Combinatorial Aspects of Key Predistribution for Resource-Constrained Networks

Michelle Louise Kendall

Thesis submitted to the University of London
for the degree of Doctor of Philosophy

Information Security Group
School of Mathematics and Information Security
Royal Holloway, University of London

2013

Declaration of Authorship

I, Michelle Louise Kendall, hereby declare that this thesis and the work presented in it is entirely my own. Where I have consulted the work of others, this is always clearly stated.

Signed: _____

Date: _____

Acknowledgements

To my patient supervisor, Keith, without whom I would not have reached this point; for agreeing to supervise me, for teaching me the ways of research and academic writing, and for keeping me motivated throughout;

To my co-authors, Keith, Siaw-Lynn, Maura, Doug, Ed and Wilfrid, for inspiring conversations and wise guidance (and a finite Erdös number!);

To all my colleagues at Royal Holloway who have supported, taught and encouraged me: to Shahram for the chocolate, to Liz and Gaven for the lifts and pancakes, to Laurence for the maths puzzles and teaching advice, to Stephanie for the conversation about expander graphs, to Tamas and Andrew for surviving the typos of The Probabilistic Method together, and to all my office mates;

To my fantastic friends and family for providing me with the love and support necessary to complete this thesis, and in particular to my brilliant housemate Liz for keeping me sane, entertained and well-fed, to Wilfrid for all the invaluable advice, and to Mum for the proof-reading and constant encouragement;

And to my wonderful husband, Ed, for putting up with me through the PhD years, for competing to get the first doctorate, for all the cups of tea and "there there"s, and for your continued belief in me;

Thank you.

Abstract

To secure a network of small devices using symmetric key cryptography is a non-trivial task. Nevertheless, it is important because public key cryptography is computationally expensive and therefore infeasible to implement on some small, battery-powered devices with limited memory. We study methods for allocating symmetric keys to devices before deployment, known as *key predistribution schemes*. Using combinatorial techniques, we analyse and design a variety of key predistribution schemes.

We provide a correction to the previously stated formula for calculating the resilience of certain random key predistribution schemes, presenting instead a rigorously proved and widely applicable formula. We also present a simplified formula for calculating the connectivity.

Next, we examine the role of expander graphs in key predistribution schemes. We demonstrate that good expansion is desirable for robust schemes, and discuss how this can be achieved. In particular, we examine the expansion of key predistribution schemes built from expander graph constructions, which provide perfect resilience. We show that if perfect resilience is not required, key predistribution schemes with higher connectivity and expansion can be created from hypergraphs and designs, and we explore the relationships between these constructions. We argue for the use of hypergraphs to represent and analyse key predistribution schemes, and identify open problems which, if solved, could lead to further suitable and robust constructions for key predistribution schemes.

Finally, we study a class of schemes which we call 'broadcast-enhanced key

predistribution schemes'. These are schemes which make use of a trusted base station and a broadcast channel to update and revoke keys in a network whilst it is operational. We explore the range of benefits which such schemes can provide, and present and analyse two constructions for particular scenarios: a scheme which allows efficient revocation of devices, and a scheme which creates hierarchy amongst the devices for efficient routing and battery consumption. We demonstrate that our schemes provide effective and flexible trade-offs between the conflicting parameters of connectivity, resilience, key storage and broadcast load.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many questions in information security can be approached from a combinatorial perspective. In this thesis we demonstrate combinatorial, and in particular graph-theoretical, approaches to the construction and analysis of key predistribution schemes for networks. We use graph theory to suggest new approaches for the construction of key predistribution schemes, and to draw links between existing approaches. We also use combinatorial methods to simplify expressions and proofs of existing results, and we give details of certain claims from the literature which have not been rigorously proven, before providing the corrected statements and formulae.

Our analysis covers the calculation of resilience in random key predistribution schemes, the role of expander graphs in key predistribution schemes, designs and hypergraphs with good expansion, and finally a class of schemes which unite ideas from broadcast encryption and key predistribution schemes. We now give an overview of each of the chapters which follow.

1.1 Chapter overviews

In Chapter 2 we outline the cryptographic principles which form the foundations of all our key predistribution scheme scenarios. Next, we formally define key predistribution schemes and the relevant combinatorial tools for their construction and analysis. We then provide a brief overview of the literature on key predistribution schemes, drawing distinctions between deterministic and random key predistribution schemes, and explaining their connection with key establishment schemes.

To analyse and compare key predistribution schemes it is helpful to calculate their resilience, which is a measure of the proportion of the network compromised by an adversary which has learned keys from a small number of nodes. In Chapter 3 we provide a single formula to express the resilience of a wide range of random key predistribution schemes. We give details of two previous statements of this formula: that of Chan et al. [20], which makes an incorrect assumption of probabilistic independence, and the formula given by Yum and Lee [75], which is difficult to compute. The chapter also includes a proof of the commonly-stated formula for the resilience of Eschenauer and Gligor's seminal random key predistribution scheme [32], and a simplified expression for the connectivity of the q-composite scheme from [20].

It is common to represent both key predistribution schemes and networks with graphs. Chapters 4 and 5 are concerned with graph theory in key predistribution schemes, and in particular the concept of expansion in graphs.

A graph which represents the key sharing in a network is known as the key graph, and a graph which represents the relative locations of distributed devices is known as the communication graph. Vertices represent devices or 'nodes', edges in the key graph represent shared keys, and edges in the communication graph correspond to the pairs of nodes which are within wireless communication range. A pair of devices share an edge in the intersection of these two graphs exactly when they are within communication range and share common key(s).

In Chapter 4 we study the use of graph-theoretical tools for the analysis of key predistribution schemes, and in particular consider the role of expander graphs in their construction and analysis. We critique the suggestions of Ghosh in [36], demonstrating that his claim of good expansion being desirable in the product graph is unsubstantiated. We provide a simple example which demonstrates that good expansion in the product graph can be achieved even when the intersection graph is worst-possible.

Instead, we identify that good expansion in the intersection graph is desirable for well connected, robust networks, and refer to two key predistribution schemes [17, 60] which are based on expander graph constructions. These provide perfect resilience but at the expense of lower connectivity than many other key predistribution schemes with comparable key storage. In particular, we show that random key predistribution schemes and many of the combinatorial designs which have been suggested for use as KPSs have good expansion. This is a previously unstated advantage of using these constructions for KPSs.

In Chapter 5 we argue for the use of hypergraphs to represent and construct key predistribution schemes. We show that a hypergraph representation has benefits over the previously-used graph representations because it clearly demonstrates the key storage and resilience, as well as the connectivity. Developing ideas from Chapter 4, we propose that wherever perfect resilience is not required, higher connectivity can be achieved through the use of expander hypergraphs to construct key predistribution schemes. We present a simple example of a key predistribution scheme based on a Cayley hypergraph and show that, whilst far from optimal amongst expanding hypergraph constructions, it achieves our aims of increasing connectivity and slightly lowering resilience, whilst maintaining low key storage and good expansion. We argue that further research into random uniform hypergraph constructions and random strongly regular graphs would be likely to provide further robust key predistribution scheme constructions.

