station. If the application allows for messages to be combined in some manner on a hop-by-hop basis, then this achieves a significant decrease in the energy necessary to collect readings from the entire network. For example, if an application requires only statistical information about sensor readings, such as the sum, average, or sum of squares (standard deviation), then nodes can simply add their sensor readings into an aggregate total before passing it onwards.

This chapter investigates the aggregation problem in settings where an active adversary is present within the network, and whose goal is to alter the correct network-wide aggregate by a sufficient amount without being detected. Results are first presented for a special family of networks, known as linear networks. The resulting protocols are then used as building blocks to construct secure aggregation protocols for a much wider variety of network topologies. We first review the aggregation problem in more detail and summarize previous solutions to the problem. In particular, we consider two previous protocols in detail and discuss their shortcomings when applied to linear networks. Three natural key pre-distribution schemes for linear networks are identified, and three protocols for secure aggregation are presented based on these schemes. The role of linear subnetworks in both randomized networks and structured networks, such as a grid topology, is then discussed, in order to demonstrate the use of linear protocols in more general network settings.

5.1 The Aggregation Problem

Although this section's title suggests that aggregation is a "problem", in-network data aggregation is a solution widely used in sensor networks to conserve energy, and therefore prolong the lifetime of the network. Aggregation is simply the practice of combining, or aggregating, two or more messages into a single message as they travel toward a base station, thereby reducing the total amount of energy needed to deliver both messages. This practice introduces a loss of precision, as the base station receives $f(x, y)$, for some aggregation function f, instead of individual readings x and y. Many applications may only be concerned with aggregate statistics, such as the total number of observed events, average number of events, or the variance across the network, for example, so the loss of precision may be worth the energy savings. Figure 5.1 depicts one potential technique for aggregation. The shaded nodes collect readings from nearby nodes, apply an aggregation function, and send a single aggregate value onwards. This process allows the aggregate total for all 19 nodes to reach the base station using just 19 messages. If each node reported a reading individually, the same network would require 52 messages to deliver each individual reading.

Fasolo et al. [39] provide a comprehensive survey of existing aggregation techniques, and identify several important aspects of aggregation protocols for sensor networks. To begin, aggregation protocols can be designed with or without *size reduction*. An aggregation function, such as simple addition, allows two readings to be combined into a single reading of similar size. Such an approach only makes sense if all readings are drawn from the same domain, such as two temperature readings. Adding together a temperature and a humidity reading, for example, would not yield a useful result. An alternative is to simply

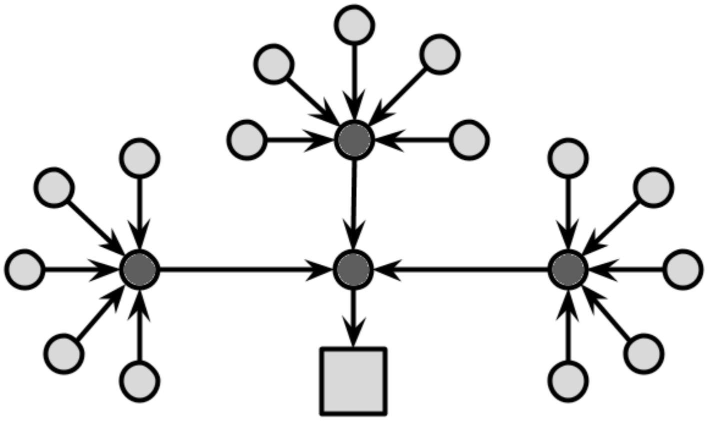


Figure 5.1: A simple aggregation tree within a network. Each of the darker shaded nodes aggregates the received readings, along with its own reading, and forwards the result as a single aggregate message toward the base station.

concatenate the two readings and forward them together as a single message or packet. This approach does not reduce the total size of the data, but instead reduces the overhead of delivering two separate messages. This chapter is mainly concerned with aggregation techniques that provide size reduction, but the solutions considered could be adapted to settings without size reduction.

A related aspect to size reduction is *lossless* vs. *lossy* aggregation functions. The addition function is lossy, as there is no way to determine individual sensor readings without extra information, while concatenation is lossless when messages are of a known size, as it can be trivially reversed. The protocols presented in this chapter are, in general, lossy protocols, but require nearby nodes to exchange enough information such that the aggregation function is lossless within a small enough radius in the network.

Another important aspect of an aggregation function is whether or not it is sensitive to duplicate readings, as this greatly impacts the design of an aggregation protocol. A function, such as addition, is *duplicate sensitive*, as including the same reading multiple times changes the result. Duplicate sensitive functions typically require information to follow only a single path to the base station, and therefore naturally yield tree-based aggregation topologies, such as the network in Figure 5.1. Alternatively, an aggregation function such as the boolean OR function, is *duplicate insensitive*, as for any reading x, it holds that x OR $x = x$. The protocols considered in this chapter are duplicate sensitive in general.

Various design considerations give rise to three distinct approaches to aggregation:

1. **Tree-based** - Nodes are organized into an aggregation tree, typically dictated by the underlying routing protocol. Each node receives aggregates from each of its children, combines them, and forwards the result to its parent. The Tiny Aggregation Protocol (TAG) [69] is an example of this approach.

2. **Cluster-based** - Nodes are placed, or self-organize, into clusters with a distinguished node acting as a clusterhead. Each node forwards its reading to the clusterhead, who then aggregates all readings and forwards the result to the base station. Although similar in spirit to tree-based approaches, cluster-based approaches are considered a distinct category because the group and cluster head selection typically require nodes to exchange messages with many other nearby nodes. An example of such an approach is the Low Energy Adaptive Clustering Hierarchy (LEACH) protocol [46].

3. **Diffusion-based** - Nodes use a duplicate insensitive aggregation function and forward readings down multiple potentially overlapping paths towards the base station, thus adding redundancy and avoiding potential single points of failure. These approaches often utilize the fact that wireless sensor nodes communicate over a broadcast medium, so messages can be sent to multiple neighbors with a single broadcast. Examples include Directed Diffusion [51,52] and Synopsis Diffusion [78].

5.1.1 Secure Aggregation

Although aggregation can be thought of as a solution that conserves energy, the “problem” considered here is to address the presence of an active adversary within the network, as aggregating data significantly increases the impact of even a single malicious node. Recall Figure 5.1 and assume one of the aggregating nodes is malicious. Each aggregator is responsible for forwarding the readings of at least five other nodes. Therefore, a malicious aggregator is granted a much higher influence than it would otherwise have, as it can arbitrarily alter the readings of five other nodes in the network. In the worst case, a malicious aggregator next to the base station may be able to arbitrarily set the aggregate total for the entire network. More specifically, we may expect a malicious node to do one or more of the following:

- Report an invalid sensor reading,

- Arbitrarily alter the aggregate total, or
- Deviate from the aggregation protocol (i.e., refuse to forward messages).

The goal of a *secure aggregation protocol*, is to protect against one or more of these, or other potential malicious behavior. The specific goal of this chapter is to defend against both of the first two behaviors. Beyond detection, addressing deviations from the protocol are outside the scope of this work. Nothing at the protocol level can prevent a malicious node from refusing to participate or broadcasting random noise. In other words, the goal is a protocol that limits the actions of a malicious node that wishes to modify the aggregate total by more than a single valid sensor reading. Note that unless additional assumptions are made about the distribution of nodes or the environment, such as the property that any given reading will be sensed and reported by at least two nodes, it is impossible to force a malicious node to report its true reading, as opposed to a valid possible reading. It is also the nature of sensor networks that nodes are expected to fail over time, either during deployment (dropped from an airplane, for example), due to environmental issues, or as their batteries fail. Thus, even in a non-adversarial model, there is motivation to verify the integrity of data.

The above descriptions are concerned with the correctness of the final aggregate total, and with limiting the ability of a malicious node to modify it. Secure aggregation protocols that are concerned only with the integrity of the aggregate total are often referred to as *resilient data aggregation protocols*. An orthogonal family of secure aggregation protocols, known as either *concealed* or *private data aggregation protocols*, focus not on integrity, but on the secrecy of individual sensor readings, and of the aggregate total itself. The protocols presented in this chapter are resilient aggregation protocols, and are focused on detecting malicious behavior not only after the fact, but as it occurs during aggregation.

Resilient Aggregation

Resilient aggregation protocols have been studied in a number of different settings. Wagner [103] formally investigates which aggregation functions are possible to securely compute. The model considers several nodes that directly send their readings to a single aggregating node, who then outputs the aggregate total, and demonstrates that even simple aggregation functions, like the sum or average, are insecure in the presence of a single adversary. Other functions, such as counting functions or the median, are more robust. These results are not unexpected, as many statistical functions, such as the average, are greatly affected by extreme values. These results also demonstrate the need for a resilient aggregation protocol to enforce constraints on the readings submitted by malicious nodes. The protocols presented later in this chapter do just that, by relying on nearby nodes to verify that each reading lies within a pre-defined range.

Manulis and Schwenk [71] provide a formal security model and framework for resilient aggregation, alongside a protocol that is secure in their model. The problem setting considered in this chapter matches their model quite closely; however, for ease of understanding, a simpler presentation is used to our protocols.

Hu and Evans [49] present one of the first protocols for resilient aggregation. The protocol aggregates over a tree where each leaf node submits a sensor reading, and each non-leaf node is responsible for aggregating its subtrees. In order to protect against a single malicious node, nodes also forward authentication information to their grandparents in an attempt to detect any malicious behavior by the node between them. As the protocols considered in this chapter are similar in spirit to this approach, a more detailed description is given in Section 5.1.4.

Przydatek et al. [89] provide a different approach to tree-based resilient aggregation, which focuses on computing a good approximation of the aggregate total when there is a single aggregator and some fraction of the nodes are compromised. This approach has inspired improvements which allow multiple aggregating nodes, such as proposals by Chan et al. [25] and Frikken and Dougherty [40]. These approaches detect malicious behavior during a verification phase, which occurs after aggregation has completed. The protocols considered in this chapter attempt to detect malicious behavior as it occurs, so that energy can be saved by terminating the protocol early. A more detailed description of these approaches is given in Section 5.1.4.

Other approaches based on clustering have been proposed, such as SDAP [111], with the goal of minimizing the impact of malicious aggregators located near the root of a spanning tree, who are responsible for reporting aggregate totals for a large portion of the network. The capability of malicious nodes is minimized by partitioning the aggregation tree into subgroups, and including additional authentication information for each subgroup. Synopsis-diffusion approaches have also been proposed, such as those by Roy et al. [95,96], which utilize duplicate insensitive methods. Additional approaches include random set sampling [113], detecting statistical anomalies [16], and homomorphic MACs [64]. While these protocols do provide resilient aggregation for a variety of settings, they are either not compatible, or do not perform well for the topology and key pre-distribution requirements of the family of linear networks considered here.

Concealed and Private Aggregation

In some applications, individual sensor readings, or the aggregate total of them, may be considered sensitive information. In this setting, the focus of protocols may be on providing confidentiality of sensor readings rather than integrity, if not both. Such aggregation protocols are referred to as *concealed data aggregation protocols (CDA)* [41] or *private data aggregation protocols (PDA)* [43] depending on the aggregation topology and the privacy goals of the protocol [22].

Chan and Castelluccia [22] provide a formal security framework for concealed and private aggregation. Peter et al. [87] provide a survey of concealed data aggregation protocols, while Bista and Chang [8] provide a survey of private data aggregation protocols separated into three categories: perturbation-based, shuffling-based, and homomorphism-based. Perturbation-based approaches work by scrambling individual sensor readings before sending them to an aggregator who is able to recover the aggregate, but not individual readings. Examples of such approaches include CPDA [43] and PRDA [82]. Shuffling-based approaches break individual sensor readings into multiple pieces that are sent to multiple destinations. Examples include SMART [43] and improvements on it [110], and iPDA [44], with the latter also focusing on integrity/resiliency as well. Homomorphism-based approaches utilize homomorphic cryptosystems to allow the aggregation of encrypted data. Examples include work by Girao et al. [41], Armknecht et al. [5], and Castelluccia et al. [19].

5.1.2 Problem Statement

We now state the specific problem studied in the rest of this chapter. Let $\mathcal{N} = \{n_1$, n_2, …, $n_N\}$ be a sensor network of N nodes. Assume that nodes have some mechanism to determine when aggregation is to take place. In practice, this may occur at fixed time intervals, or as the result of some network-wide event, such as a request from the base station. Each node n_i possesses a reading r_i from a pre-specified set of possible readings $\mathcal{R} = \{0, 1, \ldots, R-1\}$. A reading $r_i \in \mathcal{R}$ will be referred to as a *valid* sensor reading, while a reading not in the set $\mathcal{R}$ is *invalid*. Information flows through the network toward the base station according to some aggregation protocol, with each honest node behaving according to the protocol. A subset $\mathcal{M} \subset \mathcal{N}$ of nodes are malicious, and under the control of an adversary. For ease of presentation, assume the aggregate function used by all nodes is addition.

Definition 11. *Let $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$ be a sensor network, let $\mathcal{M} \subseteq \mathcal{N}$ be the set of* *malicious nodes in $\mathcal{N}$, and let*

$$T = \sum_{i=1}^{N} r_i$$

be the actual correct aggregate total of all readings in $\mathcal{N}$. An aggregation protocol is secure *(or* resilient*) if it either outputs an aggregate total T' such that*

$$|T - T'| \leq |\mathcal{M}|(R - 1),$$

or identifies the presence of one or more malicious nodes.

Definition 11 states that an aggregation protocol is secure if a malicious entity can modify the aggregate total by at most one valid sensor reading for each node under its control. We can accomplish this by restricting any single malicious node to modifying the aggregate total by at most a single valid sensor reading. That is, a malicious node can lie about its own reading, but cannot otherwise alter another honest node's reading within the aggregate total. Should the adversary modify the aggregate total by more than a single valid sensor reading for each node it controls, then the base station should be alerted to this fact.

The model here is identical to that of Chan et al. [25], who introduced the notion of a *direct data injection attack*, as an attack where a malicious entity controlling one or more nodes may submit a false reading for each node that it controls, with the constraint that false readings must be valid. This attack model leads to their definition of an *optimally secure* aggregation protocol as a protocol where the base station will not accept any maliciously modified aggregate total, except for what can be achieved using a direct data injection attack.

We assume the presence of an adversary that can eavesdrop on all communication and that can selectively compromise a subset of nodes, but cannot compromise the base station. The adversary learns all key information from compromised nodes and may reprogram or alter the behavior of compromised nodes, but it cannot otherwise alter the physical capabilities of any node. The goal of an adversary controlling k nodes is to modify the aggregate total by more than k valid sensor readings without detection. Attacks such as jamming communication, refusing to participate in the protocol, or other denial-of-service attacks are outside the scope of our threat model. However, such attacks can be detected and addressed through other means, such as placing an upper bound on the running time of the protocol.

In the event that malicious behavior occurs, it is not specified how the base station should respond. The protocols in this chapter detect the approximate location of malicious activity down to a set of k nodes, but nodes simply abort the protocol and inform the base station if this occurs. The base station could instruct nearby nodes to simply ignore the potentially compromised subset in the future, or could engage in a more expensive interactive protocol to pinpoint the precise point of failure. The appropriate response to malicious behavior depends on the specific application.

5.1.3 Naive Solutions

As a baseline for comparison, we first consider some bounds on the performance of a resilient aggregation protocol. Let $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$ be a sensor network and assume each node in the network is aware of its children and parents in a spanning tree rooted at the base station. Such knowledge arises naturally in a network, as nodes must, at a minimum, have knowledge of at least one path to the base station in order to deliver any messages. Typically, the spanning tree is minimal with respect to some established metric, such as energy use / distance between nodes.

- **No Aggregation** - Each node sends its reading as a message routed directly to the base station with no aggregation. A node at distance d requires each node on the path between itself and the base station to send a single message, at a cost of d messages in the network. The node's upstream neighbor does the same, adding an additional $d - 1$ messages, and so on. Combined, nodes along any given path of length d collectively generate $d(d + 1)/2$ messages.
- **The Optimal Protocol** - If no adversary is present, then each leaf sends its reading to its parent, and each internal node receives a single message from each child, aggregates them, and sends a single message to its parent. The total communication cost of this protocol is a single message per node, for a total of N messages.
- **Optimal with Integrity Check** - The optimal protocol can be modified to provide integrity for the aggregate total. Assume a single message authentication code accompanies each message. Then each node sends a pair of messages, and the total cost is $2N$.

The above protocols demonstrate the best ($O(N)$) and worst ($O(N^2)$) case for an aggregation protocol. Alongside the total communication cost, the per-node cost must also be considered. In the optimal case, each node is required to send a constant number of messages, while in the worst case, nodes closer to the base station handle a disproportionate number of messages. A good aggregation protocol should balance energy use across nodes so that the lifetime of the network can be maximized.

5.1.4 Comparison to Existing Approaches

Although the problem of secure/resilient aggregation is well studied, most approaches are hierarchical in nature, with each node forwarding readings to an aggregator node higher up in an aggregation tree. Linear networks are a special case of trees, so existing tree-based approaches could be naively applied to linear networks; however, linear networks often produce worst-case performance for tree-based algorithms. This fact is true of many of the tree-based approaches mentioned thus far, and is explored in more detail in Sections 5.1.4 and 5.1.4. Analysis often assumes a balanced aggregation tree is already defined, without specifying how to construct a balanced tree. Similarly, in a linear network, each non-endpoint node is an aggregator, whereas in a complete binary tree only half of the nodes are aggregators. Linear networks also do not match the desired topology for synopsis diffusion-based approaches [96], which utilize multiple node-disjoint paths, and are inefficient if random set sampling is used, as any query must traverse the entire network on its way to a destination. For these reasons, protocols built specifically for linear networks are useful. Existing clustering-based approaches to aggregation could be considered, but the cluster-based approach presented in this chapter exploits the linear arrangement of groups to perform both aggregation and verification with only two messages per node, which matches the theoretical best case performance. Overall, this chapter presents three protocols for resilient data aggregation built specifically for simple linear sensor networks, inspired by three natural key pre-distribution protocols for linear networks (described in Section 5.2.1).

The protocols presented here are designed from the ground up to:

- Exploit the known topology of linear networks;
- utilize natural key pre-distribution schemes for linear networks;
- limit the capability of a malicious node;
- detect malicious behavior as it occurs, not afterwards; and
- provide proofs of security.

To motivate our specialized aggregation protocols for linear networks, we first describe two existing approaches that demonstrate the problem with applying a tree-based algorithm to a linear network.

Hu and Evans Protocol

One of the earliest protocols to provide resilient aggregation is given by Hu and Evans [49], who provide a tree-based protocol that protects against singular (non-adjacent) malicious nodes by having a node and its grandparent exchange information to ensure the node in the middle performed aggregation correctly. This approach is the same basic idea that underlies the protocols presented in this chapter. For simplicity, Hu and Evans assume that all non-leaf nodes in the network are simply aggregators and do not contribute a reading themselves. This assumption can be eliminated in a straightforward manner by having each aggregator create a virtual leaf node attached to itself.

Hu and Evan's protocol consists of two distinct phases. In the first phase, each leaf forwards its sensor reading to its parent, along with a MAC computed using a key known only to the leaf and the base station. Each aggregating node receives a set of readings and/or aggregate totals along with MACs, computes the updated aggregate from the received reading, and computes a MAC on the updated aggregate using a key shared only with the base station. The received readings, the MACs generated by its children, and the updated MAC are all forwarded to the node's parent. Aggregation continues in this manner until it reaches the root.

When aggregation is complete, the verification phase begins. Each aggregating node possesses a set of readings, a set of MACs on those readings from its grandchildren, and the resulting aggregate total. Using an authenticated broadcast [86], the base station then reveals the individual keys used by each node to compute the MACs, thus enabling every node's grandparent to verify that the intermediate node between them properly forwarded the correct data. If any node fails to verify the received MACs, then it can raise an alarm and alert the base station. Subsequent runs of the protocol require each node to move on to the next key in a deterministic key chain.

The protocols presented in this chapter are similar in spirit to the approach by Hu and Evans, hereafter referred to as the HE protocol; however, the assumptions and execution differ in several key ways:

1. The HE protocol assumes the base station can broadcast directly to individual nodes. The protocols in this chapter do not require direct interaction with the base station.

2. The HE protocol assumes the network is dense and several nodes exist within one hop of any node. This assumption does not hold in linear networks.

3. The HE protocol assumes that if an aggregating node is malicious, then its child

and grandparent are both honest. The protocols presented in this chapter provide resilience in the presence of multiple connected malicious nodes.

4. The protocols in this chapter detect malicious activity as it occurs, not during a later verification phase. This property allows for energy savings by terminating the protocol early.

5. The basic HE protocol is not scalable, as the base station communicates with each node after each run of the protocol. A scalable variant is proposed in which the base station assists each relevant pair of nodes in establishing shared secret information to be used locally within the network. The protocols in this chapter do not require any extra setup prior to the first run of the protocol.

The protocols considered in this chapter follow the approach of Hu and Evans by utilizing nearby nodes to ensure no set of nodes misbehaves, but also leverage the key features of a strict linear topology to achieve resilience using only a single flow of data from one end of the network to the other. This reduction in communication cost is possible due to the additional assumptions that can be made for a fixed linear topology.

The CPS Aggregation Protocol

The aggregation protocol by Chan et al. [25], hereafter referred to as the CPS protocol, is a widely cited hierarchical aggregation protocol in the sensor network literature. The adversarial model is identical to the model in this chapter, but the topology and key requirements for individual sensor nodes are different. Each node is expected to share a key with the base station, and the base station must be able to perform an authenticated broadcast to all sensor nodes in the network. The first assumption is quite common in sensor network literature, and the second can be realized through existing protocols, such as μTESLA [86]. The CPS protocol proceeds in four phases:

1. **Query Dissemination** - The base station initiates the aggregation request, which creates an aggregation tree spanning the network as the request propagates.