For network environments where a trusted base station and broadcast channel are available, we propose a category of schemes called *broadcast-enhanced key predistribution schemes* (BEKPSs) which utilise the extra resources to improve upon standard key predistribution schemes. In Chapter 6 we provide a simplification of some of the proofs from the scheme of Cichoń et al. [23], which we classify as a BEKPS. We then study two particular benefits which BEKPSs can provide over key predistribution schemes where a base station and broadcast channel are not available, namely the ease of revocation and the possibility to create a dynamic hierarchy amongst the devices. We propose families of BEKPSs which are suitable for each of these scenarios, and our analysis demonstrates that they are effective in their aims, whilst providing

practical and flexible trade-offs between connectivity, resilience and broadcast load.

We conclude in Chapter 7 with a summary of our work, and propose further questions for future consideration.

1.2 Publications

The research in Chapter 3 is joint work with Ed Kendall, Wilfrid S. Kendall and Keith M. Martin, and appears as a paper on the Cryptology ePrint Archive [41].

Chapter 4 is largely based on the paper 'On the role of expander graphs in key predistribution schemes for wireless sensor networks' [42] with Keith M. Martin, which was presented at *WEWoRC 2011*.

Finally, Chapter 6 is joint work with Keith M. Martin, Siaw-Lynn Ng, Maura B. Paterson and Douglas R. Stinson and also appears on the Cryptology ePrint Archive [43].

Chapter 2

Preliminaries

Contents

In this chapter we present the core principles, definitions and notation on which the subsequent chapters rely. We begin with an introduction to some of

the fundamental principles of cryptography and information security in Section 2.1. In Section 2.2 we define key predistribution schemes and explain the motivation for their study. Next, in Section 2.3, we give an introduction to the relevant mathematical tools used in the construction of key predistribution schemes, namely combinatorial designs and graphs. In particular, Section 2.3.2 demonstrates how graphs can be used to represent key predistribution schemes. Finally, in Section 2.4 we present a brief review of the literature on deterministic and random key predistribution schemes, and explain the connection to a related concept, key establishment schemes.

2.1 Principles of cryptography and information security

2.1.1 Cryptography for secrecy and authentication

The principle of *encrypting* information to provide secrecy is reasonably familiar: we are surrounded by scenarios where information needs to be stored or transmitted with restrictions on who can access or read it. Encryption provides a method for ensuring (or, in many cases, ensuring with high probability) that only an intended recipient is able to decrypt and view the original data. Encryption and decryption algorithms generally require at least one cryptographic *key*; we will provide further details in Section 2.1.2.

Another important use of cryptographic keys is to provide forms of *authentication*. For our purposes, it suffices to say that cryptographic keys can be used to provide *entity authentication*, an assurance that the message originated

from a specific person or device, and *data authentication* or *data integrity*, an assurance that the message received is identical to the message sent, and has not been altered in any way. For more details and examples, see [51].

We note that, in the variety of scenarios which we consider, many will have no particular need for secrecy. For example, if we consider a network of devices measuring temperature over an area of land, this data may not be confidential, and indeed, could easily be obtained by anyone visiting the area. However, it may well be important for the devices to use cryptography when sharing their measurements in order to provide assurance that each data point really did originate from the device claiming to have sent it, and that the measurement has not been altered during transmission. Thus, we note that there is a variety of reasons why it may be necessary for devices to store cryptographic keys. In the analysis which follows we will not be concerned with the purpose of the keys, or the particular algorithms and protocols in which they will be used, but simply the question of how to distribute the keys to the devices.

2.1.2 Symmetric and asymmetric keys

Before the 1970s, cryptographic keys were *symmetric*, that is, the same key was used to encrypt and decrypt the data. In 1973, James Ellis, Clifford Cocks and Malcolm Williamson developed an *asymmetric* algorithm, whereby the keys needed for encryption and decryption were different. Their work at GCHQ was not publicised, but similar ideas were developed independently by Whitfield Diffie and Martin Hellman, who in 1976 proposed the idea of an asymmetric cryptosystem [28] and Ron Rivest, Adi Shamir and Leonard

Adleman who proposed the RSA cryptosystem in 1977 [59]. We refer the interested reader to [61, 63] for further details.

Asymmetric schemes are also known as *public key* algorithms because the encryption key can be made public. That is, they rely on the idea that it is computationally infeasible to derive the decryption key from the encryption key, as long as a particular computational problem (such as factoring or the discrete logarithm problem) is computationally infeasible. These ideas revolutionised cryptography as they enabled entities to send and receive encrypted data without having previously agreed a symmetric key. In particular, public key cryptography is ideal for exchanging encrypted messages between entities which have no pre-existing relationship. However, public key algorithms are currently more computationally expensive than symmetric key algorithms. There remain applications and devices where public key cryptography is infeasible, and pre-agreeing a symmetric key is still necessary. It is on such scenarios that we focus in this thesis.

2.1.3 Adversary models

When considering the strength of a cryptographic algorithm or protocol, it is important to consider the type of adversary against which one wishes to be secure. Notice that no cryptographic protocol between users A and B is secure if A or B tells the adversary all of the keys being used. Similarly, many cryptographic algorithms would be insecure against an adversary with infinite time and/or computing power at its disposal. It is therefore important to specify exactly which threats one is seeking to protect against. We gen-

erally make worst-case assumptions, so as to provide an upper bound on the damage caused by an adversary. We will give our detailed adversary model in Section 2.2.4.

2.1.4 Key management

Key management is an important part of any system which uses cryptography, and can easily be a weak link in an otherwise well-designed cryptographic system. Areas of key management include:

- **Key generation:** the production of 'good' keys, that is, keys which ideally do not conform to any pattern which an adversary could exploit;
- **Key distribution:** the allocation of keys to devices;
- **Key refreshing/update:** replacing keys - it may be desirable to define a 'lifetime' for keys, that is, a window of time during which they may be used, and after which they should be changed;
- **Key revocation:** removing a key from use - if a key becomes known to an adversary, we ideally want to be able to stop that key being used for further communication.

Our focus will be mainly on key distribution, in particular key *predistribution*, as defined in Section 2.2.1. In Chapter 6 we will also discuss how revocation and updates can be achieved in certain scenarios, and present efficient ways of doing so.

2.2 Key predistribution schemes

2.2.1 Definition and applications

We consider the distribution of keys to networks of small, resource-constrained devices, or 'nodes'. A wireless sensor network (WSN) is an example of such a network. It is a collection of static, small, battery powered devices called sensor nodes, which communicate with each other wirelessly. The resulting network is usually used for monitoring an environment by gathering local data such as temperature, light or motion. Much of the literature on key predistribution schemes is concerned with wireless (or distributed) sensor networks, including at least a third of the papers cited in this thesis. However, we will be considering schemes which are applicable to any distributed, stationary network of homogeneous, resource-constrained nodes. As we assume that the nodes are lightweight and battery powered, it is important to consider battery conservation in order to allow the network to remain effective for the appropriate period of time, and to ensure that the storage required of the nodes is not beyond their memory capacity.

Resource-constrained networks can be deployed in a range of different environments, including potentially hostile areas such as military or volcanic zones [71], where it would be dangerous or impractical to carry out the monitoring or data gathering by hand. In hostile environments it may be necessary to encrypt messages for security and/or authentication. Various cryptographic key management schemes have been proposed for such scenarios. In some cases there is an online key server or base station to distribute keys to the nodes

whenever necessary; if not, *key predistribution schemes* are required.