2. **Aggregation Commitment** - The sensor nodes collaboratively construct a commitment tree over the aggregation result. Each leaf constructs a node containing its identifier and its sensor reading, which is forwarded to its parent. Each internal aggregating node constructs a node in the commitment tree by computing the updated aggregate result along with a hash over all of its children, including a virtual leaf node containing its own sensor reading.

3. **Distributed Verification** - Each node independently verifies that its reading was included in the result. To accomplish this, each node in the tree must receive enough information to independently compute the same result its parent did. In other words, each node in the network must receive the result computed by every sibling of a node on the path from itself to the root. This information is disseminated by having each node, starting with the base station and progressing downward, broadcast the set of aggregation nodes it received. The base station also includes a nonce for use in the next phase. The root of the tree and the nonce are sent using an authenticated broadcast.

4. **Confirmation** - Each node independently recomputes the aggregation result and verifies it is correct. If so, each node n_i forwards $\text{MAC}_{k_i}(\text{nonce}||\text{OK})$, or $\text{MAC}_{k_i}(\text{nonce}\ ||\text{NOT OK})$ otherwise, where k_i is the unique key shared between node n_i and the base station. Each internal node computes the XOR of its received MACs and forwards the result to its parent. Because the base station possesses all keys, it is able to compute the expected result independently. If the received and computed MACs are identical, then the aggregation result is accepted as correct.

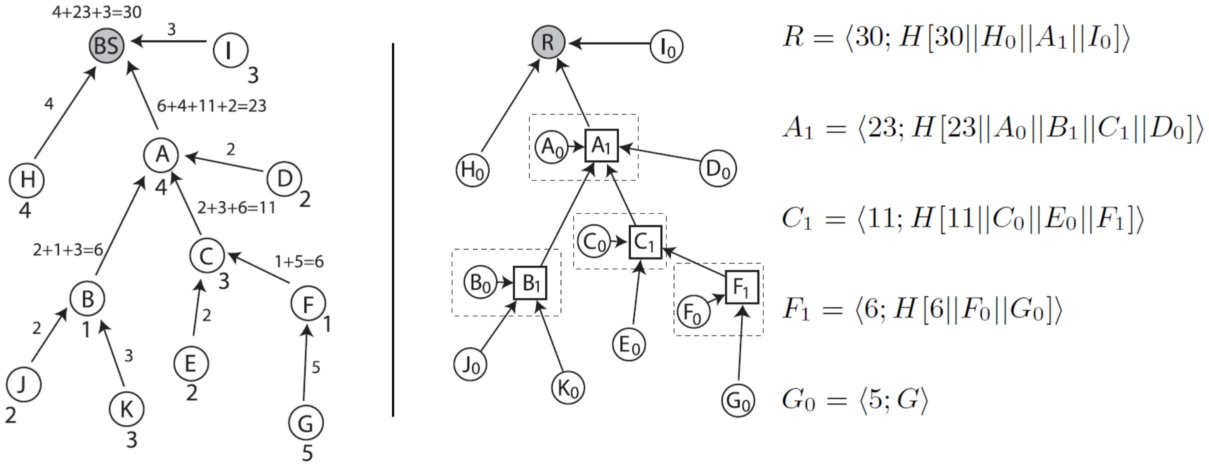

4+23+3=30
BS
3
I
3
4
6+4+11+2=23
H
4
A
4
2
D
2
2+3+6=11
2+1+3=6
C
3
1+5=6
B
1
F
1
2
3
2
E
2
5
J
2
K
3
G
5
R
I_0
H_0
A_0
A_1
D_0
C_0
C_1
B_0
B_1
F_0
F_1
E_0
J_0
K_0
G_0
$R = \langle 30; H[30||H_0||A_1||I_0]\rangle$
$A_1 = \langle 23; H[23||A_0||B_1||C_1||D_0]\rangle$
$C_1 = \langle 11; H[11||C_0||E_0||F_1]\rangle$
$F_1 = \langle 6; H[6||F_0||G_0]\rangle$
$G_0 = \langle 5; G\rangle$

Figure 5.2: Commitment tree generation in the CPS aggregation protocol. Each aggregator node creates a virtual leaf containing its sensor reading, and creates a node in the commitment tree by computing a hash over its children. Figure taken from Chan et al. [25, Figure 1].

Figure 5.2 demonstrates the aggregation commitment phase of the CPS protocol. During each phase of the protocol, information flows either upwards or downwards throughout the entire network. In the verification phase, each node must receive the aggregation result computed by the sibling of each node on the path from itself to the root. For example, in Figure 5.2, the base station informs node A_1 of the result H_0 and I_0, while node A_1 informs node C_1 of the same, along with the results A_0, B_1, and D_0. This pattern continues, with the node G_0 requiring the set $\{F_0, C_0, E_0, A_0, D_0, H_0, I_0\}$. The resulting communication cost to disseminate these results is dependent on the degree of each node in the tree, along with the length of the path from a leaf to the root.

Frikken and Dougherty [40] provide a method for generating a tree for use in the CPS protocol such that the resulting communication cost of the protocol is $O(\lg N)$ for each node. The authors of the CPS protocol have revisited it [23], demonstrating that the protocol itself achieves a more general functionality than just aggregation. More specifically, the protocol can be altered in such a way that authenticated broadcast is no longer necessary, as the commit-verify process can provide this functionality on its own. The ability to compute and verify hash trees over an aggregation topology can be utilized for other purposes, such as node-to-node signatures or a simple public key infrastructure, with the only requirement being that each node shares a unique pairwise key with the base station.

Although the CPS protocol performs well and provides useful functionality in a general setting, the hierarchical nature of the protocol does not lend itself well to linear networks. To verify its reading is included in the total, each node recomputes the result for each node on the path from itself to the root. Using Frikken and Dougherty's method, the aggregation tree has expected depth $\lg N$, which is the main factor in communication cost for each node. In a linear network, the depth of the tree is N, leading to a per-node cost of $O(N)$, and $O(N^2)$ total communication cost. In this worst-case setting, the naive approach of simply forwarding each reading directly to the base station is a cheaper option. This result is not unexpected, as tree-based algorithms generally incur a cost depending on the depth of the tree, so linear networks are expected to achieve worst-case performance when a generic tree-algorithm is applied.

The communication cost of the CPS protocol is not uniform across the network. Nodes closer to the root require fewer messages to verify that its contribution was included in the root node. While asymmetry in the communication cost may be seen as a drawback, it can also be argued that such a pattern of communication works well in a hierarchical network, where nodes closer to the root are generally responsible for routing messages from a larger proportion of the network. By comparison, the protocols for linear networks considered in this chapter achieve a constant cost per node. If aggregating sensor readings is the primary activity of the network, then balancing communication cost across all nodes is the approach that best prolongs the lifetime of the network by ensuring each node's energy costs are the same.

5.2 Linear Networks

Linear networks are a natural deployment model for sensor networks. Applications such as pipeline, subway, border, or perimeter monitoring are inherently linear. Jawhar and Mohamed [53] give a classification of linear sensor networks that focuses on the topology of the network (thin, thick, or very thick), as well as a hierarchical classification of nodes according to capability and role within the network.

The simplest linear network is a *thin / one-level* network of uniformly distributed identical sensor nodes deployed along a straight line. Such a network may also be referred to as a *simple* linear network or a *one-dimensional* network.

Definition 12 (Simple Linear Network)**.** *A* simple linear network *is a connected non-cyclic graph where each node has exactly one neighbor, or two distinct neighbors. The two nodes with only one neighbor are referred to as the* endpoints *of the network.*

Although the physical arrangement of nodes is linear, the fact that nodes communicate wirelessly means that a node may be able to communicate with more than just its direct neighbors in the physical network. Thus, the *communication graph* may have additional edges not present in the physical network graph.

Definition 13 ((N, d)-Linear Network)**.** *A* (N, d)-linear network *is a linear network containing N nodes, each able to communicate with nodes up to d hops away. The nodes of such a network are denoted by $n_1, n_2, \ldots, n_N$ where nodes n_1 and n_N are the endpoints, and node n_i is located between nodes n_{i-1} and n_{i+1} for $1 < i < N$.*

Figure 5.3 shows an example of an $(8, 2)$-linear network.

We refer to nodes that are directly adjacent to each other as *neighbors*, and neighbors at distance d if there are $d - 1$ nodes in between them.

In terms of Martin and Paterson's framework, a simple linear network is:

- Homogeneous - All nodes that are not endpoints are identical.
- Fixed - Nodes are not mobile.

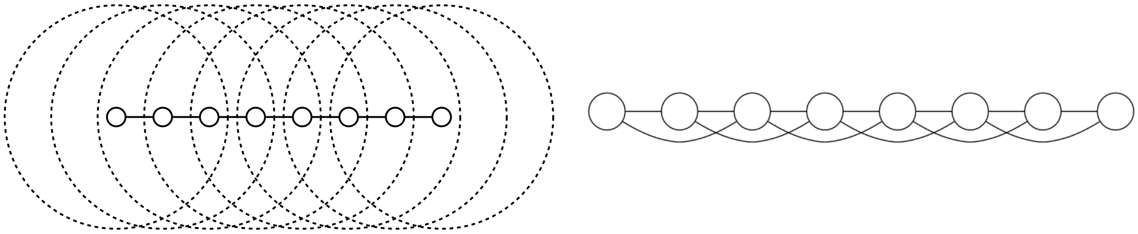


Figure 5.3: An $(8, 2)$-linear sensor network. The left diagram shows the physical layout of the network along with dotted circles denoting the communication range of each node. The right diagram shows the communication graph for the same network

- Full control - Given the sequential deployment nature of linear networks, it is reasonable to assume that the order nodes are deployed in is fully controlled.

These properties, in particular, the high degree of control over the deployment of nodes, allow for the use of very efficient key pre-distribution schemes to facilitate secure communication.

An important factor that distinguishes linear sensor networks from more general networks is the expected difference in node density. In general, sensor networks are modeled as nodes deployed on a flat plane, with each node having a given communication radius. Because linear networks are one-dimensional, it is expected that far fewer nodes will be located within any given node's communication range. In the extreme case, each node in a linear network may have at most two neighbors within its communication range.

Jawhar and Mohamed do not formally define a thick linear network, but rather give an intuitive definition where nodes are distributed randomly between two parallel lines, with the width of the network being far greater than the height of the network. The overall shape of the network is linear, but the local topology need not be. In a two-level linear network, the higher-tier nodes would form a linear network. Examples of thick linear topologies are given in Figure 5.4. These networks differ from the simple case in that node density is much higher, and therefore may be more amenable to general aggregation techniques. Note that because of the linear nature of the network, there still exist a large number of nodes along the edges of the network with lower connectivity than might be expected in a more regularly shaped network. The two-level network could be abstracted as several side-by-side sensor networks, as each lower level node has a nearest second level node acting as a base station to which it sends its readings. The second level nodes are arranged in a simple linear network, and can therefore use the protocols in this chapter to

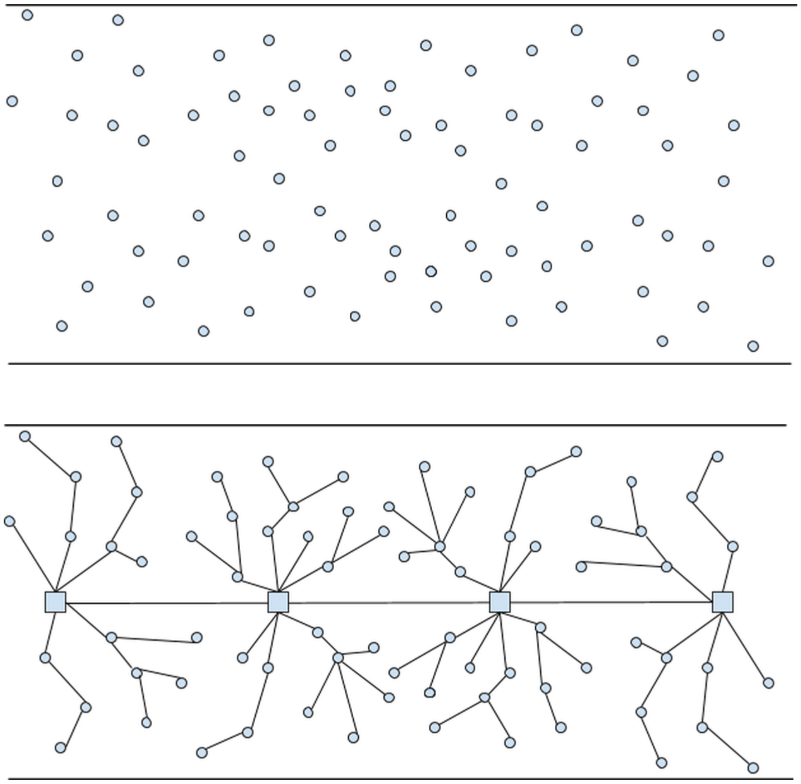


Figure 5.4: Examples of subsets of a thick linear network, and thick two-level linear network. A thick linear network would continue with a similar deployment pattern in either direction.

perform their aggregation.

Linear Protocols as Building Blocks

Although the linear problem setting differs from that of most general networks, linear subnetworks are often utilized in general networks, such as simply routing a single message between two nodes. Therefore, depending on the specific setting, protocols for linear networks can serve as building blocks for protocols in more general settings. While a natural approach is to utilize a tree structure for aggregation, typically a minimal spanning tree in the underlying routing protocol, linear networks still arise naturally in the spanning trees of a random network. Figure 5.5 demonstrates this for a network of 100 random nodes. Nodes are distributed randomly across a square plane, and are connected if they are within a fixed distance of each other (i.e., the network is a geometric graph). The minimal Euclidean spanning tree is highlighted, and demonstrates that even in a well-connected network, many nodes in the spanning tree have only a single child, and those that don't typically only have two children.

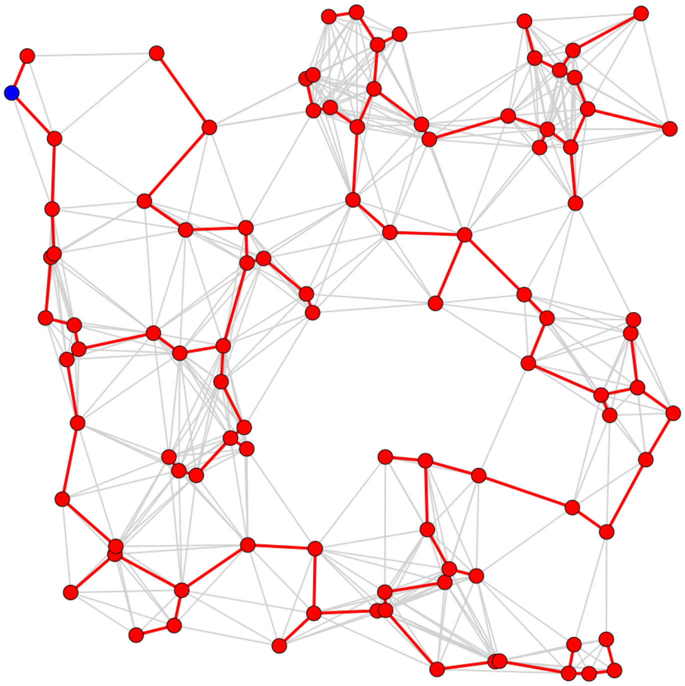


Figure 5.5: A minimal Euclidean spanning tree on a geometric graph of 100 nodes. The MST is rooted at the darker node in the top left corner, and two nodes are connected with a grey line if they are within each other's communication range.

The protocols in this chapter are designed with simple linear networks in mind, but Section 5.7 describes how the same approach can be used to merge two or more linear networks at a single point, provided that certain assumptions about key pre-distribution hold. Therefore, although initially designed for simple linear networks, the results in this chapter can be utilized for aggregation in random topologies without much modification, without sacrificing any resilience.

5.2.1 Key Pre-distribution for Linear Networks

In order to facilitate secure communication, nodes are pre-loaded with cryptographic keys using a *key pre-distribution scheme (KPS)*. Earlier chapters of this thesis investigated a family of combinatorial key pre-distribution schemes that are suited for a wide variety of homogeneous, fixed, randomly distributed sensor networks. Although these schemes perform extremely well for random topologies, the simplicity and deployment pattern of a (N, d)-linear sensor network (e.g., fixed, full-control) lends itself well to three specific key pre-distribution schemes:

1. **Optimal** (N, r)-KPS - Each node shares a pairwise key with nodes at distance 1 and distance r.

2. **Pairwise** (N, r)-KPS - Each node shares a distinct key with each neighbor at distance j, for $1 \leq j \leq r$.

3. **Group-based** (N, r)-KPS - Each consecutive subset of $r + 1$ nodes shares a distinct group key.

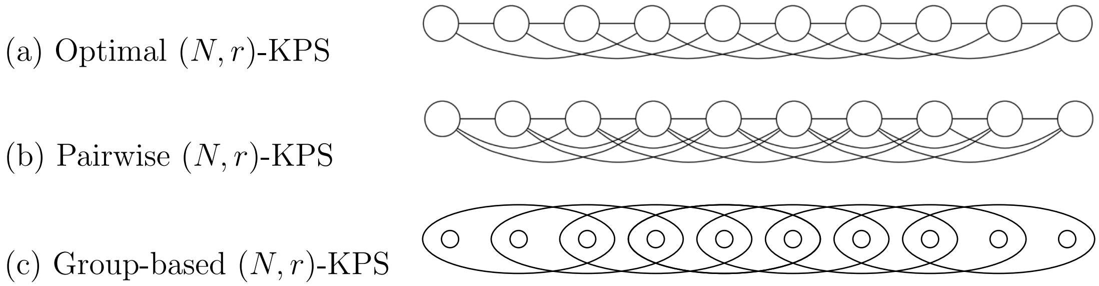
The Optimal (N, r)-KPS scheme was introduced by Martin and Paterson [75], and is optimal with respect to the number of keys necessary to maintain secure connectivity in a network when $d = r$ and up to $r - 1$ nodes are unavailable. If r consecutive nodes are unavailable, then the network becomes disconnected regardless of the underlying KPS. Additionally, each key in the optimal scheme is possessed by exactly two nodes, thus minimizing the number of nodes affected by the compromise of any given key.

The above schemes can be combined, for example, by issuing both pairwise keys and group keys to nodes. Additionally, these basic schemes may be augmented with additional specialized keys, such as each node sharing a unique pairwise key with a base station. In most settings, it is natural to choose $r = d$, although depending on the network setting and requirements, the value of r used for key pre-distribution need not match the maximum communication range d of the network.

The goal of the next sections is to establish resilient data aggregation protocols for simple linear networks under pairwise, group-based, and optimal key pre-distribution.

5.3 Secure Aggregation using the Optimal KPS

The optimal (N, d)-KPS for (N, d)-linear networks specifies that each node shares unique keys with each neighbor at distance 1 and each neighbor at distance d, for a total of four



(a) Optimal (N, r)-KPS

(b) Pairwise (N, r)-KPS

(c) Group-based (N, r)-KPS

Figure 5.6: Optimal and pairwise KPS for a $(10, 3)$-linear network, and group-based KPS for a $(10, 2)$-linear network. In (a) and (b), shared keys are denoted by a link between two nodes, whereas in (c) each circled group of nodes possess a unique key

keys per non-endpoint node when $d > 1$, except for nodes less than distance $d - 1$ from an endpoint, which possess only three keys. This scheme is optimal with respect to the number of keys possessed by each node to maintain secure connectivity even if up to $d - 1$ consecutive nodes fail. In this section, we present a technique for secure aggregation using a KPS inspired by the optimal KPS.

As a starting point, consider a $(N, 1)$-linear network $\mathcal{N} = \{n_1, \ldots, n_N\}$, where each node is capable of communication only with nodes at distance 1. This case is the most restrictive case possible, as each node can only speak to its direct neighbors in the network. We assume that some nodes in $\mathcal{N}$ are malicious, but that no more than k consecutive nodes are malicious. Note that we still allow multiple disjoint sets of up to k consecutive malicious nodes, as long as there is at least one honest node in between them. In the same way that the optimal (N, r)-KPS provides resilience against $r - 1$ node failures by ensuring nodes distance r apart have a shared key, we can provide resilience during aggregation against up to k consecutive malicious nodes by ensuring that nodes at distance $k + 1$ have a shared key, even if they are not capable of direct communication. More specifically, although $\mathcal{N}$ is a $(N, 1)$-linear network, we distribute keys using the optimal $(N, k + 1)$-KPS. That is, each node shares a distinct key with nodes at distance 1 and nodes at distance $k + 1$. The key shared between two nodes n_i and n_j will be referred to by $k_{i,j}$ where $|i - j| = 1$ or $k + 1$.

5.3.1 Aggregation when $k = 1$

To begin, consider the case when $k = 1$, i.e., where no two consecutive nodes in $\mathcal{N}$ are both malicious. In order to prevent a single malicious node, say $m_i = n_i$ from contributing an invalid sensor reading, we have node n_{i-1} tag the aggregate total X_{i-1} from the first $i-1$ nodes using a *message authentication code (MAC)* under key $k_{i-1,i+1}$. Node n_{i+1} proceeds if and only if it receives from m_i the value X_{i-1} along with a MAC to verify it is unaltered, as well as the updated aggregate X_i (or, equivalently, the sensor reading r_i). Thus, n_{i+1} can verify that both X_{i-1} and r_i are correct, and, if so, node n_{i+1} can correctly compute $X_{i+1} = X_{i-1} + r_i + r_{i+1}$. Figure 5.7 demonstrates this process in three cases. The first case demonstrates the beginning of the protocol, the second case demonstrates non-endpoint nodes, and the third cases demonstrates the termination of the protocol.

The protocol in Figure 5.7 is resilient as long as no two malicious nodes are neighbors within the network, because single malicious nodes are forced to pass their reading, along with an authenticated aggregate sum from its previous neighbor, on to an honest node. The honest node verifies the validity of the received reading, adds it into the aggregate sum, and then forwards the sum, its own reading, and an authentication tag on to the next node so the process can be repeated. The protocol terminates either when a node outputs `reject`, or when the base station n_N receives and validates the values X_{N-2} and r_{N-1}. The resilience of the protocol is proven in Theorem 19.

Theorem 19. *Suppose that the protocol in Figure 5.7 is being used for data aggregation, that node n_i is honest, that no two neighbors are both dishonest, and that nodes $n_1, \ldots, n_{i-1}$ do not output* `reject`*. Then, n_i either outputs* `reject`*, or node n_i correctly computes $X_i = \sum_{j=1}^{i} r_j$ where $r_1, \ldots, r_i \in \{0, \ldots, R-1\}$.*

Proof. We prove this by strong induction on i, first considering $i = 1$ as a base case. If node n_1 is honest, then it knows the correct value of $r_1 = X_1$, and it can compute $t_{1,3} = \mathrm{MAC}_{k_{1,3}}(X_1)$. If node n_1 is dishonest, then node n_2 must be honest. Node n_2 has knowledge of r_2 and receives $X_1 = r_1$ directly from n_1, with which it can verify that $r_1 \in \{0, \ldots, R-1\}$. Therefore, node n_2 can correctly compute $X_2 = r_1 + r_2$ and $t_{2,4} = \mathrm{MAC}_{k_{2,4}}(X_2)$.

For the purpose of strong induction, assume that all honest nodes n_j for $1 \leq j \leq i-1$ can correctly compute

$$X_{j-1} = \sum_{l=1}^{j-1} r_l,$$

where $r_l \in \{0, \ldots, R-1\}$, as well as correctly compute $t_{j-1,j+1}$.

Suppose that node n_{i-1} is honest, and that node n_{i-2} is possibly dishonest. Then, by induction,

$$X_{i-1} = \sum_{l=1}^{i-1} r_l$$

$\mathbf{n_1}$		$\mathbf{n_2}$		$\mathbf{n_3}$
$X_1 \leftarrow r_1$ $t_{1,3} \leftarrow \mathrm{MAC}_{k_{1,3}}(X_1)$				
	$X_1, t_{1,3}$ $\longrightarrow$			
		verify $X_1 = r_1 \in \{0, \ldots, R-1\}$, else `reject` $X_2 \leftarrow r_1 + r_2$ $t_{2,4} \leftarrow \mathrm{MAC}_{k_{2,4}}(X_2)$		
			$X_1,\ X_2$ $t_{1,3},\ t_{2,4}$ $\longrightarrow$	

$\mathbf{n_{i-1}}$		$\mathbf{n_i}$		$\mathbf{n_{i+1}}$
	$X_{i-2},\ X_{i-1}$ $t_{i-2,i},\ t_{i-1,i+1}$ $\longrightarrow$			
		verify $t_{i-2,i}$, else `reject` $r_{i-1} \leftarrow X_{i-1} - X_{i-2}$ verify $r_{i-1} \in \{0, \ldots, R-1\}$, else `reject` $X_i \leftarrow X_{i-1} + r_i$ $t_{i,i+2} \leftarrow \mathrm{MAC}_{k_{i,i+2}}(X_i)$		
			$X_{i-1},\ X_i$ $t_{i-1,i+1},\ t_{i,i+2}$ $\longrightarrow$	

$\mathbf{n_{N-2}}$		$\mathbf{n_{N-1}}$		$\mathbf{n_N}$
	$X_{N-2},\ X_{N-1}$ $t_{N-2,N}$ $\longrightarrow$			
		verify $t_{N-3,N-1}$, else `reject` $r_{N-2} \leftarrow X_{N-2} - X_{N-3}$ verify $r_{N-2} \in \{0, \ldots, R-1\}$, else `reject` $X_{N-1} \leftarrow X_{N-2} + r_{N-1}$		
			$X_{N-2},\ X_{N-1}$ $t_{N-2,N}$ $\longrightarrow$	

Figure 5.7: A resilient data aggregation protocol for $(N, 1)$-linear networks where $k = 1$ using the optimal KPS

and $r_{i-1} \in \{0, \ldots, R-1\}$. If $t_{i-2,i} \neq \mathrm{MAC}_{k_{i-2,i}}(X_{i-2})$, then node n_i outputs `reject`. Otherwise, $X_{i-2} = X_{i-1} - r_{i-1}$ and node n_i can correctly compute $X_i = X_{i-1} + r_i$ and $t_{i,i+2} = \mathrm{MAC}_{k_{i,i+2}}(X_i)$.

Next, suppose that node n_{i-1} is dishonest, then node n_{i-2} must be honest. By induction,

$$X_{i-2} = \sum_{l=1}^{i-2} r_l$$

is the correct aggregate total and $t_{i-2,i}$ is a valid MAC for X_{i-2}. If either of these values are modified by n_{i-1}, then node n_i will output `reject`, unless node n_{i-1} can forge a MAC using key $k_{i-2,i}$. Node n_i receives the value X_{i-1} from node n_{i-1}, computes $r_{i-1} = X_{i-1} - X_{i-2}$, and can verify directly that $r_{i-1} \in \{0, \ldots, R-1\}$. Therefore, node n_i can correctly compute $X_i = X_{i-2} + r_{i-1} + r_i$ and $t_{i,i+2} = \mathrm{MAC}_{k_{i,i+2}}(X_i)$. □

Corollary 2. *The protocol in Figure 5.7 satisfies Definition 11.*

Although the protocol presented in Figure 5.7 allows the detection of the presence of active malicious nodes, it does not allow us to precisely identify which node is malicious. If a node n_i fails to verify the tag $t_{k_{i-2,i}}$ there are three possibilities:

1. Node n_i is malicious and falsely claims that $t_{k_{i-2,i}}$ is invalid;
2. Node n_{i-1} is malicious and altered the value $t_{k_{i-2,i}}$; or
3. Node n_{i-2} is malicious and forwarded an invalid $t_{k_{i-2,i}}$.

As these three nodes are the only nodes to handle the value $t_{k_{i-2,i}}$, they are the only nodes that could have possibly altered it. The limited key information and communication range of each node makes detecting which of these cases occurred difficult, if not impossible, without additional assumptions or the assistance of the base station. As our goal is simply to prevent an incorrect aggregate total from being reported, precise adversarial detection and how to respond to it is left as future work.

The protocol as presented in Figure 5.7 assumes that nodes can compute r_{i-1} from X_{i-1} and X_{i-2}. It is straightforward to modify the protocol to avoid this requirement simply by having node n_{i-1} forward r_{i-1} instead of X_{i-1}. Then node n_i computes X_{i-1} using X_{i-2} and r_{i-1}.

5.3.2 Analysis

As a baseline for comparison, we note that any hop-by-hop aggregation protocol requires each node to send at least one message to propagate the aggregate total. Adding an integrity check, such as a MAC, would add an additional message per node. Therefore, we can assume a lower bound of two messages per node and $2N$ messages total in the absence of any malicious nodes.

The communication cost of the $k = 1$ protocol is straightforward to compute. Each non-endpoint node n_i forwards four messages: the current aggregate total X_i and a MAC to be verified by node n_{i+2}, and the previous hop's aggregate total X_{i-1} along with the received MAC to be verified by node n_{i+1}. Therefore, the total communication cost is slightly less than $4N$ messages. Note that in a $(N, 1)$-linear network, communication between nodes at distance 1 is implicitly authenticated when traffic flows in one direction, as there is only one possible source for a received message. A malicious node attempting to impersonate a different honest node would result in duplicate messages, which is readily detected. The protocol could be altered to explicitly authenticate messages between nodes at distance 1, either by adding an additional MAC (yielding a total of five messages per node), or by utilizing an authenticated mode of encryption.

For ease of presentation, no additional information was included in any MAC to avoid *replay attacks*. If the protocol is run more than once, then an adversary can reuse messages and MACs from a prior run of the protocol without detection. In practice, a *nonce* must be included in each MAC to ensure freshness. Depending on the specific application, a nonce is readily available in the form a counter, time stamp, or session identifier.

This protocol can be viewed as the natural adaptation of the approach of Hu and Evans to a linear network. The fact that nearby nodes are guaranteed to possess shared keys allows for verification to take place during the aggregation phase rather than afterwards, thereby eliminating the need for caching or re-computing results, and allowing for early termination if malicious activity is detected. A large drawback of Hu and Evans' approach is the inability to detect pairs of connected colluding nodes. The next section demonstrates the difficulty in naively extending this approach in an attempt to protect against multiple malicious nodes.

5.3.3 An Attack Against a Naive $k > 1$ Protocol

The protocol in the previous section can be naturally extended to larger values of k simply by distributing keys using the optimal $(N, k + 1)$-KPS. Each node would then receive an authenticated aggregate sum from $k+1$ hops back, along with readings from the previous k hops to verify directly.

Figure 5.8 demonstrates such an approach for $k = 2$. Unfortunately, this method is not adequate to protect against colluding malicious nodes.

```
$\mathbf{n_{i-1}}$              $\mathbf{n_i}$                                   $\mathbf{n_{i+1}}$
         $X_{i-1}$, $t_{i-1,i+2}$
         $X_{i-2}$, $t_{i-2,i+1}$
          $X_{i-3}$, $t_{i-3,i}$
           $\longrightarrow$
                        verify $t_{i-3,i}$, else reject
                        $r_{i-1} \leftarrow X_{i-1} - X_{i-2}$
                        $r_{i-2} \leftarrow X_{i-2} - X_{i-3}$
                        verify $r_{i-1}, r_{i-2} \in \{0, \ldots, R-1\}$
                            else reject
                        $X_i \leftarrow X_{i-1} + r_i$
                        $t_{i,i+3} \leftarrow \mathrm{MAC}_{k_{i,i+3}}(X_i)$
                                                          $X_i$, $t_{i,i+3}$
                                                        $X_{i-1}$, $t_{i-2,i+2}$
                                                        $X_{i-2}$, $t_{i-2,i+1}$
                                                           $\longrightarrow$
```

Figure 5.8: A flawed data aggregation protocol for $(N, 1)$-linear networks where $k = 2$

Although this approach is the natural generalization of the secure $k = 1$ protocol presented in the previous section, it is now possible for a pair of malicious nodes to alter the aggregate total by more than two valid sensor readings. This attack arises due to the fact that, in the $k > 1$ setting, an honest node immediately preceding a malicious node is not guaranteed to have its aggregate total verified by an honest node. For example, if node n_i is honest, it is possible that nodes n_{i+1} and n_{i+3} are both dishonest. If this occurs, it is possible for node n_{i+1} to alter the total without being detected, as malicious node n_{i+3} would normally be the node to detect such an attack. This situation could not arise in the $k = 1$ case, as any malicious node always has an honest node on either side of it. In other words, this approach works to defend against two consecutive malicious nodes, but no longer protects against colluding non-consecutive malicious nodes.

To illustrate an attack against the naive $k = 2$ protocol, consider a network where the honest reading of node n_i is $r_i = 0$. A pair of malicious nodes should be able to modify the total by at most $2(R-1)$. Consider a connected subset of the network $\{n_i, m_{i+1}, n_{i+2}, m_{i+3}, n_{i+4}\}$, where n_k denotes an honest node and m_k denotes a malicious node. The attack proceeds as follows:

1. Node n_i sends the following to node m_{i+1}:
$$\begin{aligned}&X_{i-2} = X_{i-1} = X_i = 0\\&t_{i-2,i+1}, t_{i-1,i+2}, t_{i,i+3}.\end{aligned}$$
2. Node m_{i+1} replaces X_i with $X_i' = R-1$, adds the reading $r_{i+1} = R-1$ to the total such that $X_{i+1} = 2(R-1)$, and computes the MAC $t_{i+1,i+4}$ on this total. The following is forwarded to node n_{i+2}:
$$\begin{aligned}&X_{i-1} = 0, X_i' = R-1, X_{i+1} = 2(R-1)\\&t_{i-1,i+2}, t_{i,i+3}, t_{i+1,i+4}.\end{aligned}$$
3. Node n_{i+2} verifies the unaltered value $X_{i-1} = 0$ and accepts, forwarding the following to node m_{i+3}:
$$\begin{aligned}&X_i' = R-1, X_{i+1} = 2(R-1), X_{i+2} = 2(R-1)\\&t_{i,i+3}, t_{i+1,i+4}, t_{i+2,i+5}\end{aligned}$$
4. Node m_{i+3} is supposed to verify that $t_{i,i+3}$ is a valid MAC (on $X_i = 0$), but ignores the fact that the malicious node m_{i+1} has replaced it with $X_i' = R-1$. Node m_{i+3} sets $r_{i+3} = R-1$, computes $X_{i+3} = 3(R-1)$, and computes a MAC $t_{i+3,i+6}$ on this value. The following is forwarded to node n_{i+4}:
$$\begin{aligned}&X_{i+1} = X_{i+2} = 2(R-1), X_{i+3} = 3(R-1)\\&t_{i+1,i+4}, t_{i+2,i+5}, t_{i+3,i+6}\end{aligned}$$
5. Node n_{i+4} verifies that $t_{i+1,i+4}$ is a valid MAC on X_{i+1} and accepts X_{i+1} as valid. At this point, the malicious nodes have managed to modify the total by $3(R-1)$ without being detected.

This attack can be avoided if any MAC generated by an honest node is always verified by an honest node, however this requires a much stronger assumption on the distribution of malicious nodes when working in the optimal KPS setting, that is unlikely to apply in any practical setting. The next section demonstrates how this problem can be overcome if additional keys are distributed to nodes, such as in the pairwise KPS model.

5.4 Aggregation Using the Pairwise KPS

The previous section presented a protocol for detecting non-consecutive malicious nodes during aggregation and demonstrated the difficulty in extending it to detecting coalitions of malicious nodes. In this section we extend the protocol to protect against a coalition of up to k consecutive malicious nodes when $k > 1$. The goal of this protocol remains the same: to prevent a malicious node from altering the aggregate total by more than a single valid sensor reading. A coalition of consecutive malicious nodes should therefore only be able to modify the aggregate total by no more than k valid sensor readings.

The basic idea behind the single-node protocol is to exploit the existence of honest nodes on either side of any malicious node. The first honest node computes and forwards an authenticated aggregate total that the next honest node can verify. If the malicious node in the middle misbehaves, one of the next two hops will detect it. The k-resilient version of the protocol is based on the same idea using the pairwise $(N, k+1)$-KPS. Any coalition of up to k consecutive nodes must have an honest node on either side of it. Thus, authentication information from $k+1, k, \ldots, 2$ hops back will be forwarded at each hop to ensure that none of the preceding k nodes have altered the aggregate total in an invalid manner. Therefore, the extended protocol requires that any two nodes n_i and n_j within distance $d \leq k+1$ of each other must have a unique pairwise key, $k_{i,j}$, shared with each other. Figure 5.9 demonstrates a 2-resilient aggregation protocol, broken into three cases demonstrating the beginning, general case, and termination of the protocol. A security proof when $k = 2$ follows.

The protocol presented here assumes a $(N, 1)$-linear network with a pairwise $(N, k+1)$-KPS. In an $(N, k+1)$-linear network, the protocol can be made much more efficient by node i forwarding the value $t_{i,i+k+1}$ directly to node n_{i+k+1}. In general, for any communication range d where $1 < d < k+1$, each node will forward messages directly to any node at distance $r \leq d$, or d hops further in the network for nodes at distance $r > d$. As in the $k = 1$ protocol, if the communication range of nodes is only 1 (i.e., a $(N, 1)$-linear network), then a separate MAC may not be necessary to authenticate messages between nodes at distance $d = 1$.

As in the $k = 1$ case, the protocol as presented here assumes that nodes can compute readings from the previous hops using only the X_i values. It is straightforward to modify the protocol to avoid this requirement simply by having nodes from k forward their individual readings instead of aggregate totals. Node i then computes X_i from these readings rather than computing the readings from aggregate totals.

Theorem 20. *Suppose the protocol in Figure 5.9 is being used for data aggregation, that*

$\mathbf{n_1}$		$\mathbf{n_2}$		$\mathbf{n_3}$
$X_1 \leftarrow r_1$ $t_{1,2} \leftarrow \mathrm{MAC}_{k_{1,2}}(X_i)$ $t_{1,3} \leftarrow \mathrm{MAC}_{k_{1,3}}(X_i)$ $t_{1,4} \leftarrow \mathrm{MAC}_{k_{1,4}}(X_i)$				
	$X_1,\ t_{1,2},\ t_{1,3},\ t_{1,4}$ $\longrightarrow$			
		verify $t_{1,2}$, else `reject` verify $r_{i-1} = X_1 \in \{0, \ldots, R-1\}$, else `reject` $X_2 \leftarrow X_1 + r_2$ $t_{2,3} \leftarrow \mathrm{MAC}_{k_{2,3}}(X_2)$ $t_{2,4} \leftarrow \mathrm{MAC}_{k_{2,4}}(X_2)$ $t_{2,5} \leftarrow \mathrm{MAC}_{k_{2,5}}(X_2)$		
			$X_1,\ t_{1,3},\ t_{1,4}$ $X_2,\ t_{2,3},\ t_{2,4},\ X_{2,5}$ $\longrightarrow$	

$\mathbf{n_{i-1}}$		$\mathbf{n_i}$		$\mathbf{n_{i+1}}$
	$X_{i-3},\ t_{i-3,i}$ $X_{i-2},\ t_{i-2,i},\ t_{i-2,i+1}$ $X_{i-1},\ t_{i-1,i},\ t_{i-1,i+1},\ t_{i-1,i+2}$ $\longrightarrow$			
		verify $t_{i-1,i},\ t_{i-2,i},\ t_{i-3,i}$ else `reject` $r_{i-1} \leftarrow X_{i-1} - X_{i-2}$ $r_{i-2} \leftarrow X_{i-2} - X_{i-3}$ verify $r_{i-1}, r_{i-2} \in \{0, \ldots, R-1\}$ else `reject` $X_i \leftarrow X_{i-1} + r_i$ $t_{i,i+1} \leftarrow \mathrm{MAC}_{k_{i,i+1}}(X_i)$ $t_{i,i+2} \leftarrow \mathrm{MAC}_{k_{i,i+2}}(X_i)$ $t_{i,i+3} \leftarrow \mathrm{MAC}_{k_{i,i+3}}(X_i)$		
			$X_{i-2},\ t_{i-2,i+1}$ $X_{i-1},\ t_{i-1,i+1},\ t_{i-1,i+2}$ $X_i,\ t_{i,i+1},\ t_{i,i+2},\ t_{i,i+3}$ $\longrightarrow$	

Figure 5.9: A resilient data aggregation protocol for $k = 2$. Each node receives the aggregate sum from the three nodes preceding it, along with a MAC from neighbors at distance $d = 1, 2, 3$ to ensure that the sums are unaltered by any potentially malicious intermediate nodes. (Continued in Figure 5.10)

$\mathbf{n_{N-2}}$	$\mathbf{n_{N-1}}$	$\mathbf{n_N}$
X_{N-4}, $t_{N-4,N-1}$		
X_{N-3}, $t_{N-3,N-1}$, $t_{N-3,N}$		
X_{i-1}, $t_{N-2,N-1}$, $t_{N-2,N}$,		
$\longrightarrow$		
	verify $t_{N-2,N-1}$, $t_{N-3,N-1}$, $t_{N-4,N-1}$	
	else `reject`	
	$r_{N-2} \leftarrow X_{N-2} - X_{N-3}$	
	$r_{N-3} \leftarrow X_{N-3} - X_{N-4}$	
	verify $r_{N-2} \in \{0, \ldots, R-1\}$, else `reject`	
	verify $r_{N-3} \in \{0, \ldots, R-1\}$, else `reject`	
	$X_{N-1} \leftarrow X_{N-2} + r_{N-1}$	
	$t_{N-1,N} \leftarrow \mathrm{MAC}_{k_{N-1,N}}(X_i)$	
		X_{N-3}, $t_{N-3,N}$
		X_{N-2}, $t_{N-2,N}$
		X_{N-1}, $t_{N-1,N}$
		$\longrightarrow$

Figure 5.10: (Continued from Figure 5.9) A resilient data aggregation protocol for $k = 2$. Each node receives the aggregate sum from the three nodes preceding it, along with a MAC from neighbors at distance $d = 1, 2, 3$ to ensure that the sums are unaltered by any potentially malicious intermediate nodes.

node n_i is honest, that at most $k = 2$ consecutive nodes are dishonest, and that nodes $n_1, \ldots, n_{i-1}$ do not output **`reject`**. *Then, n_i either outputs* **`reject`**, *or node n_i correctly computes $X_i = \sum_{j=1}^{i} r_j$ where $r_1, \ldots, r_i \in \{0, \ldots, R-1\}$.*

Proof. We prove this by strong induction on i, first considering $i = 1, 2, 3$ as base cases.