A *key predistribution scheme* (KPS) is a method for allocating keys to the nodes of a network before they are deployed into their chosen environment. We consider KPSs which assign symmetric keys, since small sensor nodes are resource-constrained with low storage, communication and computational abilities, and are often unable to support asymmetric cryptography. A major drawback of KPSs is that once the keys have been predistributed, subsequent key management operations are challenging to conduct [4]. We will present examples of KPSs in Section 2.4. In order to make best use of the nodes' limited resources, it is usually desirable to minimise the *key storage* requirement whilst maximising the *connectivity* and *resilience* of a network. We define these concepts more precisely in Section 2.2.3.

2.2.2 Framework

We identify four aspects of key predistribution which together form a framework in which to categorise and study KPSs. Whilst many papers in the KPS literature include specifications and/or analysis for each of these stages, others only briefly mention or omit altogether the details of key generation, shared key discovery and network alterations, focusing purely on the key predistribution aspect.

Key generation Before the nodes are deployed, an entity which we call the *trusted base station* must create a set of keys. Specifically, a *key pool* $\mathcal{K}$ of

n symmetric keys $\{K_1, K_2, \ldots, K_n\}$ is selected from the space of all possible keys.

Key predistribution The trusted base station allocates to each node a subset of keys from the key pool. The size of the key pool and the number of keys allocated to each node are chosen to provide a trade-off between the conflicting metrics of key storage, connectivity and resilience, as defined in Section 2.2.3.

Shared key discovery Once the nodes have been deployed, in order for them to begin secure communication, a shared key discovery protocol such as one of those given in [17, 73] should be implemented. This ensures that each node determines the set of other nodes with which it shares keys. If the keys are assigned in a way known to all the nodes, then a node N_i can broadcast information about its identity, its *node identifier*, from which any node N_j can derive the list of key identifiers which correspond to N_i's key set. It then remains for each node to look up whether any of these keys is also known to them. If keys are not assigned in a deterministic or publicly known way, then each node has to broadcast its whole list of key identifiers in order to perform shared key discovery.

Network alterations Some implementations of KPSs will include the capacity for a network to make alterations after the initial shared key discovery. Such alterations can include the revoking of keys or nodes, the establishing of

new keys between nodes, and the updating of old keys. These can be effected by node voting systems [20, 71] and techniques discussed in Section 2.4.4. If a trusted base station is able to broadcast instructions to a network as in a BEKPS (Chapter 6), then a wide range of update and revocation protocols is possible.

2.2.3 Metrics

The metrics typically used to analyse key predistribution schemes are:

Key storage: the number of keys which each node is required to store. Unless otherwise stated, the key storage will be constant and denoted by k.

Connectivity: the proportion of nodes which are 'connected' by sharing keys. Connectivity can be measured or estimated both *globally* and *locally* [30, 53]. We will refer again to global connectivity in Section 4.6 but in general we will use the measure of local connectivity Pr_1, which is the probability that a randomly-chosen pair of nodes share at least $q \geq 1$ keys, where q is an intersection threshold dictated by the KPS. Many KPSs only require nodes to have a single key in common in order to be connected, i.e. $q = 1$. Where two nodes share q or more keys, some protocols dictate that they should use a combination of those keys, such as a hash, to encrypt their communications.

Resilience: a measure of the network's susceptibility to compromise by an adversary. (We define our adversary precisely in Section 2.2.4.) To measure the resilience we use the parameter fail_s, which is defined to be the probability that a randomly-chosen link between a pair of uncompromised nodes is broken after the adversary has compromised s nodes. By 'broken', we mean that the key or keys securing that link are all known to the adversary. Equivalently, fail_s measures the fraction of compromised links between uncompromised nodes throughout the network, after an adversary has compromised s nodes. Notice that this is a conditional probability, conditioning on the two nodes in question being connected.

To the best of our knowledge, the notation 'Pr_1' and 'fail_s' were first used in [53] and [45] respectively. They are common measures, and many papers such as [20, 30] calculate them in the same way but with different notation.

If $\mathsf{fail}_s = 0$ for all $1 \leq s \leq v - 2$ (where v is the total number of nodes) then it is said that the network has *perfect resilience*. We note that lower values of fail_s represent better resilience, and that fail_s is not defined for $s > v - 2$.

To illustrate the trade-offs required between these three parameters, we consider some trivial examples of KPSs.

Example 2.1. *Every node is assigned the same single key K.*

This would require minimal key storage and ensure that any pair of nodes could communicate securely, so $\mathsf{Pr}_1 = 1$ for all pairs of nodes. However, there would be minimal resilience against an adversary, as the compromise of a single node

would reveal the key K, rendering all other links insecure. Formally, $\mathsf{fail}_s = 1$ *for all* $1 \leq s \leq v - 2$.

Example 2.2. *A unique key K_{ij} is assigned to every pair of nodes N_i, N_j.*

That is, for all $1 \leq i, j \leq v$, nodes N_i and N_j are both preloaded with a key K_{ij}, with the condition that $K_{ij} \neq K_{lm}$ for all pairs $(l, m) \neq (i, j)$, $1 \leq l, m \leq v$. This is called the complete pairwise *KPS. Such a KPS has perfect resilience and maximum connectivity, as* $\mathsf{Pr}_1 = 1$ *for all pairs of nodes. However, each node has to store $v - 1$ keys, which is infeasible when v is large.*

Example 2.3. *Every node N_i is assigned a unique single key K_i.*

This example is purely illustrative, since although it provides minimal key storage and perfect resilience, it is an ineffective KPS as there is no connectivity: $\mathsf{Pr}_1 = 0$ *for all pairs of nodes.*

We see, then, that it is trivial to optimise any two of the three parameters: key storage, connectivity and resilience. However, in most applications the above examples are inappropriate, thus we need to consider KPSs which provide trade-offs between all three of these metrics.

2.2.4 Network adversary model

We will assume that if an adversary has compromised a device and learned at least one of the keys which it stores, then the adversary knows *all* of the keys which it stores. This provides a worst-case analysis of the number of keys known to the adversary.

We model our network adversary by assuming that nodes are compromised at random. It is of course possible in practice that an adversary could employ a better strategy. For example, the adversary could target two nodes which appear not to be communicating with each other, in the hope of learning the maximum $2k$ keys. (If the nodes were communicating then they must share at least q keys, and so the adversary would learn at most $2k - q$ keys by their compromise.) The random adversary model can therefore be thought of as calculating a lower bound on fail_s, hence an upper bound on the resilience, and is useful as a metric for comparison of KPSs.

We note the distinction between 'passive' and 'active' adversaries, denoted in [29] as 'listening' and 'disrupting' adversaries, respectively. It is usually assumed that a listening adversary can intercept any message sent through the network, but can only decrypt a message if he knows all of the keys used to encrypt it. Thus, if an adversary knows a set of keys $\mathcal{K}_A \subset \mathcal{K}$, any message sent through the network which is encrypted by a subset of keys from $\mathcal{K}_A$ can be intercepted and decrypted by the adversary, regardless of whether or not the message is routed through compromised nodes.

A disrupting adversary is one which can alter network transmissions. Defensive measures can be taken against a disrupting adversary, for example in [29], messages are transmitted in such a way that alterations can be identified, and the correct message recovered (up to a threshold number of alterations). This is called *fault tolerance*, and is useful even in the absence of an adversary to cope with communication errors and node malfunctions. However, defending a network against a disrupting adversary is outside of our scope in this thesis:

we are concerned with efficient methods for distributing keys to devices, rather than the communication protocols to be used afterwards.