- If node n_1 is honest, then it knows the correct value of $r_1 = X_1$, and it can correctly compute

$$\begin{aligned} t_{1,4} &= \mathrm{MAC}_{k_{1,4}}(X_1), \\ t_{1,3} &= \mathrm{MAC}_{k_{1,3}}(X_1), \text{ and} \\ t_{1,2} &= \mathrm{MAC}_{k_{1,2}}(X_1). \end{aligned}$$

- If node n_1 is dishonest, then node n_2 may be honest. If so, node n_2 receives $X_1 = r_1$ directly from n_1, with which it can verify $t_{1,2}$ is valid and that $r_1 \in \{0, \ldots, R-1\}$, outputting `reject` if not. Therefore, node n_2 can correctly compute

$$\begin{aligned} X_2 &= r_1 + r_2, \\ t_{2,5} &= \mathrm{MAC}_{k_{2,5}}(X_2), \\ t_{2,4} &= \mathrm{MAC}_{k_{2,4}}(X_2), \text{ and} \\ t_{2,3} &= \mathrm{MAC}_{k_{2,3}}(X_2), \end{aligned}$$

  or outputs `reject`.

- If both nodes n_1 and n_2 are dishonest, then node n_3 must be honest. Node n_3 receives X_1, $t_{1,3}$, $t_{1,4}$, X_2, $t_{2,3}$, $t_{2,4}$, $t_{2,5}$. From these values, node n_3 can verify that $t_{1,3}$ and $t_{2,3}$ are valid and compute $r_1 = X_1$ and $r_2 = X_2 - X_1$, verifying that $r_1, r_2 \in \{0, \ldots, R-1\}$, and outputting `reject` if not. Therefore, node n_3 can correctly compute

$$\begin{aligned} X_3 &= r_1 + r_2 + r_3, \\ t_{3,6} &= \mathrm{MAC}_{k_{3,6}}(X_3), \\ t_{3,5} &= \mathrm{MAC}_{k_{3,5}}(X_3), \text{ and} \\ t_{3,4} &= \mathrm{MAC}_{k_{3,4}}(X_3), \end{aligned}$$

  or outputs `reject`.

For the purpose of strong induction, assume an honest node n_j for $1 \leq j \leq i-1$ can correctly compute

$$X_{j-1} = \sum_{l=1}^{j-1} r_l,$$

where $r_l \in \{0, \ldots, R-1\}$, as well as correctly compute $t_{i-1,i+2}$, $t_{i-1,i+1}$, $t_{i-1,i}$.

Suppose that node n_{i-1} is honest, and that nodes n_{i-2} and n_{i-3} are possibly dishonest. Then, by induction,

$$X_{i-1} = \sum_{l=1}^{i-1} r_l,$$

where $r_l \in \{0, \ldots, R-1\}$. If $t_{i-3,i}$, $t_{i-2,i}$, and $t_{i-1,i}$ are not valid MACs, then node n_i outputs `reject`. Otherwise, the received values X_{i-3}, X_{i-2}, and X_{i-1} are the same values forwarded by nodes n_{i-3}, n_{i-2}, and n_{i-1} respectively. Because node n_{i-1} is honest, $r_{i-1} = X_{i-1} - X_{i-2} \in \{0, \ldots, R-1\}$, and node n_i can correctly compute

$$\begin{aligned} X_i &= X_{i-1} + r_i, \\ t_{i,i+1} &= \mathrm{MAC}_{k_{i,i+1}}(X_i), \\ t_{i,i+2} &= \mathrm{MAC}_{k_{i,i+2}}(X_i), \text{ and} \\ t_{i,i+3} &= \mathrm{MAC}_{k_{i,i+3}}(X_i). \end{aligned}$$

Next, suppose that node n_{i-1} is dishonest and that node n_{i-2} is honest. Then, by induction,

$$X_{i-2} = \sum_{l=1}^{i-2} r_l$$

is the correct aggregate sum at node n_{i-2}, and $t_{i-2,i}$ is a valid MAC for X_{i-2}. If $t_{i-3,i}$, $t_{i-2,i}$, or $t_{i-1,i}$ are invalid, then node n_i outputs `reject`. Otherwise, the received values X_{i-3} and X_{i-2} are the same values forwarded by nodes n_{i-3} and n_{i-2} respectively. Therefore, node n_i can correctly compute $r_{i-1} = X_{i-1} - X_{i-2}$ and verify that $r_{i-1} \in \{0, \ldots, R-1\}$, outputting `reject` if not. If node n_i does not output `reject`, then X_{i-1} is correct and node n_i can compute

$$\begin{aligned} X_i &= X_{i-1} + r_i, \\ t_{i,i+1} &= \mathrm{MAC}_{k_{i,i+1}}(X_i), \\ t_{i,i+2} &= \mathrm{MAC}_{k_{i,i+2}}(X_i), \text{ and} \\ t_{i,i+3} &= \mathrm{MAC}_{k_{i,i+3}}(X_i). \end{aligned}$$

Finally, suppose that both nodes n_{i-1} and n_{i-2} are dishonest. Then, by assumption, node n_{i-3} must be honest, and, by induction,

$$X_{i-3} = \sum_{l=1}^{i-3} r_l$$

is the correct aggregate sum at node n_{i-3}, and $t_{i-3,i}$ is a valid MAC for X_{i-3}. If $t_{i-3,i}$, $t_{i-2,i}$, or $t_{i-1,i}$ are invalid, then node n_i outputs `reject`. Otherwise, node n_i can compute $r_{i-2} = X_{i-2} - X_{i-3}$ and $r_{i-1} = X_{i-1} - X_{i-2}$ and verify $r_{i-2}, r_{i-1} \in \{0, \ldots, R-1\}$, outputting `reject` if not. If node n_i does not output `reject`, then X_{i-1} and X_{i-2} are correct and node n_i can compute

$$\begin{aligned} X_i &= X_{i-1} + r_i, \\ t_{i,i+1} &= \mathrm{MAC}_{k_{i,i+1}}(X_i), \\ t_{i,i+2} &= \mathrm{MAC}_{k_{i,i+2}}(X_i), \text{ and} \\ t_{i,i+3} &= \mathrm{MAC}_{k_{i,i+3}}(X_i). \end{aligned}$$

□

Corollary 3. *The protocol in Figure 5.9 satisfies Definition 11.*

This protocol generalizes naturally to larger values of k. Each node receives the aggregate total for each of the preceding $k+1$ nodes, along with a MAC on each of them to very authenticity. From this information, a node can compute the previous k sensor readings, verify that they are valid, and compute the updated aggregate total. The node then computes a MAC on the updated aggregate total for each of the $k+1$ nodes at distance $1, 2, 3, \ldots, k+1$. The security proof is similar to the $k = 2$ case.

5.4.1 Analysis

The communication cost of each node is dependent on both the maximum number of consecutive malicious nodes k, and the maximum communication range of each node d.

To begin, assume that $d = 1$, and each message must be routed hop-by-hop. Each node n_i must forward the current aggregate total, as well as authentication information for $k+1$ nodes at distance $1, 2, \ldots k+1$ nodes further in the network. Therefore, node n_i creates and sends X_i as well as k MACs on X_i to the next hop (giving a total of $k+2$ messages).

In addition to the messages node n_i generates, it is also responsible for forwarding authentication information from the previous k hops:

- The aggregate total X_{i-k} along with one MAC to be forwarded for node n_{i+1} (two messages).
- The aggregate total X_{i-k+1} along with two MACs to be forwarded for nodes n_{i+1} and n_{i+1} (three messages).
- $\vdots$
- The aggregate total X_{i-1} along with k MACs to be verified by nodes n_{i+1} through n_{i+k} ($k+1$ messages).

Therefore, in addition to the $k+2$ messages node n_i generates, it also forwards

$$\begin{aligned} 2+3+\ldots+k+1 &= \frac{(k+1)(k+2)}{2} - 1 \\ &= \frac{(k^2+3k+2)}{2} - 1 \\ &= \frac{1}{2}(k^2+3k) \end{aligned}$$

messages from previous hops. Therefore, the total communication cost when $d = 1$ for node n_i is

$$\frac{1}{2}(k^2+3k) + k + 2 = \frac{1}{2}(k^2+5k+4)$$

messages.

In the case that $d \geq k+1$, each message can be forwarded directly to its intended recipient. Therefore, the total communication cost is simply $2k+2$.

Finally, assume that $1 < d \leq k+1$. In this case, each message is forwarded to the correct node if it is within communication range, or d hops further in the network towards its destination. In order to approximate the total communication cost, we consider the amount of network traffic a single node not near an endpoint generates when forwarding authentication information to the next $k+1$ nodes in the network.

For ease of presentation, assume $k+1$ is a multiple of d. Then messages may need to travel up to $h = \frac{k+1}{d}$ hops in the network during aggregation.

- Authentication information sent to nodes up to distance d away is sent directly (d messages in total).

- Authentication information sent to nodes between distance $d+1$ and $2d$ travels *two* hops ($2d$ messages in total).

- $\vdots$

- Authentication information sent to nodes between distance $(h-1)d$ and hd travels h hops (hd messages in total).

Therefore, the authentication information generated by a node incurs a communication cost of

$$\begin{aligned} d + 2d + \ldots + hd &= \frac{h(h+1)}{2}d \\ &= \frac{\frac{k+1}{d}\left(\frac{k+1}{d}+1\right)}{2}d \\ &= \frac{(k+1)(k+d+1)}{2d} \\ &= \frac{k^2+kd+2k+d+1}{2d} \end{aligned}$$

If $k+1$ is not a multiple of d, then there are an additional $k+1 \mod d$ authentication messages that must travel $\lceil \frac{k+1}{d} \rceil$ hops.

Each node also forwards the aggregate total for itself to the next $k+1$ hops, generating an additional $k+1$ messages. Therefore, the total communication cost incurred by each node is

$$\frac{k^2+3kd+2k+3d+1}{2d} + \left(\left\lceil \frac{k+1}{d} \right\rceil\right)(k+1 \bmod d).$$

Because each node sends and receives the same number of messages (except for those within $k+1$ hops of an endpoint), the amount of traffic a non-endpoint node generates is identical to the number of messages it is responsible for forwarding during a run of the protocol. Therefore, the expression above describes the total number of messages each non-endpoint node must send, as well as providing an upper bound on the number of messages nodes near an endpoint must send.

As in the previous aggregation protocol, an application-specific nonce must be included in each MAC to prevent replay attacks.

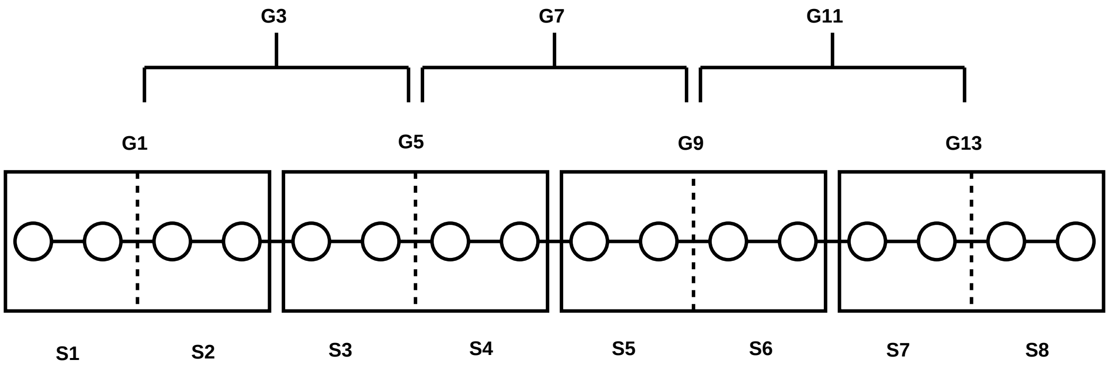


Figure 5.11: A $(16, 3)$-linear network partitioned into groups and subgroups

5.5 Aggregation Using the Group-Based KPS

In group-based key pre-distribution, each node is a member of one or more *groups*, and each group has an associated *group key* to allow secure communication among group members. In an (N, d)-linear network, it is natural to divide the network into connected groups of size $d + 1$, as this is the largest group size such that all members of a group are within each other's communication range. We refer to the group containing nodes $n_i, \ldots, n_{i+d}$ as group g_i, the members of which all possess the shared group key k_i.

Let $\mathcal{N}$ be a (N, d)-linear network where $d > 1$ and group keys are distributed as described above. For ease of presentation, assume d is odd and N is a multiple of $d + 1$. This assumption allows $\mathcal{N}$ to be uniquely partitioned into disjoint connected *subgroups* of size $\frac{d+1}{2}$:

$$\mathcal{N} = s_1 \cup s_2 \cup \ldots \cup s_{2\frac{N}{d+1}},$$

where

$$s_i = \left\{ n_{\frac{(i-1)(d+1)}{2}+1}, n_{\frac{(i-1)(d+1)}{2}+2}, \ldots, n_{\frac{i(d+1)}{2}} \right\}$$

and $s_i \cup s_{i+1}$ is a group in the underlying group-based KPS possessing some group key

$$sk_{i,i+1} = k_{\frac{(i-1)(d+1)}{2}+1}.$$

For example, consider a $(16, 3)$-linear network partitioned under this scheme. The 16 nodes are partitioned into four groups of adjacent nodes:

$$\begin{array}{ll} g_1 = \{n_1, n_2, n_3, n_4\} & g_5 = \{n_5, n_6, n_7, n_8\} \\ g_9 = \{n_9, n_{10}, n_{11}, n_{12}\} & g_{13} = \{n_{13}, n_{14}, n_{15}, n_{16}\}. \end{array}$$

```
Each $n_j \in s_i$ performs the following:

1. broadcast $(n_j, r_j)$ encrypted under key $sk_{i,i+1}$
2. set $Y_{s_i} \leftarrow r_j$
3. for each $n_l \in s_i \setminus \{n_j\}$ do
   (a) receive $(n_l, r_l)$ encrypted under key $sk_{i,i+1}$
   (b) if $r_l \in \mathcal{R}$ then $Y_{s_i} \leftarrow Y_{s_i} + r_l$
4. resolve duplicate readings / bad readings
5. output $Y_{s_i}$ (the aggregate total of all non-rejected readings)
```

Figure 5.12: In-group aggregation for subgroup s_i. Upon completion, each node in s_i correctly learns the subgroup aggregate Y_{s_i} or outputs `reject` for one or more nodes

Each group is further partitioned into two subgroups:

$$\begin{array}{lll} s_1 = \{n_1, n_2\} & s_2 = \{n_3, n_4\} & s_3 = \{n_5, n_6\} \\ s_4 = \{n_7, n_8\} & s_5 = \{n_9, n_{10}\} & s_6 = \{n_{11}, n_{12}\} \\ s_7 = \{n_{13}, n_{14}\} & s_8 = \{n_{15}, n_{16}\}. & \end{array}$$

In this scheme, $s_1 \cup s_2 = g_1$ in the underlying group-based KPS, so all members of $s_1 \cup s_2$ share the key $sk_{1,2} = k_1$. Similarly, $s_2 \cup s_3 = g_3$ in the underlying KPS, with members of both subgroups possessing the key $sk_{2,3} = k_3$.

Because all members of a group (or subgroup) share a common key, aggregation within a group (or subgroup) is trivial. Each member can simply broadcast its reading using the group key, and all members of the group can independently verify the validity of each sensor reading, as well as compute the aggregate total for the group. If any sensor reading is out of range, then the result of the protocol is `reject`. This outcome is accomplished using only a single message from each node within the group. We refer to this sub-protocol as *in-group aggregation* (see Figure 5.12).

Next, to perform aggregation across the entire network, each subgroup of nodes s_i first computes its in-group aggregate total Y_{s_i}. Aggregation then occurs subgroup-by-subgroup along the network, with each node from a subgroup updating the aggregate total with the current subgroup aggregate, and then forwarding the result to the next subgroup. When passing the aggregate total X_i from subgroup s_i to subgroup s_{i+1}, the key $sk_{i,i+1}$ is used, ensuring that all members of s_{i+1} receive the aggregate total. The use of key $sk_{i,i+1}$ also

```
for each $s_i \in \mathcal{N}$ do
    for each $n_j \in s_i$
        1. receive $(n_l, X_{i-1})$ for $n_l \in s_{i-1}$
        2. $X_{i-1} \leftarrow \texttt{SELECT}\{(n_l, X_{i-1}) | n_l \in s_{i-1}\}$
        3. $Y_{s_i} \leftarrow$ In-Group-Agg$(s_i)$
        4. $X_i \leftarrow X_{i-1} + Y_{s_i}$
        5. Broadcast $(n_j, X_i)$ encrypted under key $sk_{i,i+1}$
```

Figure 5.13: A group-based aggregation protocol

allows all other members of s_i to independently verify that the correct aggregate value was passed on by each node. As long as a majority of nodes within each subgroup are honest, a malicious coalition of nodes will not be able to forward an incorrect aggregate total without detection. Figure 5.13 demonstrates this process. In the case that malicious behavior is detected, the honest nodes could choose to continue and re-run the protocol, with those nodes that did not agree with the majority omitted during the second run. An alternative approach is to halt aggregation and forward the labels of the detected malicious nodes to the base station.

The protocol in Figure 5.13 is broken into several steps. During the first step, nodes in subgroup s_i are informed of the aggregate total X_{i-1} by each node in s_{i-1}. In the second step, each node applies a `SELECT` function to determine the correct value of X_{i-1} in the case that all readings are not identical. A possible `SELECT` function is discussed below. Next, nodes in subgroup s_i perform in-group aggregation to determine Y_{s_i}. Finally, each node computes the updated aggregate total X_i and broadcasts the result, thereby allowing the next subgroup to continue with the protocol.

In step 1 it is possible that a malicious node m_l may choose to forward an incorrect aggregate total (m_l, X'_{i-1}). The job of the `SELECT` function is to take all received (n_l, X_{i-1}) tuples and output the correct value of X_{i-1}. A simple and natural choice for the `SELECT` function is to choose the aggregate total that a majority of nodes agree on, or output `REJECT` if no majority exists. With such a `SELECT` function, the correct aggregate total can be computed with complexity $O(d)$, and it can be shown that the group-based aggregation protocol outputs the correct aggregate total whenever a majority of nodes in each subgroup is not malicious. This result is demonstrated in Theorem 22.

Both the in-group aggregation and group-based aggregation protocols are vulnerable to message flooding or node spoofing attacks, where a malicious node submits more than one reading for itself or while pretending to be another node. Without additional shared keys or assumptions, defending against such an attack in this setting is impossible. Techniques for mitigating these attacks are discussed in Section 5.5.1. The proofs below demonstrate security in the absence of spoofing attacks.

Theorem 21. *If nodes cannot spoof their identity, then the in-group aggregation protocol in Figure 5.12 allows each node in a subgroup s_i to correctly compute the subgroup aggregate total Y_{s_i}, such that Y_{s_i} contains a valid reading from each honest node in s_i, and at most one valid sensor reading for each dishonest node in s_i. If any invalid readings are received, the protocol also outputs* `reject`*.*

Proof. Suppose each node in $n_j \in s_i$ broadcasts a valid sensor reading r_j using the key $k_{i,i+1}$. Therefore, each node in s_i receives r_j directly, verifies $r_j \in \mathcal{R}$, and adds r_j to Y_{s_i}. On the other hand, if a dishonest n_j broadcasts $r_j \notin \mathcal{R}$, then it is omitted from the aggregate total and the node n_j is marked as malicious. Thus, only valid readings from dishonest nodes are included in Y_{s_i}. □

Theorem 22. *Suppose a majority of nodes in each subgroup are honest, nodes cannot spoof their identity, and subgroups $s_1, \ldots, s_{i-1}$ do not output* `reject`*. Then, using the protocol in Figure 5.13 with the* `SELECT` *function choosing the majority answer, each node in s_i learns the correct aggregate total $X_i = \sum_{j=1}^{i} Y_{s_j}$.*

Proof. Nodes in s_1 can run the in-group aggregation protocol, and, by Theorem 21, they correctly learn the aggregate total $Y_{s_1} = X_1$.

For the purpose of induction, assume nodes in s_{i-1} can correctly compute $Y_{s_{i-1}}$ and X_{i-1}, where $Y_{s_{i-1}}$ is the subgroup aggregate for s_{i-1} and X_{i-1} is the aggregate total $\sum_{j=1}^{i-1} X_{s_j}$.

If a node $n_j \in s_{i-1}$ is honest, then it broadcasts the correct message X_{i-1} using key $k_{i,i+1}$, which is received by each node in s_i. Because a majority of nodes in s_{i-1} are honest, the output of $\texttt{SELECT}\{(n_j, X_{i-1}) | n_j \in s_{i-1}\}$ is the correct aggregate total X_{i-1}. Therefore, each node in s_i will accept the correct value X_{i-1}. By Theorem 21, each node in s_i can correctly compute Y_{s_i}. Therefore, each node can correctly compute $X_i = X_{i-1} + Y_{s_i}$. □

Corollary 4. *The protocol in Figure 5.13 with the majority* `SELECT` *function satisfies Definition 11.*

Note that although the group-based KPS issues keys to each connected subset of d nodes, the group-based aggregation protocol described here only utilizes the group keys $s_1, \ldots, s_{2\frac{N}{d+1}}$.

5.5.1 Reacting to Node Spoofing Attacks

The in-group aggregation and group-based aggregation protocols in this section rely on the assumption that each node can contribute only a single reading or aggregate total in each protocol. In particular, the protocols are not secure if a malicious node m_i can pretend to be an honest node n_j. In this case, the malicious node can spoof n_j's identity by broadcasting a message of the form (n_j, r'_j), causing all nodes to receive readings of both r_j and r'_j for node n_j. Therefore, a mechanism to react to duplicate readings must be in place.

The node spoofing attack is possible due to the lack of source authentication when sending messages using group keys. There are several approaches that can be used to mitigate this, depending on the individual capabilities of each node. Some potential solutions are discussed below.

Pairwise Keys

Assume that each pair of nodes in any group possess a unique shared key. If two or more readings are received for a given node n_i, then node n_i can prove which reading is correct by sending a MAC on (n_i, r_i) to each member of the subgroup individually using its pairwise keys instead of the group key. Thus, each node learns the correct value r_i and can continue with the protocol. If multiple valid MACs are received, then it can be assumed that node n_i is compromised or malicious.