In summary, we make the following assumptions about our adversary:

- on compromising a node, the adversary learns all of its stored keys;
- the adversary compromises nodes at random;
- the adversary can intercept all messages sent through the network;
- the adversary can decrypt a message if and only if he knows the key(s) used to encrypt that message.

2.3 Combinatorics

2.3.1 Graphs

We now briefly introduce some graph-theoretic terminology and definitions, collated from [15, 21, 37]. In Section 2.3.2 we explain how graphs are used to represent and analyse KPSs, and we develop the graph-theoretic understanding of KPSs in Chapters 4 and 5.

Definition 2.1. A *graph* $G = (V, E)$ is a set of *vertices* $V = \{x_1, \ldots, x_v\}$ and a set of *edges* $E \subseteq V \times V$. We use the notation $(x_i, x_j) \in E$ to express that there is an edge between the vertices x_i and x_j, and we say that the edge (x_i, x_j) is *incident* to its endpoints x_i and x_j. Wherever an edge (x_i, x_j) exists, x_i and x_j are said to be *adjacent*.

All graphs considered in this thesis will be *simple graphs*, that is, they are *unweighted*, *undirected* and do not contain *self-loops* or *multiple edges*. These terms respectively mean that we do not assign different weights to vertices or edges, edges are not directed from one vertex to the other, there are no edges from a node to itself, and there is at most one edge between any two vertices.

Given subsets of vertices $X, Y \subset V$, the set of edges which connect X and Y is denoted

$$E(X,Y) = \{(x,y) : x \in X, y \in Y \text{ and } (x,y) \in E\} \ .$$

The *complement* $\overline{X}$ of X is the vertices which are not in X, that is, $\overline{X} = V \setminus X$.

An ordered set of consecutive edges $\{(x_{i1}, x_{i2}), (x_{i2}, x_{i3}), \ldots, (x_{i(p-1)}, x_{ip})\}$ in which all the vertices $x_{i1}, x_{i2}, \ldots, x_{ip}$ are distinct is called a *path* of length $p-1$. A *cycle* is a 'closed' path which begins and ends at the same vertex, i.e. a cycle is a path $\{(x_{i1}, x_{i2}), (x_{i2}, x_{i3}), \ldots, (x_{i(p-1)}, x_{ip})\}$ where $x_{i1}, x_{i2}, \ldots, x_{i(p-1)}$ are distinct but $x_{i1} = x_{ip}$. We say that a graph is *connected* if there is a path between every pair of vertices, and *complete* if there is an edge between every pair of vertices.

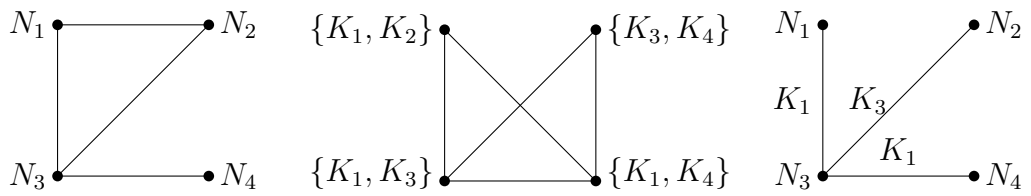
The *diameter* of a graph is the maximum 'distance' between pairs of vertices. That is, let $D(x_i, x_j)$ be the length of the shortest path between vertices x_i and x_j. Then the diameter of the graph is given by $\max_{x_i, x_j \in V} D(x_i, x_j)$. Finally, the *degree* $d(x_i)$ of a vertex x_i is the number of edges incident to that vertex. If all nodes have the same degree d, the graph is said to be *d-regular*.

2.3.2 Graph representations of KPSs

It is common to represent KPSs using simple graphs. We draw a graph of a network by representing the nodes as vertices and the 'connections' as edges. That is, we associate each node N_i with a vertex x_i. From now on, we will refer to the vertex set using the notation $V = \{N_1, N_2, \ldots, N_v\}$.

2.3.2.1 Intersection graphs

To be precise in our analysis, we distinguish between the two possible types of 'connection' and consider the separate constituent graphs of a network: the *communication graph* $G_1 = (V, E_1)$ where $(N_i, N_j) \in E_1$ if nodes N_i and N_j are within communication range, and the *key graph* $G_2 = (V, E_2)$ where $(N_i, N_j) \in E_2$ if N_i and N_j share at least q common keys. An example of a communication graph and a key graph are given in Figures 2.1(a) and 2.1(b) respectively.



(a) Comm. graph (b) Key graph (c) Intersection graph

Figure 2.1: Example of corresponding communication, key and intersection graphs

If the communication graph is complete, it is often omitted from the analysis as there is no need to check whether nodes can communicate. However, as we will explain in more detail in Section 4, the communication graph is commonly

modelled using a random graph, and it then becomes important to analyse how the communication and key graphs relate to each other.

We say that two nodes N_i and N_j can *communicate securely* if $(N_i, N_j) \in E_1 \cap E_2$, that is if they are adjacent in the *intersection* graph $G_1 \cap G_2 = (V, E_1 \cap E_2)$. This is illustrated in Figure 2.1(c). Where two nodes are not adjacent in the intersection graph, there are possible key establishment and message routing protocols which can be used to allow communication between them; we give further details in Section 2.4.4.

2.3.3 Combinatorial designs

Many KPS constructions rely on combinatorial designs, and so we provide an introduction to the theory of designs here. A brief review of existing KPSs constructed from designs is given in Section 2.4.2, and we present some examples in more detail in Chapter 4.

The following definitions are widely accepted throughout the literature, and were compiled with reference to [10, 11, 15, 68], to which we refer the interested reader for further details and examples. In particular, [15] explains the links between design theory and coding theory.

Definition 2.2. The *power set* of a set $\mathcal{X}$ is the set of all subsets of $\mathcal{X}$, and is denoted $\mathcal{P}(\mathcal{X})$.

Definition 2.3. A *set system (on $\mathcal{X}$)* is a pair $(\mathcal{X}, \mathcal{B})$ where $\mathcal{X}$ is a set and $\mathcal{B} \subseteq \mathcal{P}(\mathcal{X})$.

For example,

$$\begin{aligned}\mathcal{X} &= \{1,2,3,4,5\},\\ \mathcal{B} &= \{\{4\},\{1,3\},\{2,5\},\{3,4,5\},\{1,2,3,4,5\}\}\end{aligned}$$

is a set system.

A *combinatorial design* (or, when the context is clear, a *design*) is a general term used to describe a set system with some specified conditions such as regularity, uniformity or set intersection, as we shall now explain.

In the context of combinatorial designs, the elements of the set $\mathcal{X}$ are called *points* and the elements of $\mathcal{B}$ are called *blocks*. The *degree* of a point $x \in \mathcal{X}$ is the number of blocks containing x. We say that $(\mathcal{X}, \mathcal{B})$ is a *regular* design of degree r if every point has degree r. The *rank* is defined to be the size of the largest block. If all blocks have the same size, k, then the design is said to be *uniform* of rank k and is often called a *block design*.

We usually add a prefix to the word 'design' to specify the properties of the set system in question, for example, we define a $t-(v,k,\lambda)$ *design* to be a pair $(\mathcal{X}, \mathcal{B})$ where $|\mathcal{X}| = v$, uniform of rank k, and every set of t points is contained in exactly λ blocks.

Example 2.4. *(from [56]) Let* $\mathcal{X} = \{1,2,3,4,5,6,7,8,9\}$ *and* $\mathcal{B} = \{\{123\}, \{456\},\{789\},\{147\},\{258\},\{369\},\{159\},\{267\},\{348\},\{168\},\{249\},\{357\}\}$.

Then $(\mathcal{X}, \mathcal{B})$ *is a block design which is regular of degree four and uniform of rank three. Notice that every pair of points occurs in exactly one block, and so this is a* $2-(9,3,1)$ *design.*

Combinatorial designs were first proposed for use in KPSs in [16]. A KPS can be constructed from a design by associating a key with each point, and a node with each block. That is, node N_j is given the set of keys $\{K_i : i \in B_j\}$, where B_j is a block in $\mathcal{B}$. Thus, Example 2.4 could be used to create a KPS for twelve nodes $N_1, N_2, \ldots, N_{12}$ using nine keys, $K_1, K_2, \ldots, K_9$, where the first node N_1 stores keys K_1, K_2, K_3, node N_2 stores keys K_4, K_5, K_6, ..., and node N_{12} stores keys K_3, K_5, K_7. Figure 2.2 demonstrates a graph representation of the KPS associated with Example 2.4. The graph is regular of degree $k(r-1) = 3 \times 3 = 9$. For ease of notation, we write key K_i simply as 'i'.

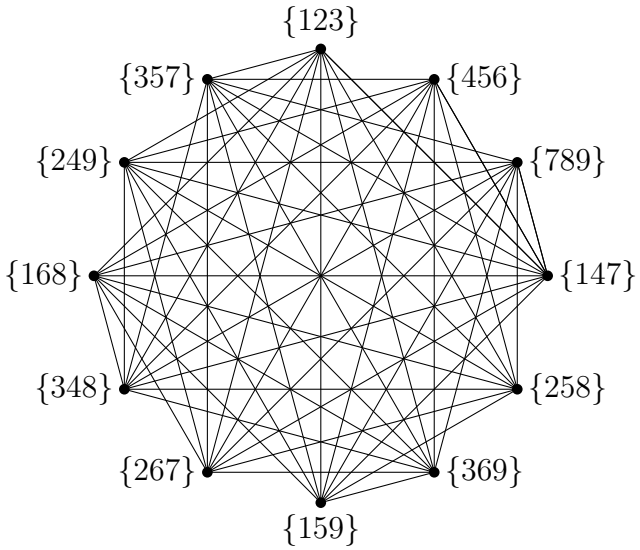


Figure 2.2: Graph representation of KPS from Example 2.4

Finally, we introduce the definitions of some classes of designs which have been used to construct KPSs. In Section 4.5.3.1 we will discuss some properties of these designs which make them particularly suitable for constructing KPSs.

Definition 2.4. (from [47, Definition 1.2]) A design $(\mathcal{X}, \mathcal{B})$ with $|\mathcal{X}| = n$, $|\mathcal{B}| = v$ is called a (n, v, r, k)-*configuration* if it is regular of degree r, uniform

of rank k and any two points occur in at most one block.

A class of configurations called μ-common intersection designs were defined by Lee and Stinson in [46, 47].

Definition 2.5. Let $(\mathcal{X}, \mathcal{B})$ be a (n, v, r, k)-configuration. We say that $(\mathcal{X}, \mathcal{B})$ is a *μ-common intersection design* if, for blocks B_i and B_j, either

$$B_i \cap B_j \neq \emptyset$$

or

$$|\{B_k \in \mathcal{B} : B_i \cap B_k \neq \emptyset \text{ and } B_j \cap B_k \neq \emptyset\}| \geq \mu.$$

In terms of the key graph of a KPS, this means that if nodes N_i and N_j corresponding to blocks B_i and B_j do not share any keys and so are not adjacent, then they have at least μ common neighbours, i.e. μ nodes with which they both share a key.

Strongly regular graphs may be regarded as a special type of μ-common intersection design, and are defined as follows.

Definition 2.6. (from [14]) A $(v, k(r-1), \lambda, \mu)$-*strongly regular graph* is a graph on v vertices which is regular of degree $k(r-1)$ and has the following properties:

- any two adjacent vertices have exactly λ common neighbours
- any two nonadjacent vertices have exactly μ common neighbours.

Equivalently, the design $(\mathcal{X}, \mathcal{B})$ is a *strongly regular graph* if it is regular of degree r, uniform of rank k, and for blocks B_i and B_j,

$$B_i \cap B_j \neq \emptyset \quad \Rightarrow \quad |\{B_k \in \mathcal{B} : B_i \cap B_k \neq \emptyset \text{ and } B_j \cap B_k \neq \emptyset\}| = \lambda$$

and

$$B_i \cap B_j = \emptyset \quad \Rightarrow \quad |\{B_k \in \mathcal{B} : B_i \cap B_k \neq \emptyset \text{ and } B_j \cap B_k \neq \emptyset\}| = \mu$$

Strongly regular graphs have been shown to exist for various combinations of the parameters v, k and r in [45]. Constructions are given in [14, 67, 47] and we refer the reader to [13] for a discussion on constructing random strongly regular graphs.

2.4 Existing schemes and techniques

Having seen trivial examples of KPSs in the preceding sections, we now present a brief summary of the KPSs which have been proposed in the literature to provide practical trade-offs between the conflicting metrics of key storage, connectivity and resilience. There are randomised and deterministic constructions for KPSs, and we give an overview of these in Sections 2.4.1 and 2.4.2 respectively. In Section 2.4.3 we discuss other schemes which are outside of our scope, but have strong similarities to key predistribution schemes. Finally, in Section 2.4.4, we outline possible methods for communication between non-adjacent nodes in a deployed network, which motivates later discussion in Chapters 4 and 5 about the desirable properties of KPSs.

2.4.1 Random KPSs

One approach to key predistribution is through randomised allocation of keys. The seminal paper by Eschenauer and Gligor [32] presented the first randomised approach to key predistribution, as follows:

Scheme 2.1 (**Eschenauer Gligor random key predistribution**). *A key pool $\mathcal{K}$ of n symmetric keys is generated from the space of all possible keys. Each node is independently assigned a k-subset of keys from the key pool, chosen uniformly at random from the set of all k-subsets of $\mathcal{K}$. (That is, each node stores k distinct keys; for each node the keys are chosen without replacement.) Nodes are deployed into the environment and use a shared key discovery protocol such as those described in [17, 73] to identify the other nodes with which they share keys.*

Two nodes are said to be 'connected' if they have at least one key in common. If they have more than one key in common, they should select a single one of their common keys at random to use to secure their communications.

This KPS can achieve high connectivity with relatively low key storage by careful choice of the size of the key pool. We consider the Eschenauer Gligor scheme in more detail in Chapter 3, where we state and prove the formulae for calculating its connectivity and resilience, and discuss the resulting key graph. In the same chapter we also describe some adaptations of the scheme which provide different trade-offs between the key storage, connectivity and resilience.