Wireless Fingerprinting

Wireless fingerprinting [91] allows one node to generate a deterministic identifier from the physical characteristics of any received message from another node. If fingerprinting is available, then two identical messages from different senders are distinguishable. There are two approaches to using fingerprinting: pre-shared fingerprints, and discovered fingerprints. In the former case, each node is pre-loaded with the fingerprint of each other node in the group, while in the latter, each node learns the fingerprint of a node after deployment.

In the case of pre-shared fingerprints, determining the true reading from a set of duplicates is trivial. If multiple messages match a single fingerprint, it can be assumed that the node is malicious. When fingerprints are not pre-loaded, during the first run of the protocol each node can record the fingerprint associated with the received message for each node. Because fingerprints are unique, each node can commit to at most one identity within the network. If one fingerprint is associated with multiple nodes, then it can be assumed that the fingerprint belongs to a malicious node. If two different fingerprints commit to the same identity there may be no possible way to tell which one is honest. In this case, both nodes can be marked as malicious as a precaution. This approach limits the impact of a malicious node to knocking out a single honest node.

Directional Antennae and Distance Bounding

Directional antennae [6] allow a node to estimate the direction of the sender of a message, which is easier to accomplish in a linear network than in a two-dimensional network. If duplicate messages are received, any message that does not originate from the correct side of the network (i.e., left or right) cannot possibly be from an honest node.

Similarly, distance bounding [92] allows a node to put an upper bound on the distance of a sender. If nodes are homogeneous, uniformly spaced, and broadcast at fixed power levels, precise distance location, rather than just an upper bound, may also be available. If duplicate messages are received, any message that does not originate from a node within the correct distance cannot possibly be from an honest node.

On their own, neither direction nor distance bound information is sufficient to determine which message from a set of duplicates came from the correct sender. However, direction and precise distance trivially solve the problem, as only a single node can be at a given distance and direction. Precise distance on its own is sufficient as well, however, determining the correct message requires all nodes in a group to participate in the process. All nodes in the next subgroup are always located later in the network, and, by assumption, this subgroup contains an honest majority of nodes. If nodes in the next subgroup also confirm the measured distances of a given message, there will be an honest majority of nodes agreeing on a specific location.

5.6 Comparing Linear Approaches

In a non-adversarial setting, resilient hop-by-hop aggregation in a simple linear network can be performed using N messages to aggregate each reading hop-by-hop, and an additional N messages to ensure integrity. Therefore, we can use a baseline of $2N$ messages as a point of comparison for our protocols. The performance results for each linear protocol are summarized in Table 5.1.

Using the optimal KPS, we presented a protocol that protects against any number of malicious nodes, provided that no two malicious nodes are directly next to each other in the network. This result is achieved with only four keys per node, and requires each node to send two messages (a single reading, and the aggregate reading), along with two authentication tags. The total cost of this protocol is $4N$ messages, with two MACs being computed by each node. The limited number of keys in this setting makes extending such an approach to a larger number of adjacent malicious nodes difficult.

Using the pairwise KPS, we presented a protocol that protects against k adjacent malicious nodes within the network. The cost of this protocol scales with the value of k. As k grows, information from $k+1$ hops back, along with the associated authentication information, must be passed along the network and verified by each node. Depending on the communication range of the nodes, the total communication complexity of the protocol is between $2(k+1)N$ messages when nodes can send authentication information directly to the intended recipient, and $\left(\frac{1}{2}(k^2+5k+6)-2\right)N$ messages when $d=1$ and all authentication information must be routed hop-by-hop.

Our final protocol utilized shared group keys and protects against localized clusters of malicious nodes. The network is divided into subgroups, and the protocol can proceed as long as there is an honest majority of nodes in each subgroup. The communication overhead in this protocol is minimal, however it may require specialized tools to protect against certain attacks. Each node broadcasts its own reading, and later broadcasts the updated aggregate, but must be active to listen to and record the messages of other nodes within its subgroup. The communication complexity of this protocol during regular operation is therefore $2N$ messages.

5.7 Moving Beyond Linear Networks

The linear protocols presented thus far provide resilient aggregation in the presence of multiple adversarial nodes, or the real-time detection of malicious behavior, with a tweakable parameter that balances the trade-off between protocol overhead and maximum number of consecutive malicious nodes in the network. These protocols were purpose-built to provide resilient aggregation in simple linear networks, and therefore utilize the specific key pre-distribution patterns expected in such a deployment. In this section we establish that the functionality of these protocols is independent of the underlying linear topology, and instead relies only on the key pre-distribution scheme used. If nearby nodes possess the relevant shared keys, then the same approach can be used to aggregate over an arbitrary tree.

Table 5.1: Summary of communication costs for the optimal, pairwise, and group-based linear aggregation protocols.

Malicious nodes	Communication cost (per node)
Optimal KPS	
$k = 1$	4
Pairwise KPS	
$k = 1$	$\frac{1}{2}(k^2 + 5k + 4)$
$d < k$ and $d \mid (k + 1)$	$\frac{k^2+3kd+2k+3d+1}{2d}$
$d < k$ and $d \nmid (k + 1)$	$\frac{k^2+3kd+2k+3d+1}{2d} + (\lceil \frac{k+1}{d} \rceil)(k + 1 \bmod d)$
$k = d - 1$	$2k + 2$
Group-based KPS	
honest majority (per subgroup)	2

5.7.1 Merging Two Paths Into One

We begin by considering the simplest relevant tree topology, where two simple linear networks converge at a single point and continue as a simple linear network. In other words, we study a family of trees where exactly one node has two children. Figure 5.14 demonstrates such a tree.

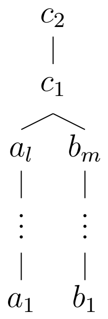


Figure 5.14: A network that consists of two simple linear networks converging into a single linear network.

By Theorem 19, nodes along any linear path in a network can either correctly aggregate their sensor readings in the presence of non-connected adversarial nodes, or detect malicious action by adversarial nodes. Our goal is to extend this process so that it can be used to merge two disjoint paths in the network at a single point, allowing aggregation to occur over a binary aggregation tree.

Let $\mathcal{A} = \{a_1, a_2, \ldots, a_l\}$ and $\mathcal{B} = \{b_1, b_2, \ldots, b_m\}$ be two disjoint paths in a network where node a_l and node b_m are both connected the same node c_1, such that they form an incomplete binary tree rooted at c_1. Let A_i and B_j represent the aggregate total of readings from nodes $\{a_1, \ldots, a_i\}$ and $\{b_1, \ldots, b_j\}$ respectively, and let C_i represent the aggregate total of the entire subtree rooted at node c_i.

Assume that no two connected nodes in the tree containing $\mathcal{A}$, $\mathcal{B}$, and c_1 are malicious, and that nodes at distance 1 and distance 2 have distinct pairwise keys shared with each other. In other words, $\mathcal{A} \cup \{c_1\}$ and $\mathcal{B} \cup \{c_1\}$ are both linear networks that satisfy the requirements of the "optimal $k = 1$ protocol". The idea is to perform aggregation over both of these linear networks independently.

Assume that node c_1 is honest. Node c_1 is the last node in the linear path $\mathcal{A} \cup \{c_1\}$, so, by Theorem 19, node c_1 can correctly learn the aggregate sum $A_l + r_{c_1}$, or it outputs `reject` if malicious behavior is detected. Similarly, node c_1 is the last node in the path $\mathcal{B} \cup \{c_1\}$ and it correctly learns $B_m + r_{c_1}$, or outputs `reject`. Therefore, node c_1 can correctly compute $C_1 = A_l + B_m + r_{c_1}$ as the correct aggregate sum for the tree rooted at c_1, or outputs `reject`.

If node c_1 is dishonest then it must be have an honest neighbor c_2, where $c_2 \notin \mathcal{A} \cup \mathcal{B}$ (i.e., node c_2 is the parent of node c_1 in the tree). Node c_2 is the last node in the path $\mathcal{A} \cup \{c_1, c_2\}$ (similarly, $\mathcal{B} \cup \{c_1, c_2\}$). By Theorem 19, node c_2 correctly learns the aggregate total $A_l + r_{c_1} + r_{c_2}$ (similarly, $B_m + r_{c_1} + r_{c_2}$), or outputs `reject`. Recall that during the "optimal $k = 1$" protocol, node c_2 has knowledge of the individual values A_l, B_m, and r_{c_1}. Therefore, node c_2 can correctly compute $C_2 = A_l + B_m + c_1 + c_2$.

5.7.2 Merging Multiple Paths

Following the approach in the previous section, a merge node c_1 can viewed as the terminal point of multiple linear networks, and it can therefore learn the aggregate total for each of them. Similarly, c_1's parent node c_2 can be viewed as the terminal point of multiple linear networks, each of which merges at node c_1. Therefore, if node c_1 is dishonest, then the honest node c_2 can compute the aggregate total for each path merging at c_1, as well as aggregate readings from additional children other than c_1. Figure 5.15 demonstrates this idea.

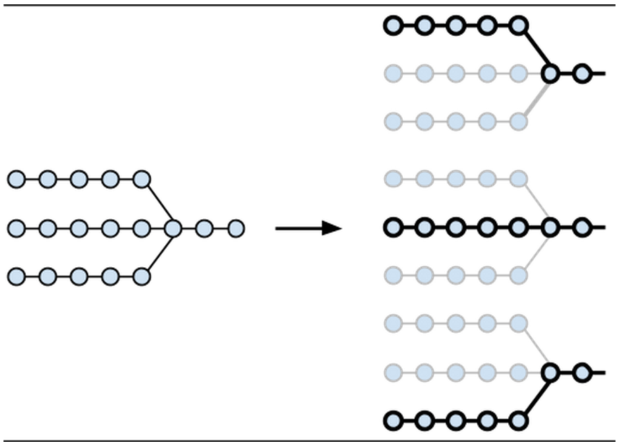


Figure 5.15: Breakdown of a tree into overlapping linear networks. Aggregation is performed over each subnetwork such that readings from the overlap are only included once. As a result of linear aggregation, both the merge point and its parent learn the correct aggregate total for each subtree. By extending each linear subnetwork to the parent of the merging node, the parent is able to verify that the merge node computed each aggregate correctly.

We now demonstrate that secure aggregation is possible over an arbitrary tree structure, given the constraint that no two consecutive nodes (i.e., no parent-child pair) are both malicious. Let $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$ be a network of N nodes arranged in a tree. The height of a node in the tree is the length of the longest path from that node to one of its descendant leaf nodes. The individual reading for node n_i is denoted by r_{n_i}, and the aggregate total of the subtree rooted at node n_i is denoted by X_{n_i}. Figure 5.16 describes an aggregation protocol for $\mathcal{N}$.

The protocol in Figure 5.16 has the same structure as its linear counterpart, with each node receiving authenticated copies of the aggregate subtotal from the previous nodes at distance 1 and distance 2. If all received messages are authentic, then the new aggregate total is computed, and the result, along with the necessary authentication information, is forwarded to the next node. It is assumed that each node knows the complete local topology, and therefore exactly which messages are expected, and which should be sent.

Theorem 23. *Suppose that the "optimal $k = 1$" tree-based aggregation protocol is being used for data aggregation in a tree-based network $\mathcal{N}$, and the following are true:*

1. *Node n_i is honest;*

- Each leaf node n_l node sends to its parent:
    1. Its sensor reading $r_{n_l} = X_{n_l}$
    2. If node n_l has a grandparent n_{l+2}, it also sends $t_{k_{n_l,n_{l+2}}} = \mathrm{MAC}_{k_{n_l,n_{l+2}}}(X_{n_l})$
- Each non-leaf node n_h receives from each child n_{h-1}:
    1. For each child n_{h-2} of node n_{h-1}:
        (a) The aggregate total $X_{n_{h-2}}$
        (b) The authentication tag t_{n_{h-2},n_h}
    2. The sensor reading $r_{n_{h-1}}$
- Each non-leaf node verifies the received applicable authentication tags, and outputs `reject` if necessary
- Each non-leaf node computes the aggregate total for each child
- Each non-leaf, non-root node n_h sends to its parent n_{h+1}:
    1. Its sensor reading r_{n_h}
    2. For each child n_{h-1}:
        (a) The aggregate total $X_{n_{h-1}}$
        (b) The received authentication tag $t_{n_{h-1},n_{h+1}}$
    3. If node n_h has a grandparent n_{h+2}, it also sends $t_{k_{n_h,n_{h+2}}} = \mathrm{MAC}_{k_{n_h,n_{h+2}}}(X_{n_h})$

Figure 5.16: The "optimal $k = 1$" tree-based aggregation protocol.

2. no two connected nodes are both dishonest; and,

3. no node in the subtree rooted at node n_i outputs ***`reject`****.*

Then node n_i either outputs ***`reject`****, or node n_i correctly computes X_{n_i}, where X_{n_i} contains the true sensor reading for any honest node, and a single valid sensor reading for each dishonest node in the subtree rooted at n_i.*

Proof. We prove this through strong induction on the height of a node h, first considering nodes at height $h = 0$ and $h = 1$ as base cases. Let n_0 be a node with height $h = 0$ (i.e., a leaf node). If node n_0 is honest, then it knows the value $X_{n_0} = r_{n_0}$ and can compute the necessary authentication tags for its grandparent. If node n_0 is dishonest, then its parent, n_1, at height 1 must be honest. Node n_1 receives $X_{n_0} = r_{n_0}$ directly from each of its children and can verify that r_{n_0} is a valid sensor reading for each of its children. Therefore, node n_1 can correctly compute X_{n_1}. and the necessary authentication tags for its grandparent.

For the purpose of strong induction, assume that any honest node n_j at height j, for $0 \leq j \leq h-1$ can correctly compute X_{n_j} and the associated authentication tags, as well as $X_{n_{j-1}}$ for any child n_{j-1}.

Let n_h be an honest node at height $h > 1$ in the aggregation tree. Without loss of generality, let node n_{h-1} be a child of node n_h, and, if it exists, let node n_{h-2} be a child of node n_{h-1}.

- If nodes n_{h-1} and n_{h-2} are honest, then node n_h correctly learns $X_{n_{h-2}}$ (if it exists) and r_{n_h}.
- Otherwise, one (but not both) of n_{h-1} or n_{h-2} are dishonest.
  - If node n_{h-1} is dishonest, then:
    - ∗ If n_{h-1} is a leaf, then $r_{n_{h-1}} = X_{n_{h-1}}$ and node n_h can directly verify that the reading is valid.
    - ∗ If n_{h-1} is not a leaf, then all of its children must be honest. By induction, the aggregate total computed by $X_{n_{h-2}}$ for the subtree rooted at n_{h-2} is correct. If $t_{n_{h-2},n_h} \neq \mathrm{MAC}_{k_{n_{h-2},n_h}}(X_{n_{h-2}})$, then node n_h outputs `reject`. Otherwise, the received value of $X_{n_{h-2}}$ is correct. If the received value $r_{n_{h-1}}$ is not valid, then node n_h outputs `reject`. Otherwise, node n_h has knowledge of correct values of $X_{n_{h-2}}$ and $r_{n_{h-1}}$.
  - If node n_{h-1} is honest, then node n_{h-2} is dishonest. By induction, node n_{h-1} correctly learns $X_{n_{h-1}}$ and $X_{n_{h-2}}$. If $\mathrm{MAC}_{k_{n_{h-2},n_h}}(X_{n_{h-2}})$, then node n_h outputs `reject`. Because node n_{h-1} is honest, the received value $X_{n_{h-2}}$ is correct and $r_{n_{h-1}}$ is valid. Therefore, node n_h correctly learns $X_{n_{h-2}}$ and $r_{n_{h-1}}$.

In each case, if node n_h does not output `reject`, then for each grandchild n_{h-2}, node n_h correctly learns $X_{n_{h-2}}$, and for each child n_{h-1}, node n_h correctly learns r_{h-1}. Therefore, for any child n_{h-1}, node n_h can correctly compute $X_{n_{h-1}}$ as the sum of all aggregate totals from grandchildren rooted at node n_{h-1} plus the reading of $r_{n_{h-1}}$. Similarly node n_h can correctly compute X_{n_h} as the sum of all aggregate totals from its children, plus its own individual reading r_{n_h}. □

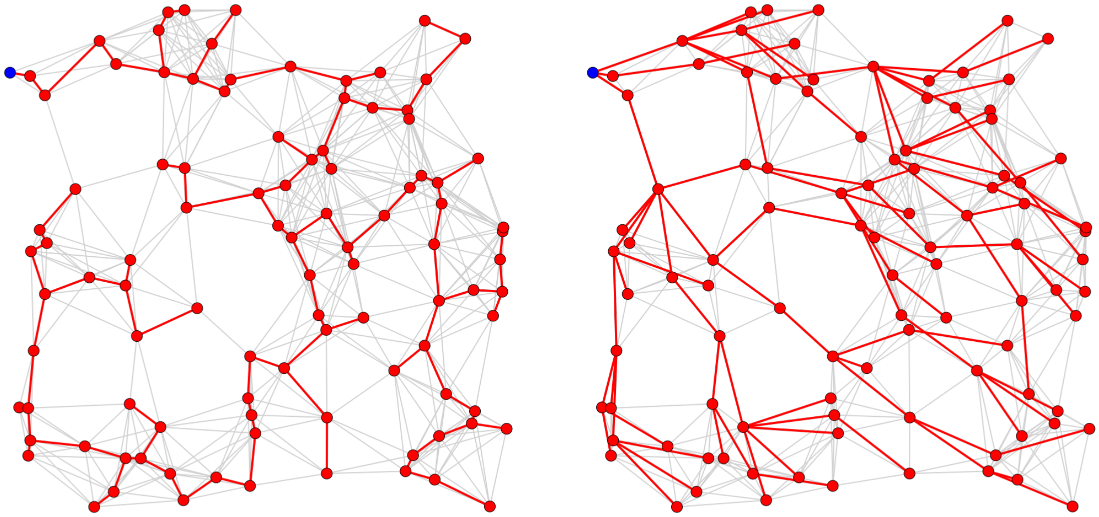
Analysis

The "optimal $k = 1$" tree-based protocol has the same message structure as the linear protocol. More specifically, the linear protocol can be viewed as a special case of the tree based protocol, with a single leaf, and each non-leaf node having exactly one child. Each node forwards its reading, an aggregate total, authentication information from one hop back, and authentication information for two hops forward. Therefore, the total number of messages is the same as in the linear case; approximately $4N$ messages total. The difference lies in the number of messages sent by each individual node. If a node has c children, then it must forward c subtree aggregates and c authentication tags to its parent. This increase in communication is balanced by the fact that for each path that merges, there is an extra leaf node in the network that is only required to send two messages. For example, the simple tree in Figure 5.15 has a single merge point where three linear subtrees come together, and therefore contains three leaf nodes. The two additional leaves send two fewer messages than most others in the network, but the as a result the merge node is required to send an additional two messages for each path it merges. This imbalance in communication cost in addressed in Section 5.10 through the use of multiple aggregation trees.

5.8 Linear Sub-Networks in Random Topologies

The performance of a tree-based aggregation protocol is highly dependent on the structure of the underlying tree. The linear-based protocols presented in this chapter perform best when a network can be naturally modeled as a set of linear networks that occasionally merge, while the more traditional tree-based approaches perform best when aggregating over a balanced tree whose depth is logarithmic in the total number of nodes. The goal of this section is not to establish the "best" aggregation model, but to explore viable possibilities with the goal of establishing linear networks as a natural sub-topology in many settings.

As a starting point, consider the random network depicted in Figure 5.17. The network consists of 100 nodes distributed randomly in a unit square. The base station is located near the top-left corner, and two nodes are connected if the distance between them is less



Total Length: 6.96
Depth: 40
Leaves: 25
Non-Merge: 52
Merge points: 23

Total Length: 16.01
Depth: 8
Leaves: 50
Non-Merge: 26
Merge points: 24

Figure 5.17: A comparison of a Euclidean minimum spanning tree and a breadth-first spanning tree generated from a random breadth-first traversal of nodes distributed in a unit square. Two nodes are connected with a grey line if they are within each other's communication range.

than or equal to 0.2. The left figure depicts a *Euclidean minimal spanning tree (MST)*, which connects all nodes in the network to the base station using the shortest possible total Euclidean distance. If all nodes are to participate in an aggregation protocol, and if the energy needed to transmit a message is proportional to the distance it must travel (i.e., nodes can intentionally broadcast at a lower power if they know their neighbor is close), then the MST represents the topology capable of aggregating all messages using the least amount of total energy. The energy cost for each node is proportional to the distance between itself and its parent in the MST. Although the MST is efficient, there are several drawbacks that may make it unsuitable in some scenarios. Namely, the individual sensor nodes must have some mechanism for determining and disseminating the MST, there is no control over the depth and degree of nodes within the tree, and the resulting communication pattern may be considered counterintuitive. For example, the longest path in the MST in Figure 5.17 is 40 hops, and terminates at a node that is only 2 hops away from the base station in the underlying communication graph.

The right portion of Figure 5.17 depicts a random *breadth-first search (BFS)* of the network. Such a traversal is generated by starting at the root and adding each a link to each unvisited neighbor, then marking it as visited. The process is repeated for each newly visited node in a random order, thus growing the height of the tree by one each iteration. This process replicates the common technique of constructing an aggregation tree by broadcasting a "HELLO" message. Each node that receives such a message chooses the sender as its parent in the aggregation tree, and rebroadcasts the message to inform its children, if any exist, that it can serve as a route back to the base station. The resulting tree in Figure 5.17 can be seen as a natural method of constructing a spanning tree over a random network, and, unlike the MST, each node lies along a short path to the base station. The depth of the tree is 8, but total length (and resulting energy consumption) is over twice that of the MST.