We mention here two other random key predistribution schemes from [19], namely the *multipath key reinforcement scheme* and the *random pairwise keys scheme*:

Scheme 2.2 (**The multipath key reinforcement scheme** [19])**.** *We mention this scheme here as it is presented in [19] as a random key predistribution scheme. However, the key predistribution stage is the same as that for Scheme 2.1, and so multipath reinforcement should perhaps be regarded as a protocol for a deployed network, as discussed in depth in Sections 2.4.3 and 2.4.4. Briefly, the multipath key reinforcement protocol allows any nodes which share a key to update it using any disjoint paths between them in the network, thereby improving the resilience of the network whilst maintaining the connectivity and key storage. Such a protocol could be used after any initial key predistribution scheme to improve resilience, as discussed in Section 2.4.4.*

Scheme 2.3 (**The random pairwise keys scheme** [19])**.** *This scheme allocates 'pairwise' keys to a random subset of all possible pairs of nodes. By 'pairwise', we mean that if a pair of nodes share a key, that key is unique. Graph-theoretically, the scheme achieves the following. Imagine the complete pairwise key graph, where every node stores* $v - 1$ *unique keys. Delete edges from this graph at random, until the graph is still connected with some acceptable probability* p*. Now assign the appropriate keys to each node so that this is the resulting key graph. This KPS has perfect resilience and is connected with probability* p*.*

However, most random key predistribution schemes are largely based on the Eschenauer Gligor scheme. Since the scope of Chapter 3 is to provide a gen-

eralised formula for the resilience of a class of random KPSs, we will reserve further discussion of the variety of random KPSs for the next chapter.

2.4.2 Deterministic KPSs

Since 2004, many different deterministic constructions have been proposed for KPSs such as [16, 46, 70]. Deterministic schemes can provide advantages over random KPSs such as

- deterministic rather than probabilistic connectivity and resilience metrics
- more efficient shared key discovery

For further details on the benefits of deterministic KPSs over random KPSs, see [48].

The majority are based on combinatorial designs, error-correcting codes and/or graph constructions. We refer the reader to [17, 50, 56] for surveys of these schemes, and in particular we note that [56] unifies various combinatorial approaches to KPS construction, pointing out that some apparently different constructions are in fact equivalent.

We will present and analyse examples of deterministic KPSs constructed from graphs and combinatorial designs in more detail in Chapters 4 and 5. Here we make some observations which motivate the use of certain classes of designs in constructing KPSs.

In [47] it is observed that the diameter of the key graph of a μ-common intersection design is two, and that if two nodes are not adjacent then they will have at least μ common neighbouring nodes. We will examine the various ways in which common neighbours can be beneficial in Section 2.4.4. For an analysis of the connectivity and resilience of KPSs based on μ-common intersection designs, see [45, 56]. In Section 4.5.3.2 we show that their good expansion provides another reason to support the choice of μ-common intersection designs for constructing KPSs.

Finally, a KPS constructed from a $(v, k(r-1), \lambda, \mu)$-strongly regular graph which is neither complete ($k = v - 1$, $\lambda = v - 2$) nor disconnected ($\mu = 0$ and $k = \lambda + 1$) provides good trade-offs between key storage, connectivity and resilience [46]. Clearly, the key graph will also have diameter two, and by proving a lower bound on the expansion in Section 4.5.3.3 we will show that they provide many desirable properties for use in KPSs.

2.4.3 Key establishment

We note the existence of many schemes which are closely related to key predistribution schemes. Although they do not fall within our definition of key predistribution schemes, they are closely related, and some authors do refer to them as key predistribution schemes.

In schemes such as [6, 9] which pre-date the key predistribution schemes literature, nodes are preloaded not with *keys*, but with secret information or 'keying materials' from which they can generate or establish keys for pairwise or group

communication. In common with much of the KPS literature, we categorise such schemes as 'key generation schemes' or 'key establishment schemes', and do not include them in our definition of key predistribution schemes. A self-contained introduction to key establishment schemes is given in [8].

Finally, we note that *key distribution patterns* as described in [63, Section 10.4] can be considered as early KPSs, differing only in that they are concerned with distributing keys for groups of two or more users, and providing a threshold level of security ω. That is, typically these schemes have the property that for some $\omega \in \mathbb{N}$, $\mathsf{fail}_s = 0$ for $1 \leq s \leq \omega$.

2.4.4 Communication between non-adjacent nodes

Finally, we discuss possible methods for communication between non-adjacent nodes. Although we will not focus on such protocols in detail, their existence motivates some design goals which we shall see in Chapters 4 and 5.

If two nodes are not adjacent in the intersection graph then there are various techniques which can enable them to securely communicate:

- If N_i and N_j do not share a key but share a single common neighbour N_k, then a message can be sent from N_i to N_k encrypted by their shared key(s), decrypted by N_k, and then re-encrypted by N_k and sent to N_j using their shared key(s). This process of encrypting and decrypting messages along a path through the network is known as *link encryption*.
- If N_i and N_j share μ common neighbours $N_{k1}, \ldots, N_{k\mu}$ (as in a KPS based

on a μ-common intersection design, Section 4.5.3.2) then the message M can be split into μ random-looking bit strings $M_1, \ldots, M_\mu$, so that if N_j receives all μ message strings, it can recover M. For example, this could be achieved by creating $\mu - 1$ (pseudo)random bit strings of length $|M|$, and choosing the final string M_μ so that $M_1 \oplus \cdots \oplus M_\mu = M$ as described in [26]. Then N_i can send each message string via one of the common neighbours $N_{k1}, \ldots, N_{k\mu}$ using link encryption. The larger the number of common neighbours, the less chance an adversary has of compromising all the neighbours and recovering M. However, this comes at the cost of increased communication through the network.

- In a similar way, common neighbours can be used to establish a shared key between N_i and N_j. Creating a new key will reduce the communication overheads in the network after the key is established, in comparison to using common neighbours for every communication. Nodes N_i and N_j can agree a new key K_{ij} by one node creating a secret key (preferably using a good pseudo-random number generator) and transmitting it to the other node by sending a share of it via each of their common neighbours, in the way described above. More sophisticated methods are also possible, such as the one given in [29] which additionally provides a threshold level of fault tolerance.

- Finally, a pair of nodes which already share a key K_{ij} may use any common neighbours to *reinforce* their key, by which we mean creating a new key K'_{ij} which is less vulnerable to compromise by an adversary. This is achieved by ensuring that the adversary can only discover K'_{ij} by compromising every node used in the reinforcement, which is comprised of

a key establishment scheme as above, followed by XORing the newly established key with the existing key K_{ij}. Such a protocol, called *multipath reinforcement*, is presented in [20] and was introduced briefly in Section 2.4.1.

For all of the above methods, it is clearly preferable in terms of communication overheads and resilience if the diameter and average path length in the intersection graph are small, so that 'most' non-adjacent nodes have at least one common neighbour or only a short path between them. If two nodes do not share a common neighbour but have multiple short paths between them, then better resilience is provided for messages routed or keys shared along these paths if they are *disjoint*. This is because if the paths are not disjoint, the adversary could focus on compromising nodes which lie on the intersection of the paths to minimise the number of nodes which have to be compromised in order to recover the message or key.

In Chapters 4 and 5 we will see methods for ensuring that the diameter and average path length of the key graph are small.

Chapter 3

Generalised formula for the resilience of random key predistribution schemes

Contents

This work, joint with Ed Kendall, Wilfrid S. Kendall and Keith M. Martin, appears as a paper on the Cryptology ePrint Archive [41].

3.1 Introduction

As explained in Section 2.2.3, a commonly used metric for comparing the resilience of key predistribution schemes is fail_s, which measures the proportion of network connections which are 'broken' by an adversary which has compromised s nodes. Correct analysis of schemes is fundamental to the proper assessment of KPSs. In [20], Chan, Perrig and Song present a formula for measuring the resilience in a class of random key predistribution schemes called q-composite schemes. We explain how this formula makes an incorrect assumption about independence, and present a correction. Our corrected formula features an additional parameter which makes it applicable to a wider variety of random key predistribution schemes, including the original Eschenauer Gligor scheme [32]. We also present a simplification of their formula for connectivity.

We refer to the paper by Yum and Lee [75] which also claims to correct the original formula for the q-composite scheme. However the resulting formula is complicated, computationally demanding, and hard to understand. The formula which we propose and prove is easily computable and can be applied to a wider range of schemes.

In Section 3.2 we give the details of two random key predistribution schemes and provide proofs of their connectivity and resilience parameters. In Section 3.3 we state the previously proposed formulae for the resilience of q-composite schemes and discuss issues arising in their proofs, before presenting and proving our generalised formula for fail_s in Section 3.4. Finally, in Section 3.5 we analyse the difference between our formula and that given in [20],

which can be considered an upper bound on the true value.

3.2 Background: random key predistribution schemes

For deterministic schemes, fail_s can usually be computed using exact knowledge of how many nodes store each key. In [56], Paterson and Stinson generalise the fail_s calculation across a range of deterministic schemes. For random key predistribution schemes, the number of nodes which store each key is only known probabilistically, adding another layer of complexity to the calculation.

Here we present two examples of random key predistribution schemes. We derive their respective connectivity and resilience formulae in order to demonstrate some of the methods for proving our main result, the generalised formula for fail_s in random key predistribution schemes. We also provide a simplified formula for the probability of two nodes having exactly i keys in common.

3.2.1 Eschenauer Gligor KPS

Recall the Eschenauer Gligor KPS [32] which we presented as Scheme 2.1 in Section 2.4.1, where each node is allocated a random k-subset of keys from a key pool $\mathcal{K}$, where $|\mathcal{K}| = n$. We noted that two nodes are connected if they have at least one key in common. Where nodes share more than one common key, they should select one of them at random to use to secure their communications. To be precise, we introduce a parameter Ω which is the

maximum number of common keys which two nodes may use to secure their communications. For the Eschenauer Gligor scheme, $\Omega = 1$.

3.2.1.1 Connectivity

We now present the probability Pr_1 of two nodes being connected in this scheme. The original paper presents and proves an equivalent expression of this formula using factorials; we use the binomial coefficient notation for consistency with the majority of the subsequent literature.

Lemma 3.1 (Eschenauer Gligor connectivity)**.** *The probability of two nodes being connected in an Eschenauer Gligor random key predistribution scheme with key pool size n and key storage k is*

$$\mathsf{Pr}_1 = 1 - \frac{\binom{n-k}{k}}{\binom{n}{k}} \quad .$$

Proof. Suppose that two nodes N_i, N_j store key sets $\mathcal{K}_{N_i}, \mathcal{K}_{N_j}$ respectively. The probability that they are connected is

$$1 - \mathsf{Pr}[\text{they have no keys in common}] = 1 - \mathsf{Pr}[\mathcal{K}_{N_i} \cap \mathcal{K}_{N_j} = \emptyset].$$

Fix $\mathcal{K}_{N_i}$. Then there are $\binom{n-k}{k}$ ways to pick a k-subset of keys for node N_j so that $\mathcal{K}_{N_i} \cap \mathcal{K}_{N_j} = \emptyset$, out of the total possible $\binom{n}{k}$ ways to pick $\mathcal{K}_{N_j}$. □

Remark 3.1. *We note that it is a common assumption in the literature that the key graph of the Eschenauer Gligor scheme is equivalent to the Erdös-Rényi random graph $G(v, \mathsf{Pr}_1)$ [31], as asserted in the original paper [32] and in [20]. That is, the key graph is a random graph on v vertices, where each edge exists with probability Pr_1, so that there are approximately $\binom{v}{2}\mathsf{Pr}_1$ edges. However,*

this is not the case, as the edge existence probabilities are interdependent. As a simple example, suppose that the key storage is $k = 1$*, and that for some nodes* N_a, N_b, N_c *we have* $(N_a, N_b) \in E$ *and* $(N_b, N_c) \in E$*. Then the probability that* $(N_a, N_c) \in E$ *is* 1*, and not dependent on* n*, the size of the key pool.*

This observation has also been made in [5, 27, 74]. These papers prove that the Eschenauer Gligor key graph is different from the Erdös-Rényi graph; in particular, for large networks the expected number of triangles is orders of magnitude larger in the key graph [5, 74] and the connectivity threshold is lower. Using the Erdös-Rényi random graph to model the Eschenauer Gligor key graph is therefore pessimistic, in the sense that the key storage required for the graph to be connected is lower than expected by the Erdös-Rényi model [27].

3.2.1.2 Resilience

Eschenauer and Gligor do not calculate the resilience of their scheme in the way that we have defined. They do, however, make the observation that in a simulation, only 50% of the keys from the key pool were used to secure links: 30% were used to secure a single link, 10% to secure two links and 5% to secure three links. Thus the compromise of a single key compromises exactly one other link with probability 0.1.

The standard metric fail_s for the Eschenauer Gligor scheme is indirectly stated within another result in [20]. Here we state and prove it formally.

Lemma 3.2 (Eschenauer Gligor resilience)**.** *In an Eschenauer Gligor random key predistribution scheme with key pool size* n *and key storage* k*, the resilience*

is given by

$$\mathsf{fail}_s = 1 - \left(1 - \frac{k}{n}\right)^s \quad . \tag{3.2.1}$$

Proof. Fix a random link in the network between uncompromised nodes N_i and N_j, and suppose that they use key K_i to secure their connection.

We begin by considering $s = 1$, that is, the adversary has compromised a single node. Let X be a uniformly random k-subset of the key pool $\mathcal{K} = \{K_1, \ldots, K_n\}$, so that it represents the keys known to the adversary after compromising one node. Then

$$\begin{aligned}
\mathsf{fail}_1 &= \mathsf{Pr}[K_i \in X] \\
&= 1 - \mathsf{Pr}[K_i \notin X] \\
&= 1 - \frac{\binom{n-1}{k}}{\binom{n}{k}} \\
&= 1 - \left(1 - \frac{k}{n}\right) \quad .
\end{aligned}$$

Now we generalise for $s > 1$. Let $X_1, \ldots, X_s$ be independent uniformly random subsets of the key pool, each of size k. Then

$$\begin{aligned}
\mathsf{fail}_s &= \mathsf{Pr}[K_i \in X_1 \cup \cdots \cup X_s] \\
&= 1 - \mathsf{Pr}[K_i \notin X_1 \cup \cdots \cup X_s] \\
&= 1 - (\mathsf{Pr}[K_i \notin X_1])^s \\
&= 1 - \left(1 - \frac{k}{n}\right)^s \quad .
\end{aligned}$$

□

3.2.2 The q-composite scheme

In many key predistribution schemes, it is possible that a pair of nodes N_i and N_j have more than one key in common. If $\Omega = k$, nodes may use all of their $\omega \leq k$ common keys $\mathcal{K}_{N_i} \cap \mathcal{K}_{N_j} = \{K_{t_1}, K_{t_2}, \ldots, K_{t_\omega}\}$ to secure the link, for example by calculating their shared key to be

$$K_{ij} = h(K_{t_1}||K_{t_2}||\cdots||K_{t_\omega}) \ ,$$

where h is a suitable function such as a hash function (see [63, Chapter 4] for an introduction), and where there is a well-defined ordering on the keys $t_1 < t_2 < \cdots < t_\omega$ so that K_{ij} is uniquely defined. Since an adversary would have to learn all ω keys to compromise the link, such schemes have better resilience than those where $\Omega = 1$, such as Scheme 2.1. However, changing Ω does not affect the connectivity.