Aside from the individual characteristics considered above, the MST and BFS tree in Figure 5.17 differ greatly in expected degree of each node. The MST can be viewed as several linear networks that occasionally merge. In this case, every merge point in the network consists of exactly two paths merging into one. The BFS tree consists of many shorter paths that frequently merge, often with more than two paths merging at the same point. The structure of Euclidean spanning trees in random graphs has been studied before, and characteristics of the single random network considered here are typical of the expected structure of a MST in general. Steele et al. [100] have shown that the probability that a given node has a specific degree in the MST converges to a constant. Using a Monte Carlo simulation they estimate that these probabilities converge to the following:

$$\begin{aligned}
\Pr[\text{degree}(n_i) = 1] &= 0.221 \\
\Pr[\text{degree}(n_i) = 2] &= 0.566 \\
\Pr[\text{degree}(n_i) = 3] &= 0.206 \\
\Pr[\text{degree}(n_i) = 4] &= 0.007 \\
\Pr[\text{degree}(n_i) \geq 5] &= 0.000
\end{aligned}$$

In other words, over 78% of the nodes in the MST of a random network are expected to occur on a linear path (degree 1 or 2), and with high probability, the remaining nodes are merge points where only two paths come together. This observation also suggests that the MST is unlikely to be of logarithmic depth, and therefore may not be an ideal choice for many tree-based algorithms. It has since been proven that an MST always exists with maximal degree $D \leq 4$ [94]. The same paper also demonstrates that while determining a bounded-degree spanning tree for $D \leq 3$ is known to be NP-Hard (with the $D = 2$ case being the traveling salesman problem) the $D \geq 4$ case is solvable in polynomial time.

The problem of balancing the efficiency of minimum spanning trees with the depth of shortest-path trees has been studied by Khuller and Raghavachari [56], who provide an algorithm that takes a MST and a shortest path tree as input, and outputs a tree which combines the two according to a parameter γ, which controls the balance between minimizing the total length of the network and the length of individual paths in the resulting tree. Figure 5.18 demonstrates this balance visually. The MST of the depicted Euclidean graph has the same issue as the MST in Figure 5.17, where a node located near the base station ends up routing its messages over a long path. Sub-figure (d) demonstrates a balance between the shortest path tree and MST in which messages are routed in a more natural manner.

An alternative approach to providing a more natural communication pattern is to model nodes as points in a directed Euclidean graph, where a directed edge connects two nodes if they are within a fixed distance of each other, and if the edge points towards the base station. Such graphs are often referred to as drainage networks, as they can be used to model the flow of small streams of water as they flow and collect into a reservoir or basin. The use of such graphs was introduced in the context of radio communication by Bhatt and Roy [7], who consider the problem of propagating a radio transmission that originates at a base station and travels only in the positive quadrant originating at each node. Their goal was to study to properties of the *minimum directed spanning tree (MDST)* of such a network. Figure 5.19 shows an example of a MDST from their paper.

The theory of MDSTs is not well studied, but they form an interesting model for the ideal communication structure of a random network. The MDST is the communication structure with smallest total length such that each hop ensures a message moves geographically closer to its destination. Simple inspection of Figure 5.19 also reveals that linear subnetworks naturally arise in such a tree, and, aside from the base station, the number of paths converging tends to be small. Penrose and Wade [85] have also studied the properties of MDSTs, and the random networks depicted in their work also show similar behavior. Their work does establish that, unlike the regular Euclidean MST, there is no upper bound on the degree of a node in a MDST. Nevertheless, MDSTs appear to be an appealing communication structure for random sensor networks, and they appear to have significant linear substructure and merge points of low degree.

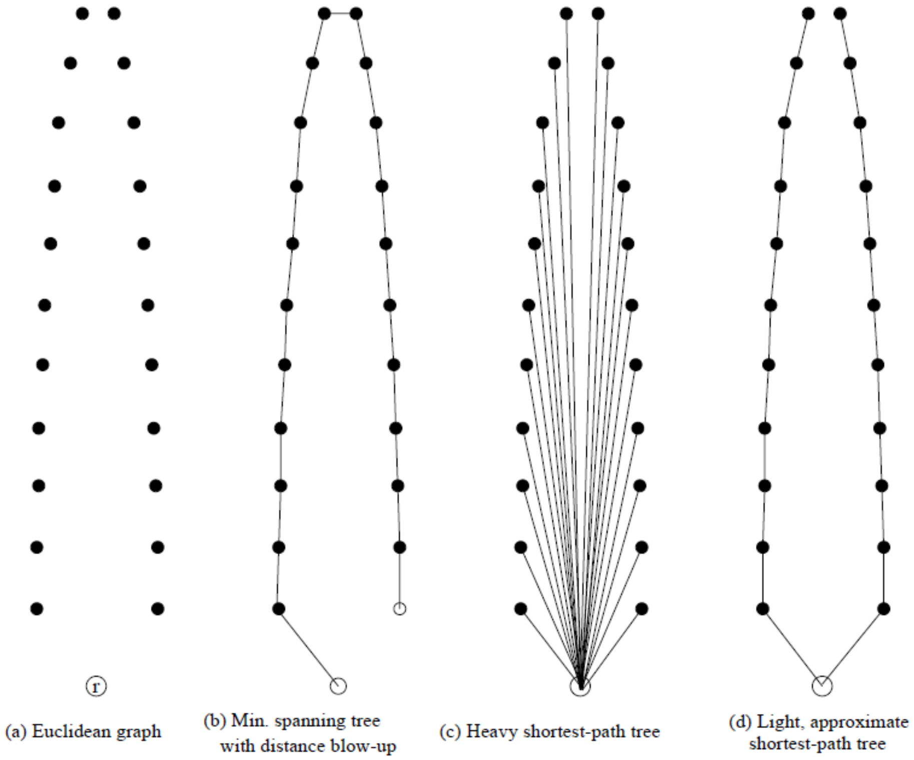

r
(a) Euclidean graph
(b) Min. spanning tree with distance blow-up
(c) Heavy shortest-path tree
(d) Light, approximate shortest-path tree

Figure 5.18: A balance between the MST and the shortest path tree in a Euclidean network. Figure taken from Khuller et al. [56].

5.9 Comparison to Previous Approaches

Having explored multiple aggregation topologies that can arise in a sensor network, we now compare the performance of the linear-based approaches in this chapter with the hierarchical approaches discussed in Section 5.1.4.

The table in Section 5.6 summarizes the communication cost of linear-based protocols when aggregating over a simple linear topology. These approaches can be used to aggregate over an arbitrary tree by considering multiple overlapping linear subnetworks and running the protocol once over each. Each leaf in the tree incurs a lower communication cost, as it does not need to route any messages from its descendants, but instead shifts these messages from itself to the first merge point in the network, where two linear networks overlap and must run the linear protocol twice. If merges are infrequent and of low degree (i.e., only

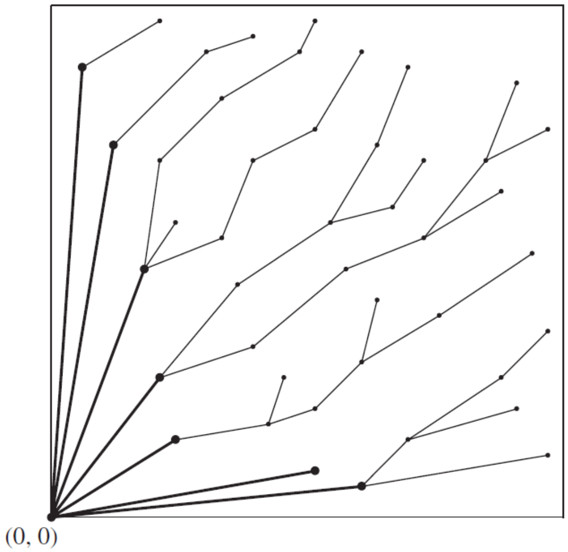


Figure 5.19: A minimum directed spanning tree. A link is only included when it lies in the positive quadrant originating at the sender. Figure taken from Bhatt and Roy [7].

two paths are merging), then the expected overhead when paths merge is at most double the cost of the linear protocol.

If multiple aggregation topologies are available, then load balancing across multiple runs of the protocol is possible. If nodes are able to alternate between acting as a leaf node and acting as a merge node, then, across multiple runs of the protocol, the amortized communication cost will approach that of the protocol when used on a simple linear network. Section 5.10 discusses techniques for accomplishing this.

Hu and Evans (HE)

Recall the HE protocol [49]. Each leaf node sends a message containing its ID, its reading, and a MAC to its parent. Each non-leaf node forwards its children's readings and MACs, as well as a MAC on the aggregate total of its children. Nodes do not possess the appropriate keys to verify any MAC at this time. Once aggregation completes, the base station reveals the keys and each node can verify the MACs that it received are correct. If no node raises an error, then the aggregation result is considered correct. This approach is essentially the same process used by the $k = 1$ protocol in this chapter. Each node provides authentication information to its grandparent that allows for malicious activity by a parent to be detected. The main difference between the linear and HE approaches lies in the key pre-distribution scheme used. The HE protocol uses an authenticated broadcast primitive to synchronize and later reveal session keys for each run of the protocol, while the linear approaches assume that nearby nodes have the necessary keys pre-loaded which is a natural assumption for the full-control deployment nature of linear networks.

1. The linear and HE approaches have similar communication costs for aggregation, but differ in the verification phase. The linear protocols provide real-time detection of malicious behavior while the HE protocols require a separate verification phase. The HE verification phase would be piggybacked on the next aggregation request that propagates across the network and therefore does not add significant communication overhead when the protocol is run multiple times. Nodes are required to cache messages until verification is complete.

2. The HE protocol only aggregates readings from leaf nodes in the aggregation tree, while the linear approaches assume all non-leaf nodes contribute a reading. It is likely possible to modify the HE protocol to allow aggregating nodes to contribute a reading, but the problem was not considered in the original proposal.

3. The HE protocol only considers non-consecutive malicious nodes ($k = 1$), while the linear protocols generalize naturally. The HE protocol could be extended in a similar manner, but it would also require nodes to cache all additional messages until the verification phase.

Chan, Perrig, Song (CPS)

The CPS protocol works by computing a hash over the aggregation tree, which is then disseminated to all nodes via an authenticated broadcast. Each node recomputes the root of the hash tree using its own input to verify that it was included in the result. This process requires each node to learn the computed values of all siblings of the nodes on the path from itself to the root. For a properly constructed tree, this can be accomplished using $O(\lg n)$ communication per node. In general, the communication cost depends on both the depth of a node, and the degree of each node along the path from itself to the root. Communication cost is not uniform across the network, as nodes further from the root require more information to compute the result.

The aggregation commitment phase of the CPS protocol requires an aggregate subtotal and an authentication tag to propagate upwards in the network, which requires a pair of messages from each node to be sent to its parent. The confirmation phase is similar, with only a single message needed per node. The aggregation verification phase is more complicated.

Recall the network in Figure 5.17. The minimum spanning tree of this network is not a balanced tree, while the BFS spanning tree is somewhat balanced and matches the aggregation topology expected by the CPS protocol. In the BFS tree, the bottom right node is at depth 8, and, if a virtual leaf node is added to each non-leaf non-root node, the path from it to the root contains 19 off-path nodes. Therefore, this node must receive 19 messages of the form (X_i, t_i) where X_i is a subtree aggregate, and t_i is an authentication tag in order to perform verification. Across all phases of aggregation, the furthest nodes in a network such as the one in Figure 5.17 incur over 20 messages per node. This cost decreases linearly as nodes get closer to the base station, with a minimum of 5 messages (2 aggregation commit messages, 2 verification messages, and 1 confirmation message) in the best case.

By design, the CPS protocol does not detect malicious activity until the confirmation phase; however, the CPS protocol can detect malicious behavior from any number of malicious nodes. A single malicious node has the ability to lie during confirmation, thus invalidating the entire result on its own. Such denial of service attacks are usually considered outside the scope of an aggregation protocol, as they are readily detected and can be addressed via other means. The linear protocols detect malicious behavior as it occurs, thus allowing a protocol to be aborted early to save energy, and also reveal the possible malicious nodes within a reasonably small radius in the network. The trade-off for such an approach is an upper bound on the number of connected malicious nodes. The linear protocols can protect against a large number of malicious nodes but they require that honest nodes exist with sufficient frequency to detect any misbehavior.

The linear approaches compare favorably to the CPS protocol for small values of k. For example, consider the MST in Figure 5.17 with $k = 1$. The communication requirement is 2 messages per leaf, 4 messages per non-merge node, and 6 messages per merge node. The worst-case congestion in the MST using the linear protocol is similar to the best case communication using the CPS protocol. For larger values of k, the resulting increase in communication overhead for merge points (and the nodes following them) can be computed from the formulas in Figure 5.1.

5.10 Grid-Based Networks and Load Balancing

The technique used in this chapter to translate a linear aggregation protocol to a tree-based aggregation protocol decomposes a tree into multiple overlapping linear networks. In a simple linear network, almost every node is a non-endpoint in the network and incurs the same communication cost. In a tree, a large number of leaves exist, each of which is the origin of a path that will eventually be merged in the aggregation tree. The additional authentication messages that would have been handled by a node in a simple linear network are instead shifted to the nearest merge point, where two linear subnetworks overlap. In this section we present some preliminary techniques that can be used to balance the communication cost across multiple runs of the protocol.

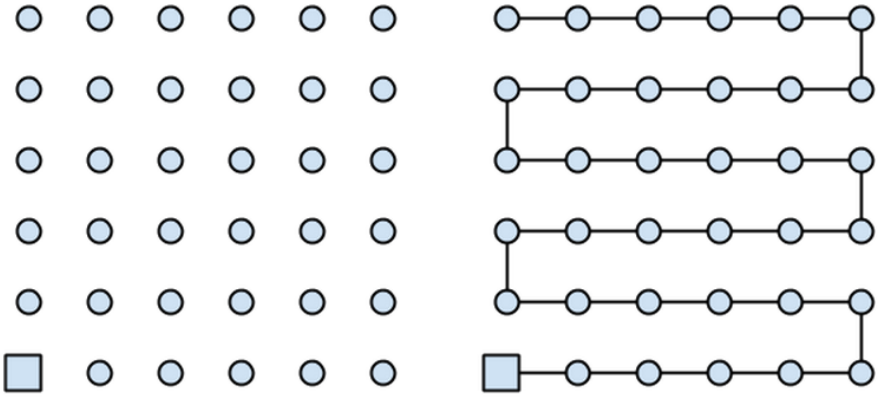


Figure 5.20: A simple grid-based sensor network and a trivial linear spanning tree. The square node denotes the base station.

Simple linear networks represent the simplest non-trivial network topology for aggregation. Despite their simplicity, they arise naturally, both due to linear deployment patterns and as important subnetworks in many settings. Another common deployment pattern is to place sensor nodes regularly in a grid pattern over a two-dimensional area. Such deployment patterns are called *grid networks*, and represent a natural communication structure for monitoring a square or rectangular area. The symmetric nature of grid-based deployments also makes it easy to define small families of spanning trees that balance the per-node communication cost of aggregation over multiple runs of the protocol.

Figure 5.20 demonstrates a simple grid based network and a trivial linear spanning tree. Provided that nearby nodes share the appropriate keys, such a spanning tree allows

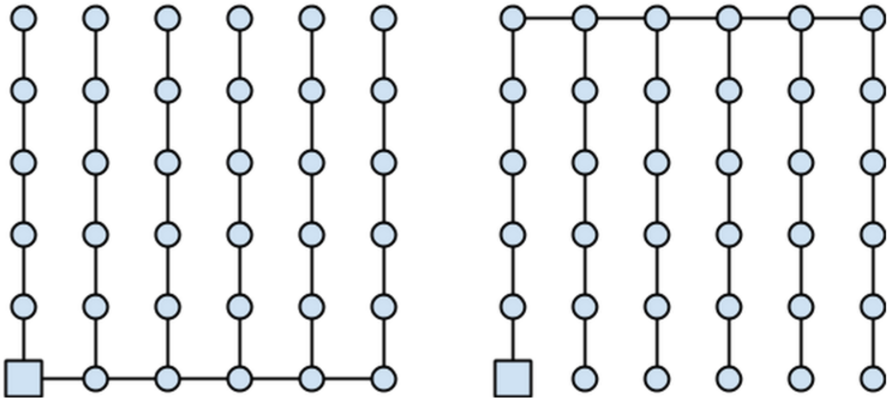


Figure 5.21: A pair of spanning trees for a grid network with communication range $d = 1$, where each merge node in one tree is a leaf node in the other.

for linear protocols to be used without modification. Figure 5.21 demonstrates a pair of spanning trees that are not linear and have much lower depth than the trivial linear spanning tree. This pair of trees has the property that all communication takes place with nodes at distance $d = 1$, and any node that acts as a merge point for two paths in the first tree is a leaf in the second tree. Therefore, over multiple runs of the protocol, the additional overhead of merging paths of offset by the reduced communication cost of acting as a leaf node. Figure 5.22 shows a second possible family of spanning trees if nodes are able to communicate over a distance of at least $d = 1.5$ to their diagonal neighbors in the grid. In each case, nearby nodes must have the appropriate shared keys in each spanning tree to enable resilient aggregation.

The problem of load balancing across multiple runs of a protocol is not unique to the tree-based approaches presented in this chapter. The CPS protocol suffers from a similar problem, where the deeper a node is in the tree, the greater the number of messages required for it to verify that its reading appears in the final aggregate. It was argued in Section 5.8 that linear subnetworks play an important role in many general settings, and an overview of several techniques for generating an aggregation tree was considered. The load balancing issue, both in the protocols presented here and in other protocols, such as the CPS protocol, suggest an interesting direction for future work in spanning tree research: generating multiple spanning trees over the same network with certain nice properties, that also satisfy certain constraints. For example, it may be desirable to have a family of trees where the average degree or the average depth of each node is constant across the

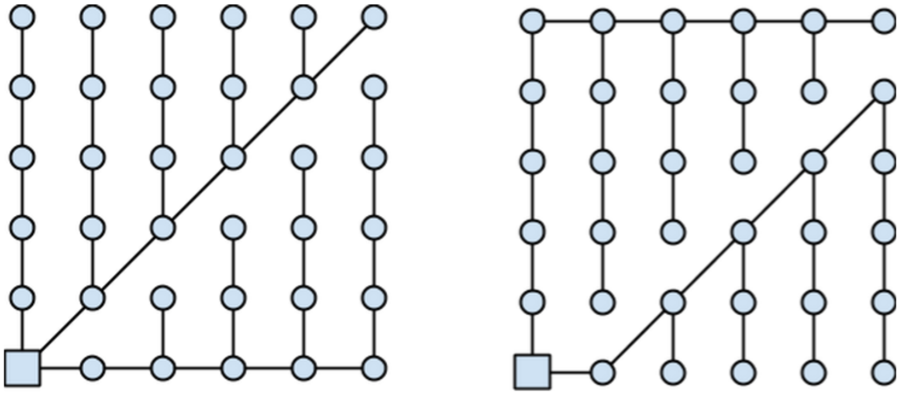


Figure 5.22: A pair of spanning trees for a grid network with communication range $d = 1.5$, where each merge node in one tree is a leaf node in the other.

family, such that each tree is also a good approximation of a minimal spanning tree. The use of multiple base stations is likely to make this problem easier. For example, placing base stations at each corner and the center of a square network and then computing the minimum spanning tree rooted at each of these base stations may result in a set of spanning trees that would balance load over multiple runs of an aggregation protocol.

5.11 Group-Based Aggregation for Grid Networks

The previous section examined some simple pairs of spanning trees that allow for load-balanced aggregation in a grid topology using tree-based aggregation. These protocols inherit the same key requirements and adversarial assumptions as the original protocol, where all nodes at distance $d \leq k+1$ in the tree must have unique pairwise keys shared with each other in order to protect against up to k connected malicious nodes in the aggregation tree. The group-based protocol allows for a different set of adversarial assumptions, where a majority of nodes in any given subgroup are required to be honest. This protocol can also be extended naturally to a grid-based setting with little modification.

At its core, the group-based protocol simply utilizes the broadcast nature of communication to perform a public vote. If a majority of the received messages are identical and a majority of the nodes are honest, then that message must be correct. The fixed nature of deployment in a linear network allowed for a simple selection of groups and subgroups that exploit this fact to propagate an updated aggregate total, one subgroup at a time, towards the base station. Grid networks also provide a natural mapping into groups and subgroups if the communication range is large enough.

Given a grid network, let $2r$ be the largest integer such that all nodes within a $2r \times 2r$ sub-grid are within each other's communication range. Each of these $2r \times 2r$ sub-grids defines a group which can be partitioned into four $r \times r$ subgroups. If the entire network is partitioned into groups of this form, each possessing a group key, then any pair of adjacent subgroups have a group key in common, and any node in these subgroups can broadcast to all members of both subgroups in a single message. Figure 5.23 demonstrates this for a network where $r = 2$.

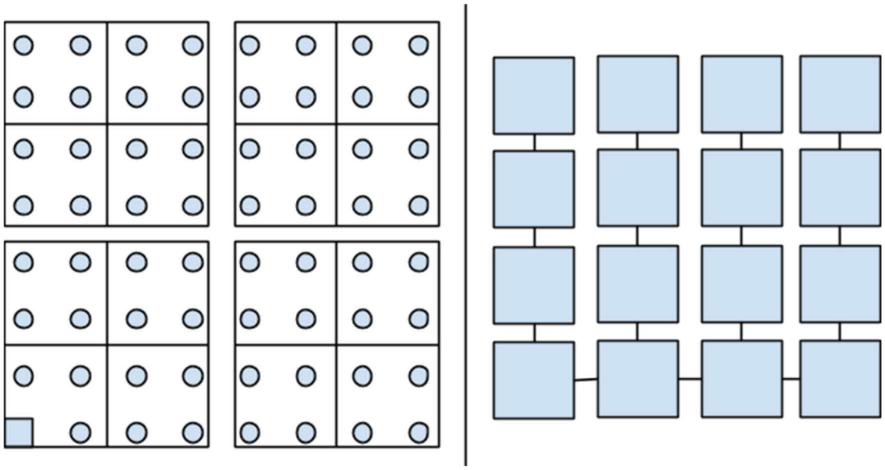


Figure 5.23: A grid network partitioned into groups of size 4×4, and subgroups of size 2×2. The right figure shows a possible aggregation topology for the group-based aggregation protocol.

In order to perform aggregation, nodes perform in-group aggregation to learn the subgroup aggregate total, and then broadcast the result using the appropriate group key. If two paths of subgroups merge, the nodes in the subgroup performing the merge simply run the protocol once for each path, and combine them together when broadcasting the updated aggregate total. If necessary, load balancing can be achieved using balanced families of aggregation trees, as discussed in Section 5.10. This approach is resilient as long as a majority of nodes in each subgroup are honest. In other words, the group-based aggregation protocol can detect up to $\lfloor \frac{r^2-1}{2} \rfloor$ malicious nodes in any subgroup, regardless of their physical layout in the underlying topology.

5.12 Summary and Remarks

This chapter presented three resilient aggregation techniques for linear topologies based on three natural key pre-distribution schemes for linear networks. These protocols were designed with a minimal set of assumptions and protect against small subsets of malicious nodes by relying on nearby honest nodes to exchange enough information to verify all intermediate nodes have behaved correctly. When pairwise keys are available, this approach incurs overhead proportional to k, the maximum number of connected malicious nodes. If nodes have long communication range and can transmit messages directly to their intended receiver, then the protocols incur a communication overhead that is linear in k. Otherwise, authentication information must be routed over multiple hops, reaching a maximum overhead that is quadratic in k. Group-based aggregation exploits the broadcast nature of communication and performs aggregation using a consensus-based approach. In each case, malicious behavior is detected as it occurs, instead of during a second verification phase. This real-time detection allows for early termination of the protocol if malicious activity occurs. For each protocol a proof of correctness is provided, and a precise analysis of communication overhead was computed.

Following the development of aggregation protocols for linear topologies, a method of generalizing the approach to more general networks was discussed. It was established that linear networks arise naturally in a variety of settings, and that many aggregation trees can be characterized as a small set of linear networks that occasionally merge, rather than a somewhat balanced tree of logarithmic depth. Traditional hierarchical aggregation approaches assume a balanced tree, while the protocols considered here perform well in the opposite case.

A key weakness of some aggregation protocols, including those presented here, was identified. Namely, that the per-node cost of aggregation is not uniform. Methods of addressing this problem for tree-based protocols in the context of a grid topology were discussed, and the more general problem of constructing a family of spanning trees that naturally provide load balancing was identified. This problem motivates future research in the area that is of both theoretical and practical interest. It was also demonstrated that the group-based aggregation protocol for linear networks can be naturally extended to grid-based networks without significant modification, which allows for extremely efficient aggregation in settings where each node can broadcast to a large group.

Chapter 6

Conclusion

This thesis presented a family of protocols that span the lifetime of a sensor network. In order to efficiently facilitate secure communication between nodes after deployment, we used a key pre-distribution scheme to issue a small number of symmetric keys to nodes during the pre-deployment phase. In particular, we presented a new family of *flexible* key pre-distribution schemes based on transversal designs. Many combinatorial design-based schemes, including transversal design-based schemes, suffer from strict constraints on the selection of network parameters. The new schemes presented in this thesis address this problem directly by decomposing a transversal design of strength t into several copies of a lower-strength design. Utilizing the theory of partially balanced t-designs, we demonstrated how to derive performance metrics for a KPS built from any subset of these lower-strength designs. Experimental results were included that demonstrate that choosing fixed-size subsets using lower-strength designs, or choosing random subsets of any size, both closely approximate the performance of the KPS where all nodes are included, as long as the subset is sufficiently large.

The technique of decomposing a design-based KPS into smaller pieces increases flexibility and could be applied to designs other than transversal designs. Similarly, the idea of nesting designs, such as in the MS-KPS, is worth pursuing further. Simple Blom schemes were considered for the inner KPS, but it is possible that other schemes might provide useful functionality. A greater understanding of the trade-off between the increase in storage and computation versus the increase in resilience as KPSs are nested poses and interesting direction for future work. In particular, a comparison of the quadratic scheme and the linear MS-KPS using a Blom scheme would be useful.

The decomposable schemes utilized the natural sub-structure of transversal designs.

A design of strength t can be decomposed into p copies of a design of strength $t - 1$. Decomposing the design in this manner retains enough structure to use results on partially balanced t-designs to analyze the connectivity. A drawback to this approach is that no two blocks within a single B_i from the resolution intersect, thereby lowering the connectivity of the network when a small number of them are deployed. A possible direction for future work is to investigate other deterministic methods for selecting subsets of a transversal design such that even small subsets of the network closely approximate the performance of the entire design. Similarly, other families of designs may be decomposable in a manner that yields better performance for deterministic subsets.

Linear key pre-distribution schemes, such as the strength 2 designs considered in this thesis, have a particularly useful property: any given pair of keys is issued to exactly one node. In other words, by proving possession of two specific keys, a node can prove its entire list of keys. This property was utilized to solve the network discovery problem, one of the main problems during the setup phase of the network. We presented a collaborative protocol where nodes directly verify keys with their neighbors and then cast "votes" publicly to inform their neighbors of the result. Each node records the votes locally and applies an "accept rule" to determine if there is enough information to accept the identity of a node as correct. As nodes accept or reject the identity of their neighbors, they broadcast the result to inform other nodes of the outcome. By repeating this process, knowledge of the network propagates and continues to grow. As long as an honest majority exists for each vote, a malicious node cannot inject false routing information without detection. Because this approach propagates the complete network topology, it can be used to establish multiple node-disjoint paths, thereby enabling the use of secure multi-path protocols, whose security relies on the existence of such paths.

The network discovery protocol considered here requires specific constraints on the capability of sensor nodes. Without any prior knowledge of the topology or public-key infrastructure, assumptions about the communication range of nodes, and their ability to computer fingerprints and directional information on incoming messages was necessary. These assumptions were used to build a set of sufficient conditions to securely discover neighbors, but no proof was given that these conditions were necessary. It would be interesting to demonstrate that these assumptions necessary, or to eliminate them if they are not.

Sensor networks have found many applications in settings where the communication topology is inherently linear. Even in settings where the underlying topology is not linear, nodes are typically concerned with efficiently routing information towards the base station, which generally takes place along a linear subnetwork. Other important communication structures, such as many types of spanning trees, naturally contain long linear subnetworks or can be characterized as sets of linear networks that occasionally merge. Targeting the operational phase of a sensor network, we presented three approaches to data aggregation in linear networks, each based on a specific natural key pre-distribution scheme for linear networks. These protocols allow for data to be aggregated hop-by-hop as it moves towards the base station, preventing an adversary from modifying the aggregate total by more than a single sensor reading without detection. We then demonstrated how this approach can be adapted to work in more general network settings without incurring significant overhead. This approach works well for networks with aggregation trees that are unbalanced or have merge points of low degree. Many tree-based approaches have worse-case performance in linear networks and are intended for use in settings where the aggregation topology is a nearly balanced tree.

The investigation of linear subnetworks identified some interesting research problems. In particular, a method for computing a family of spanning trees that balance communication cost across multiple executions of an aggregation protocol would be useful. Some simple examples were presented for grid networks, but the general problem has applications for both linear and traditional hierarchical approaches. A better understanding of how to generate spanning trees that are subject to specific constraints, particularly for random topologies, is likely have interesting applications for sensor network research.

The aggregation protocols considered here were concerned with resilience, but not with privacy. In some settings it may be desirable for aggregation to both private and resilient, such that individual readings, or even the aggregate total itself, are not revealed to any node, except for the base station. Adapting the protocols considered here to provide privacy is difficult, as they explicitly rely on nearby nodes to ensure readings are valid.

APPENDICES

Appendix A

Summary of Notation

$\mathcal{N}$		A sensor network
	N	network size
	n_i, $(1 \leq i \leq N)$	a node in $\mathcal{N}$
	$k_{i,j}$, $(1 \leq i, j \leq N)$	key shared by n_i and n_j
	C	# common shared-key neighbors of n_i and n_j
	$\mathcal{N}_i$, $(1 \leq i \leq N)$	neighborhood of node n_i
KPS		Key Pre-distribution Scheme
	$\mathcal{K}$	set of all keys in the KPS
	v	total number of keys $\lvert\mathcal{K}\rvert$
	k	number of keys stored by a node
	Pr_1	probability that two nodes have a common key in a given KPS
	Pr_2	probability that two hop path exists when two nodes do not share a key
	fail(s)	probability a link in the network is compromised after s nodes are compromised
	η	intersection threshold (min. # shared keys)
	ρ	connectivity / resilience trade-off $\frac{Pr_1}{\text{fail}(1)}$
Designs		
	X	set of points

	$\mathcal{A}$	collection of blocks
	v	number of points $\lvert X\rvert$
	b	number of blocks $\lvert\mathcal{A}\rvert$
	r	degree of points in a regular design
	k	rank of blocks in a uniform design
	λ	block intersection size
	t	strength of a transversal design
	$\mathcal{H}$	a partition of X (k groups of size n)
	p	a prime or prime power
	α	number of links a block is contained in
	β	number of links a block breaks
	L	total number of links
	μ'	number of other blocks containing a given point
Aggregation		
	r_i	sensor reading of node n_i
	R	upper bound on sensor readings
	$\mathcal{R}$	the set of valid sensor readings $\{0, 1, \ldots, R\}$
	$\mathcal{M}$	the set of malicious sensor nodes
	m_i	a malicious sensor node (replaces n_i)
	d	max communication distance of a node
	r	key sharing "radius" for linear network KPSs
	$k_{i,j}$	pairwise key between nodes n_i and n_j
	k	max number of consecutive malicious nodes
	X_i	aggregate total for the first i nodes or subgroups
	$t_{i,j}$	authentication tag (MAC) computed using $k_{i,j}$
	G_i	a group with leftmost node is node n_i
	s_i	a subgroup (see Figure 5.11)
	Y_i	aggregate total for subgroup i

Appendix B

Data Tables

Table B.1: Resilience of random KPSs derived from a TD(2,15,71). This data was used to generate Figure 3.2.

l	fail(1) (mean)	fail(1) (std. dev.)	fail(1) (min)	fail(1) (max)
2	0.013749	0.000642	0.011989	0.015357
3	0.013660	0.000381	0.012879	0.014684
4	0.013687	0.000278	0.013134	0.014481
5	0.013702	0.000234	0.013158	0.014362
6	0.013704	0.000179	0.013294	0.014108
7	0.013676	0.000140	0.013338	0.014077
8	0.013687	0.000136	0.013356	0.014063
9	0.013707	0.000109	0.013418	0.013950
10	0.013690	0.000094	0.013476	0.013964
11	0.013682	0.000077	0.013505	0.013850
12	0.013698	0.000071	0.013552	0.013897
13	0.013696	0.000068	0.013517	0.013836
14	0.013685	0.000058	0.013558	0.013820
15	0.013691	0.000055	0.013528	0.013841
16	0.013685	0.000055	0.013586	0.013830
17	0.013694	0.000053	0.013583	0.013862
18	0.013692	0.000044	0.013579	0.013800
19	0.013694	0.000042	0.013602	0.013808
20	0.013694	0.000042	0.013582	0.013812

Table B.1: (continued)

l	fail(1) (mean)	fail(1) (std. dev.)	fail(1) (min)	fail(1) (max)
21	0.013693	0.000037	0.013588	0.013780
22	0.013694	0.000033	0.013602	0.013792
23	0.013687	0.000034	0.013603	0.013760
24	0.013693	0.000031	0.013632	0.013780
25	0.013692	0.000025	0.013614	0.013746
26	0.013690	0.000026	0.013592	0.013749
27	0.013690	0.000025	0.013631	0.013743
28	0.013692	0.000021	0.013630	0.013737
29	0.013691	0.000019	0.013633	0.013730
30	0.013688	0.000019	0.013639	0.013729
31	0.013693	0.000020	0.013655	0.013749
32	0.013693	0.000018	0.013645	0.013732
33	0.013693	0.000016	0.013661	0.013752
34	0.013693	0.000016	0.013659	0.013737
35	0.013695	0.000014	0.013667	0.013724
36	0.013691	0.000014	0.013655	0.013727
37	0.013694	0.000012	0.013664	0.013725
38	0.013694	0.000015	0.013664	0.013735
39	0.013693	0.000012	0.013662	0.013726
40	0.013691	0.000012	0.013668	0.013720
41	0.013693	0.000011	0.013668	0.013726
42	0.013695	0.000013	0.013664	0.013726
43	0.013693	0.000010	0.013674	0.013725
44	0.013693	0.000009	0.013664	0.013712
45	0.013692	0.000008	0.013673	0.013715
46	0.013694	0.000007	0.013679	0.013715
47	0.013693	0.000007	0.013679	0.013709
48	0.013693	0.000008	0.013676	0.013715
49	0.013693	0.000007	0.013676	0.013710
50	0.013693	0.000007	0.013673	0.013710
51	0.013694	0.000005	0.013682	0.013709
52	0.013693	0.000006	0.013679	0.013705
53	0.013694	0.000005	0.013683	0.013707
54	0.013694	0.000005	0.013681	0.013704
55	0.013694	0.000004	0.013683	0.013707

Table B.1: (continued)

l	fail(1) (mean)	fail(1) (std. dev.)	fail(1) (min)	fail(1) (max)
56	0.013693	0.000004	0.013683	0.013703
57	0.013693	0.000004	0.013681	0.013706
58	0.013693	0.000003	0.013683	0.013704
59	0.013693	0.000003	0.013684	0.013698
60	0.013693	0.000003	0.013685	0.013700
61	0.013693	0.000002	0.013688	0.013698
62	0.013693	0.000002	0.013688	0.013700
63	0.013693	0.000002	0.013687	0.013702
64	0.013693	0.000002	0.013689	0.013697
65	0.013693	0.000001	0.013689	0.013697
66	0.013693	0.000001	0.013690	0.013697
67	0.013693	0.000001	0.013691	0.013696
68	0.013693	0.000001	0.013692	0.013695
69	0.013693	0.000000	0.013692	0.013694
70	0.013693	0.000000	0.013693	0.013694
71	0.013693	0.000000	0.013693	0.013693

Table B.2: Connectivity of random KPSs derived from a TD(2,15,71). This data was used to generate Figure 3.2.

l	fail(1) (mean)	fail(1) (std. dev.)	fail(1) (min)	fail(1) (max)
1	0.208129	0.008217	0.185111	0.228169
2	0.208647	0.003964	0.197283	0.218659
3	0.208178	0.002706	0.202719	0.215121
4	0.208296	0.001944	0.204200	0.213159
5	0.208403	0.001608	0.204138	0.212795
6	0.208455	0.001297	0.204861	0.211721
7	0.208241	0.001010	0.205467	0.210976
8	0.208214	0.000963	0.205741	0.210926
9	0.208424	0.000790	0.206175	0.210355
10	0.208313	0.000695	0.206834	0.210374

Table B.2: (continued)

l	Pr_1(mean)	Pr_1(std. dev.)	Pr_1(min)	Pr_1(max)
11	0.208234	0.000541	0.206914	0.209409
12	0.208359	0.000501	0.207433	0.209764
13	0.208359	0.000477	0.207073	0.209418
14	0.208284	0.000427	0.207436	0.209333
15	0.208332	0.000422	0.207219	0.209515
16	0.208267	0.000397	0.207473	0.209335
17	0.208340	0.000391	0.207518	0.209341
18	0.208324	0.000329	0.207455	0.209105
19	0.208339	0.000311	0.207681	0.209109
20	0.208333	0.000309	0.207523	0.209267
21	0.208333	0.000277	0.207566	0.209007
22	0.208340	0.000243	0.207638	0.209064
23	0.208282	0.000248	0.207686	0.208775
24	0.208336	0.000230	0.207860	0.208999
25	0.208327	0.000184	0.207737	0.208744
26	0.208307	0.000198	0.207572	0.208763
27	0.208309	0.000186	0.207903	0.208706
28	0.208322	0.000160	0.207863	0.208648
29	0.208314	0.000145	0.207887	0.208595
30	0.208295	0.000146	0.207896	0.208618
31	0.208337	0.000148	0.208046	0.208760
32	0.208330	0.000136	0.207953	0.208620
33	0.208334	0.000125	0.208096	0.208799
34	0.208331	0.000122	0.208052	0.208685
35	0.208344	0.000109	0.208120	0.208560
36	0.208318	0.000109	0.208046	0.208621
37	0.208338	0.000094	0.208113	0.208588
38	0.208338	0.000114	0.208114	0.208655
39	0.208330	0.000089	0.208108	0.208568
40	0.208316	0.000091	0.208140	0.208556
41	0.208336	0.000086	0.208144	0.208569
42	0.208345	0.000098	0.208104	0.208569
43	0.208331	0.000074	0.208187	0.208585
44	0.208334	0.000065	0.208109	0.208480
45	0.208328	0.000060	0.208179	0.208492

Table B.2: (continued)

l	Pr_1(mean)	Pr_1(std. dev.)	Pr_1(min)	Pr_1(max)
46	0.208335	0.000053	0.208232	0.208499
47	0.208334	0.000053	0.208223	0.208458
48	0.208331	0.000058	0.208205	0.208499
49	0.208330	0.000050	0.208211	0.208460
50	0.208332	0.000051	0.208178	0.208459
51	0.208339	0.000041	0.208246	0.208457
52	0.208332	0.000042	0.208226	0.208426
53	0.208338	0.000034	0.208256	0.208440
54	0.208339	0.000036	0.208245	0.208418
55	0.208336	0.000033	0.208255	0.208437
56	0.208333	0.000030	0.208256	0.208410
57	0.208334	0.000028	0.208241	0.208427
58	0.208329	0.000026	0.208255	0.208417
59	0.208331	0.000023	0.208262	0.208371
60	0.208332	0.000021	0.208273	0.208386
61	0.208333	0.000017	0.208295	0.208371
62	0.208335	0.000017	0.208291	0.208387
63	0.208334	0.000016	0.208285	0.208397
64	0.208332	0.000012	0.208301	0.208361
65	0.208332	0.000010	0.208305	0.208360
66	0.208332	0.000009	0.208312	0.208360
67	0.208333	0.000007	0.208316	0.208358
68	0.208334	0.000005	0.208321	0.208347
69	0.208333	0.000003	0.208324	0.208343
70	0.208333	0.000002	0.208329	0.208339
71	0.208333	0.000000	0.208333	0.208333

References

[1] Kemal Akkaya and Mohamed F. Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3):325–349, 2005.

[2] Ian F. Akyildiz, Wei Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: A survey. *Comput. Netw.*, 38(4):393–422, 2002.

[3] Jamal N. Al-Karaki and Ahmed E. Kamal. Routing techniques in wireless sensor networks: a survey. *Wireless Communications, IEEE*, 11(6):6–28, 2004.

[4] Joaquin G. Alfaro, Michel Barbeau, and Evangelos Kranakis. Secure localization of nodes in wireless sensor networks with limited number of truth tellers. In *Communication Networks and Services Research Conference, 2009. CNSR '09*, pages 86–93, 2009.

[5] Frederik Armknecht, Dirk Westhoff, Joao Girao, and Alban Hessler. A lifetime-optimized end-to-end encryption scheme for sensor networks allowing in-network processing. *Comput. Commun.*, 31(4):734–749, 2008.

[6] Joshua Ash and Lee Potter. Sensor network localization via received signal strength measurements with directional antennas. In *Proceedings of the Forty-Second Annual Allerton Conference on Communication, Control, and Computing*, pages 1861–1870, 2004.

[7] Abhay G. Bhatt and Rahul Roy. On a random directed spanning tree. *Advances in Applied Probability*, 36(1):19–42, 2004.

[8] Rabindra Bista and Jae-Woo Chang. Privacy-preserving data aggregation protocols for wireless sensor networks: A survey. *Sensors*, 10(5):4577–4601, 2010.

[9] Simon R. Blackburn, Tuvi Etzion, Keith M. Martin, and Maura B. Paterson. Distinct difference configurations: Multihop paths and key predistribution in sensor networks. *IEEE Transactions on Information Theory*, 56(8):3961–3972, 2010.

[10] Simon R. Blackburn, Keith M. Martin, Maura B. Paterson, and Douglas R. Stinson. Key refreshing in wireless sensor networks. In Reihaneh Safavi-Naini, editor, *Information Theoretic Security*, volume 5155 of *Lecture Notes in Computer Science*, pages 156–170. Springer Berlin Heidelberg, 2008.

[11] Rolf Blom. Non-public key distribution. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology – CRYPTO '82*, pages 231–236. Springer US, 1983.

[12] Mausumi Bose, Aloke Dey, and Rahul Mukerjee. Key predistribution schemes for distributed sensor networks via block designs. *Designs, Codes and Cryptography*, 67(1):111–136, 2013.

[13] Ghalem Boudour, Aubin Lecointre, Pascal Berthou, Daniela Dragomirescu, and Thierry Gayraud. On designing sensor networks with smart antennas. In *7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems*, pages 349–356, Toulouse, France, 2007.

[14] Stefan Brands and David Chaum. Distance-bounding protocols. In Tor Helleseth, editor, *Advances in Cryptology – EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 344–359. Springer Berlin Heidelberg, 1994.

[15] Levente Buttyán, László Dóra, and István Vajda. Statistical wormhole detection in sensor networks. In Refik Molva, Gene Tsudik, and Dirk Westhoff, editors, *Security and Privacy in Ad-hoc and Sensor Networks*, volume 3813 of *Lecture Notes in Computer Science*, pages 128–141. Springer Berlin Heidelberg, 2005.

[16] Levente Buttyán, Péter Schaffer, and István Vajda. CORA: Correlation-based resilient aggregation in sensor networks. *Ad Hoc Netw.*, 7(6):1035–1050, 2009.

[17] Alvaro A. Cardenas, Tanya Roosta, and Shankar Sastry. Rethinking security properties, threat models, and the design space in sensor networks: A case study in SCADA systems. *Ad Hoc Netw.*, 7(8):1434–1447, 2009.

[18] David W. Carman, Peter S. Kruus, and Brian J. Matt. Constraints and approaches for distributed sensor network security. Technical report, NAI Labs, The Security Research Division, Network Associates, Inc., 2000.

[19] Claude Castelluccia, Aldar C-F. Chan, Einar Mykletun, and Gene Tsudik. Efficient and provably secure aggregation of encrypted data in wireless sensor networks. *ACM Trans. Sen. Netw.*, 5(3):20:1–20:36, 2009.

[20] Seyit A. Çamtepe and Bülent Yener. Combinatorial design of key distribution mechanisms for wireless sensor networks. In Pierangela Samarati, Peter Ryan, Dieter Gollmann, and Refik Molva, editors, *Computer Security - ESORICS 2004*, volume 3193 of *Lecture Notes in Computer Science*, pages 293–308. Springer Berlin Heidelberg, 2004.

[21] Seyit A. Çamtepe and Bülent Yener. Combinatorial design of key distribution mechanisms for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 15(2):346–358, 2007.

[22] Aldar C-F. Chan and Claude Castelluccia. A security framework for privacy-preserving data aggregation in wireless sensor networks. *ACM Trans. Sen. Netw.*, 7(4):29:1–29:45, 2011.

[23] Haowen Chan and Adrian Perrig. Efficient security primitives derived from a secure aggregation algorithm. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, pages 521–534, New York, NY, USA, 2008. ACM.

[24] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, SP '03, pages 197–213, Washington, DC, USA, 2003. IEEE Computer Society.

[25] Haowen Chan, Adrian Perrig, and Dawn Song. Secure hierarchical in-network aggregation in sensor networks. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 278–287, New York, NY, USA, 2006. ACM.

[26] Rung-Ching Chen, Chia-Fen Hsieh, and Yung-Fa Huang. A new method for intrusion detection on hierarchical wireless sensor networks. In *Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication*, ICUIMC '09, pages 238–245, New York, NY, USA, 2009. ACM.

[27] Long Cheng, Chengdong Wu, Yunzhou Zhang, Hao Wu, Mengxin Li, and Carsten Maple. A survey of localization in wireless sensor network. *International Journal of Distributed Sensor Networks*, 2012:1–12, 2012.

[28] Charles J. Colbourn and Jeffrey H. Dinitz. *Handbook of Combinatorial Designs, Second Edition (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, 2006.

[29] Crossbow Technologies. Mica2 datasheet. http://pdf.datasheetarchive.com/indexerfiles/Datasheet-026/DSA00462855.pdf. Accessed 2015-04-21.

[30] Crossbow Technologies. Mica2Dot datasheet. http://pdf.datasheetarchive.com/indexerfiles/Datasheet-026/DSA00462856.pdf. Accessed 2015-04-21.

[31] Boris Danev and Srdjan Čapkun. Transient-based identification of wireless sensor nodes. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, IPSN '09, pages 25–36, Washington, DC, USA, 2009. IEEE Computer Society.

[32] Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly secure message transmission. *J. ACM*, 40(1):17–47, 1993.

[33] Dezun Dong, Mo Li, Yunhao Liu, Xiang-Yang Li, and Xiangke Liao. Topological detection on wormholes in wireless ad hoc and sensor networks. *IEEE/ACM Transactions on Networking*, 19(6):1787–1796, 2011.

[34] Junwu Dong, Dingyi Pei, and Xueli Wang. A key predistribution scheme based on 3-designs. In Dingyi Pei, Moti Yung, Dongdai Lin, and Chuankun Wu, editors, *Information Security and Cryptology*, volume 4990 of *Lecture Notes in Computer Science*, pages 81–92. Springer Berlin Heidelberg, 2008.

[35] John R. Douceur. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 251–260, London, UK, 2002. Springer-Verlag.

[36] Milica P. Durisic, Zhilbert Tafa, Goran Dimic, and Veljko Milutinovic. A survey of military applications of wireless sensor networks. In *Embedded Computing (MECO), 2012 Mediterranean Conference on*, pages 196–199, June 2012.

[37] Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 41–47, New York, NY, USA, 2002. ACM.

[38] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of the 5th*

Annual ACM/IEEE International Conference on Mobile Computing and Networking, MobiCom '99, pages 263–270, New York, NY, USA, 1999. ACM.

[39] Elena Fasolo, Michele Rossi, Jörg Widmer, and Michele Zorzi. In-network aggregation techniques for wireless sensor networks: A survey. *IEEE Wireless Commun.*, 14(2):70–87, 2007.

[40] Keith B. Frikken and Joseph A. Dougherty, IV. An efficient integrity-preserving scheme for hierarchical sensor aggregation. In *Proceedings of the First ACM Conference on Wireless Network Security*, WiSec '08, pages 68–76, New York, NY, USA, 2008. ACM.

[41] Joao Girao, Dirk Westhoff, and Markus Schneider. CDA: Concealed data aggregation for reverse multicast traffic in wireless sensor networks. In *Proceedings of the 2005 IEEE International Conference on Communications*, ICC '05, pages 3044–3049, 2005.

[42] Song Guo and Victor Leung. A compromise-resilient group rekeying scheme for hierarchical wireless sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, WCNC '10, pages 1–6, 2010.

[43] Wenbo He, Xue Liu, Hoang Nguyen, Klara Nahrstedt, and Tarek Abdelzaher. PDA: Privacy-preserving data aggregation in wireless sensor networks. In *Proceedings of the 26th IEEE International Conference on Computer Communications*, INFOCOM '07, pages 2045–2053, 2007.

[44] Wenbo He, Hoang Nguyen, Xue Liu, Klara Nahrstedt, and Tarek Abdelzaher. ipda: An integrity-protecting private data aggregation scheme for wireless sensor networks. In *Proceedings of the 2008 IEEE Military Communications Conference*, MILCOM '08, pages 1–7, 2008.

[45] Xiaobing He, Michael Niedermeier, and Hermann de Meer. Dynamic key management in wireless sensor networks: A survey. *Journal of Network and Computer Applications*, 36(2):611–622, 2013.

[46] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, HICSS '00, Washington, DC, USA, 2000. IEEE Computer Society.

[47] Kevin Henry, Maura B. Paterson, and Douglas R. Stinson. Practical approaches to varying network size in combinatorial key predistribution schemes. In Tanja Lange,

Kristin Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography – SAC 2013*, volume 8282 of *Lecture Notes in Computer Science*, pages 89–117. Springer Berlin Heidelberg, 2014.

[48] Kevin Henry and Douglas R. Stinson. Secure network discovery in wireless sensor networks using combinatorial key pre-distribution. In *2011 Workshop on Lightweight Security Privacy: Devices, Protocols and Applications – LightSec*, pages 34–43, 2011.

[49] Lingxuan Hu and David Evans. Secure aggregation for wireless networks. In *Proceedings of the 2003 Symposium on Applications and the Internet Workshops*, SAINT-W '03, Washington, DC, USA, 2003. IEEE Computer Society.

[50] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Packet leashes: A defense against wormhole attacks in wireless networks. In *Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, INFOCOM '03, pages 1976–1986, 2003.

[51] Chalermek Intanagonwiwat, Deborah Estrin, Ramesh Govindan, and John Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, ICDCS '02, pages 457–458, Washington, DC, USA, 2002. IEEE Computer Society.

[52] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom '00, pages 56–67, New York, NY, USA, 2000. ACM.

[53] Imad Jawhar and Nader Mohamed. A hierarchical and topological classification of linear sensor networks. In *Proceedings of the 2009 Conference on Wireless Telecommunications Symposium*, WTS '09, pages 72–79, Piscataway, NJ, USA, 2009. IEEE Press.

[54] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *Proceedings of the 2003 IEEE International Workshop on Sensor Network Protocols and Applications*, pages 113–127, 2003.

[55] Ed Kendall, Michelle Kendall, and Wilfrid S. Kendall. A generalised formula for calculating the resilience of random key predistribution schemes. Cryptology ePrint Archive, Report 2012/426, 2012.

[56] Samir Khuller, Balaji Raghavachari, and Neal Young. Balancing minimum spanning trees and shortest-path trees. *Algorithmica*, 14(4):305–321, 1995.

[57] David A. Knox and Thomas Kunz. Rf fingerprints for secure authentication in single-hop wsn. In *IEEE International Conference on Wireless and Mobile Computing*, WIMOB '08, pages 567–573, Oct 2008.

[58] Dileep Kumar and Shirshu Varma. An efficient localization based on directional antenna for wireless sensor networks (WSN's). *International Journal of Computer and Electrical Engineering*, 1(5):542–549, 2009.

[59] Jooyoung Lee and Douglas R. Stinson. Deterministic key predistribution schemes for distributed sensor networks. In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography*, volume 3357 of *Lecture Notes in Computer Science*, pages 294–307. Springer Berlin Heidelberg, 2005.

[60] Jooyoung Lee and Douglas R. Stinson. Common intersection designs. *Journal of Combinatorial Designs*, 14(4):251–269, 2006.

[61] Jooyoung Lee and Douglas R. Stinson. On the construction of practical key predistribution schemes for distributed sensor networks using combinatorial designs. *ACM Trans. Inf. Syst. Secur.*, 11(2):5:1–5:35, 2008.

[62] Yoonmyung Lee, Gyouho Kim, Suyoung Bang, Yejoong Kim, Inhee Lee, Prabal Dutta, Dennis. Sylvester, and David Blaauw. A modular 1mm3 die-stacked sensing platform with optical communication and multi-modal energy harvesting. In *2012 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, ISSCC '12, pages 402–404, 2012.

[63] Philip Levis, Sam Madden, Joseph Polastre, Robert Szewczyk, Kamin Whitehouse, Alec Woo, David Gay, Jason Hill, Matt Welsh, Eric Brewer, and David Culler. TinyOS: An operating system for sensor networks. In Werner Weber, Jan M. Rabaey, and Emile Aarts, editors, *Ambient Intelligence*, pages 115–148. Springer Berlin Heidelberg, 2005.

[64] Zhijun Li and Guang Gong. Data aggregation integrity based on homomorphic primitives in sensor networks. In *Proceedings of the 9th International Conference on Ad-hoc, Mobile and Wireless Networks*, ADHOC-NOW '10, pages 149–162, Berlin, Heidelberg, 2010. Springer-Verlag.

[65] Zhijun Li and Guang Gong. DHT-based detection of node clone in wireless sensor networks. In Jun Zheng, Shiwen Mao, Scott F. Midkiff, and Hua Zhu, editors, *Ad Hoc Networks*, volume 28 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 240–255. Springer Berlin Heidelberg, 2010.

[66] Libelium Comunicaciones Distribuidas S.L. Waspmote sensor overview. `http://www.libelium.com/products/waspmote/sensors/`. Accessed 2015-04-20.

[67] Hui Ling and Taieb Znati. End-to-end pairwise key establishment using node disjoint secure paths in wireless sensor networks. *Int. J. Secur. Netw.*, 2(1/2):109–121, 2007.

[68] An Liu and Peng Ning. TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. In *2008 International Conference on Information Processing in Sensor Networks*, IPSN '08, pages 245–256, 2008.

[69] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TAG: A tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):131–146, 2002.

[70] David J. Malan, Matt Welsh, and Michael D. Smith. A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. In *Proceedings of the First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, SECON '04, pages 71–80, 2004.

[71] Mark Manulis and Jörg Schwenk. Security model and framework for information aggregation in sensor networks. *ACM Trans. Sen. Netw.*, 5(2):13:1–13:28, 2009.

[72] Keith M. Martin. The combinatorics of cryptographic key establishment. In Anthony Hilton and John Talbot, editors, *Surveys in Combinatorics 2007*, pages 223–274. Cambridge University Press, 2007. Cambridge Books Online.

[73] Keith M. Martin. On the applicability of combinatorial designs to key predistribution for wireless sensor networks. In Yeow Meng Chee, Chao Li, San Ling, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology*, volume 5557 of *Lecture Notes in Computer Science*, pages 124–145. Springer Berlin Heidelberg, 2009.

[74] Keith M. Martin and Maura B. Paterson. An application-oriented framework for wireless sensor network key establishment. *Electron. Notes Theor. Comput. Sci.*, 192(2):31–41, 2008.

[75] Keith M. Martin and Maura B. Paterson. Ultra-lightweight key predistribution in wireless sensor networks for monitoring linear infrastructure. In Olivier Markowitch, Angelos Bilas, Jaap-Henk Hoepman, Chris J. Mitchell, and Jean-Jacques Quisquater, editors, *Information Security Theory and Practice: Smart Devices, Pervasive Systems, and Ubiquitous Networks*, WISTP '09, pages 143–152, Berlin, Heidelberg, 2009. Springer-Verlag.

[76] Keith M. Martin, Maura B. Paterson, and Douglas R. Stinson. Key predistribution for homogeneous wireless sensor networks with group deployment of nodes. *ACM Trans. Sen. Netw.*, 7(2):11:1–11:27, 2010.

[77] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography.* CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[78] Suman Nath, Phillip B. Gibbons, Srinivasan Seshan, and Zachary R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 250–262, New York, NY, USA, 2004. ACM.

[79] Martin Nilsson. SPIDA: A direction-finding antenna for wireless sensor networks. In Pedro J. Marron, Thiemo Voigt, Peter Corke, and Luca Mottola, editors, *Real-World Wireless Sensor Networks*, volume 6511 of *Lecture Notes in Computer Science*, pages 138–145. Springer Berlin Heidelberg, 2010.

[80] Leonardo B. Oliveira, Michael Scott, Julio Lopez, and Richard Dahab. TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. In *Proceedings of the 5th International Conference on Networked Sensing Systems*, INSS '08, pages 173–180, 2008.

[81] Suat Ozdemir. Concealed data aggregation in heterogeneous sensor networks using privacy homomorphism. In *Proceedings of the IEEE International Conference on Pervasive Services*, pages 165–168, 2007.

[82] Suat Ozdemir, Miao Peng, and Yang Xiao. PRDA: polynomial regression-based privacy-preserving data aggregation for wireless sensor networks. *Wireless Communications and Mobile Computing*, 15:615–628, 2013.

[83] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure on-demand distance vector routing in ad hoc networks. In *Proceedings of the 2005 IEEE/Sarnoff Symposium on Advances in Wired and Wireless Communication*, pages 168–171, 2005.

[84] Maura B. Paterson and Douglas R. Stinson. A unified approach to combinatorial key predistribution schemes for sensor networks. *Designs, Codes and Cryptography*, 71(3):433–457, 2014.

[85] Mathew D. Penrose and Andrew R. Wade. On the total length of the random minimal directed spanning tree. *Adv. in Appl. Probab.*, 38(2):336–372, 2006.

[86] Adrian Perrig, Robert Szewczyk, J.D. Tygar, Victor Wen, and David E. Culler. SPINS: Security protocols for sensor networks. *Wirel. Netw.*, 8(5):521–534, 2002.

[87] Steffen Peter, Dirk Westhoff, and Claude Castelluccia. A survey on the encryption of convergecast traffic with in-network processing. *IEEE Transactions on Dependable and Secure Computing*, 7(1):20–34, 2010.

[88] Marcin Poturalski, Panos Papadimitratos, and Jean-Pierre Hubaux. Secure neighbor discovery in wireless networks: Formal investigation of possibility. In *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security*, ASIACCS '08, pages 189–200, New York, NY, USA, 2008. ACM.

[89] Bartosz Przydatek, Dawn Song, and Adrian Perrig. SIA: Secure information aggregation in sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, pages 255–265, New York, NY, USA, 2003. ACM.

[90] Ramesh Rajagopalan and Pramod K. Varshney. Data-aggregation techniques in sensor networks: A survey. *Communications Surveys Tutorials, IEEE*, 8(4):48–63, 2006.

[91] Kasper Bonne Rasmussen and Srdjan Čapkun. Implications of radio fingerprinting on the security of sensor networks. In *Proceedings of the Third International Conference on Security and Privacy in Communications Networks and the Workshops*, SecureComm '07, pages 331–340, 2007.

[92] Kasper Bonne Rasmussen and Srdjan Čapkun. Realization of RF distance bounding. In *Proceedings of the 19th USENIX Conference on Security*, USENIX '10, pages 25–37, Berkeley, CA, USA, 2010. USENIX Association.

[93] Wei Ren, Yi Ren, and Hui Zhang. H2S: A secure and efficient data aggregative retrieval scheme in unattended wireless sensor networks. In *Proceedings of the Fifth International Conference on Information Assurance and Security*, volume 2 of *IAS '09*, pages 450–453, 2009.

[94] Gabriel Robins and Jeffrey S. Salowe. Low-degree minimum spanning trees. *Discrete & Computational Geometry*, 14(1):151–165, 1995.

[95] Sankardas Roy, Mauro Conti, Sanjeev Setia, and Sushil Jajodia. Secure data aggregation in wireless sensor networks: Filtering out the attacker's impact. *IEEE Transactions on Information Forensics and Security*, 9(4):681–694, 2014.

[96] Sankardas Roy, Sanjeev Setia, and Sushil Jajodia. Attack-resilient hierarchical data aggregation in sensor networks. In *Proceedings of the Fourth ACM Workshop on Security of Ad Hoc and Sensor Networks*, SASN '06, pages 71–82, New York, NY, USA, 2006. ACM.

[97] Kimaya Sanzgiri, Bridget Dahill, Brian N. Levine, Clay Shields, and Elizabeth M. Belding-Royer. A secure routing protocol for ad hoc networks. In *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*, pages 78–87, 2002.

[98] Elaine Shi and Adrian Perrig. Designing secure sensor networks. *Wireless Communications, IEEE*, 11(6):38–43, 2004.

[99] Eliana Stavrou and Andreas Pitsillides. A survey on secure multipath routing protocols in WSNs. *Comput. Netw.*, 54(13):2215–2238, 2010.

[100] J. Michael Steele, Lawrence A. Shepp, and William F. Eddy. On the number of leaves of a euclidean minimal spanning tree. *Journal of Applied Probability*, 24(4):809–826, 1987.

[101] Douglas R. Stinson. *Combinatorial Designs: Constructions and Analysis*. Springer-Verlag New York, 2004.

[102] Martin Vloet. M3 with pressure sensor. https://www.flickr.com/photos/26556146@N07/16254669028/, 2014. Accessed 2015-04-20.

[103] David Wagner. Resilient aggregation in sensor networks. In *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks*, SASN '04, pages 78–87, New York, NY, USA, 2004. ACM.

[104] Haodong Wang and Qun Li. Efficient implementation of public key cryptosystems on mote sensors (short paper). In *Proceedings of the 8th International Conference on Information and Communications Security*, ICICS '06, pages 519–528, Berlin, Heidelberg, 2006. Springer-Verlag.

[105] Yong Wang, Garhan Attebury, and Byrav Ramamurthy. A survey of security issues in wireless sensor networks. *Communications Surveys Tutorials, IEEE*, 8(2):2–23, 2006.

[106] Yongge Wang and Yvo Desmedt. Perfectly secure message transmission revisited. *IEEE Transactions on Information Theory*, 54(6):2582–2595, 2008.

[107] Brett A. Warneke, Michael D. Scott, Brian S. Leibowitz, Lixia Zhou, Colby L. Bellew, J. Alex Chediak, Joseph M. Kahn, Bernhard E. Boser, and Kristofer S.J. Pister. An autonomous 16 mm3 solar-powered node for distributed wireless sensor networks. In *Proceedings of IEEE Sensors*, volume 2, pages 1510–1515, 2002.

[108] Jiang Wu and Doug R. Stinson. Three improved algorithms for multipath key establishment in sensor networks using protocols for secure message transmission. *IEEE Trans. Dependable Secur. Comput.*, 8(6):929–937, 2011.

[109] Li Xu, Jianwei Chen, and Xiaoding Wang. Cover-free family based efficient group key management strategy in wireless sensor network. *Journal of Communications*, 3(6):51–58, 2008.

[110] Geng Yang, Sen Li, Xiaolong Xu, Hua Dai, and Zhen Yang. Precision-enhanced and encryption-mixed privacy-preserving data aggregation in wireless sensor networks. *Intl. Journal of Distributed Sensor Networks*, 2013:1–12, 2013.

[111] Yi Yang, Xinran Wang, Sencun Zhu, and Guohong Cao. SDAP: A secure hop-by-hop data aggregation protocol for sensor networks. In *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '06, pages 356–367, New York, NY, USA, 2006. ACM.

[112] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Comput. Netw.*, 52(12):2292–2330, 2008.

[113] Haifeng Yu. Secure and highly-available aggregation queries in large-scale sensor networks via set sampling. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, IPSN '09, pages 1–12, Washington, DC, USA, 2009. IEEE Computer Society.

[114] Manel Guerrero Zapata and N. Asokan. Securing ad hoc routing protocols. In *Proceedings of the 1st ACM Workshop on Wireless Security*, WiSE '02, pages 1–10, New York, NY, USA, 2002. ACM.