Chan et al. [20] present a random KPS which requires nodes to have $q > 1$ keys in common in order to be connected, called the *q-composite scheme*. We give the formal details below. Intuitively, for the same key pool size n and key storage k, nodes are less likely to be connected in the q-composite scheme than in the Eschenauer Gligor scheme, but the resilience increases with q. Such a trade-off may be advantageous for some applications, and the sizes of n, k and q can be adapted to provide a desirable level of connectivity with as high a resilience as possible.

Scheme 3.1 (**q-composite scheme** [20])**.** *The q-composite scheme is similar to Scheme 2.1, except that nodes must share at least $q > 1$ keys before they are allowed to compute a common key, and $\Omega = k$. That is, nodes with fewer than*

q keys in common will not be able to communicate directly, and nodes with q or more keys in common should hash all of their common keys to create their link key.

3.2.2.1 Connectivity

We consider the connectivity of the q-composite scheme, that is, the probability that a pair of nodes share q or more keys. We omit the full proof here because it is given in [20] and reproduced in [75]. However, we provide an improvement: the value of $p(i)$, the probability of a pair of nodes sharing exactly i keys, has previously been given as

$$p(i) = \frac{\binom{n}{i}\binom{n-i}{2(k-i)}\binom{2(k-i)}{k-i}}{\binom{n}{k}^2} \quad ,$$

but we provide an equivalent, simpler expression in Lemma 3.3. Our formula for $p(i)$ can be derived from the original by expanding the binomial coefficients and rearranging, but we provide a direct combinatorial proof:

Lemma 3.3. *In a random key distribution scheme where nodes are allocated a random k-subset of keys from a key pool of size n, the probability that a pair of nodes shares exactly i keys is given by*

$$p(i) = \frac{\binom{n-k}{k-i}\binom{k}{i}}{\binom{n}{k}} \quad . \tag{3.2.2}$$

Proof. We consider the probability of two nodes N_1 and N_2 having exactly i keys in common. Fix i keys from N_1's set of keys. For N_2 to have $(k-i)$ keys which are *unknown* to N_1, it must have $(k-i)$ keys chosen from the $(n-k)$ keys unknown to N_1. Thus there are $\binom{n-k}{k-i}$ ways to do this, out of the $\binom{n}{k}$

ways to choose keys for N_2. Finally, we multiply by the number of ways to fix i keys from N_1's set of k keys. □

Thus we can see that the connectivity of a q-composite scheme is given by the following formula:

Theorem 3.4 (from [20]). *In a q-composite scheme with key pool size n and key storage k, the connectivity probability is*

$$\mathsf{Pr}_1 = 1 - \sum_{i=0}^{q-1} p(i) \ . \tag{3.2.3}$$

Remark 3.2. *Notice that* (3.2.3) *is a generalised formula for the probability of connectivity, which agrees with that given in Lemma 3.1 for the Eschenauer Gligor scheme: setting $q = 1$ into* (3.2.3) *gives*

$$\begin{aligned} \mathsf{Pr}_1 &= 1 - p(0) \\ &= 1 - \frac{\binom{n-k}{k}}{\binom{n}{k}} \ . \end{aligned}$$

3.3 Previous formulae for the resilience of the q-composite scheme

We now discuss approaches which have been proposed for calculating the resilience of the q-composite scheme.

3.3.1 Chan, Perrig and Song

In [20], Chan et al. give the following formula:

$$\mathsf{fail}_s = \sum_{i=q}^{k} \left(1 - \left(1 - \frac{k}{n}\right)^s\right)^i \frac{p(i)}{\mathsf{Pr}_1} \ . \tag{3.3.1}$$

However, the proof is informal and incorrectly assumes independence between certain events, as we explain below.

Before we explain why this formula for resilience is incorrect, we first note an aspect of the notation in the original formula which has caused some confusion in the subsequent literature. Chan et al. consider a parameter p, defined to be the minimum node-node connectivity probability needed to make the whole network connected with some high probability. They then define $p_{connect} = 1 - \sum_{i=0}^{q-1} p(i)$ and state that the key pool size n should be chosen to be the largest (integer) such that $p_{connect} \geq p$, which is a sensible way to reduce unnecessary connectivity and keep resilience high. However, in their resilience formula, they redefine p to equal $p_{connect}$. This has caused errors to be made in its reproduction, for example in [75]. We will always use the notation Pr_1 as defined in (3.2.3) to avoid confusion, and for consistency with much of the deterministic key predistribution literature.

As Yum and Lee point out in [75], the problem with (3.3.1) is an incorrect assumption of independence. Suppose that, in a 2-composite scheme, a pair of nodes share keys K_1 and K_2. For an adversary to break the link between these nodes requires knowledge of both K_1 and K_2. Let A_{K_i} be the event that an adversary knows key K_i, and suppose that after compromising s nodes an

adversary knows x keys. Equation (3.3.1) assumes that

$$\mathsf{Pr}[A_{K_1} \wedge A_{K_2}] = \mathsf{Pr}[A_{K_1}]\mathsf{Pr}[A_{K_2}] = \left(\frac{x}{n}\right)^2 \quad .$$

However, this is not true because the events are not independent. Consider the conditional probability $\mathsf{Pr}[A_{K_2}|A_{K_1}]$. If the adversary already knows key K_1, then it is slightly less likely that the adversary also knows K_2, indeed, $\mathsf{Pr}[A_{K_2}|A_{K_1}] = \frac{x-1}{n}$. Thus, the calculation leads to an overestimation of the true value of fail_s, as we demonstrate in Section 3.5.

3.3.2 Yum and Lee

In [75], another formula is proposed for the calculation of fail_s for q-composite schemes. However, the formula is difficult to compute, as we now demonstrate.

In [75, Theorem 2], Yum and Lee propose that fail_s for the q-composite scheme is given by

$$\sum_{\tau=k}^{\min\{ks,n\}} \left[\binom{n}{\tau} \left(\sum_{j=q}^{k} \frac{\binom{\tau}{j}}{\binom{n}{j}} \frac{p(j)}{\mathsf{Pr}_1} \right) \left(\frac{\binom{\tau}{k}^s - \sum_{\lambda=1}^{\tau-k} (-1)^{\lambda+1} \binom{\tau}{\lambda} \binom{\tau-\lambda}{k}^s}{\binom{n}{k}^s} \right) \right] \quad . \tag{3.3.2}$$

This formula is complicated and computationally laborious to evaluate; in addition we had difficulty in following the proof. We present a direct proof of a computationally simpler formula in Corollary 3.7 below. We also note that, whilst we are able to compute (3.3.2) for small values of n such as $n = 17$, our results are different from those given in [75, Table 1]. We are unable to reproduce any of their sample values, either by interpreting the 'p' in (3.3.2) to mean Pr_1 or $p_{connect}$. We conclude that there must be a typographical error somewhere in their formula and/or proof.

3.4 Generalised resilience for random key predistribution schemes

In order to generalise across many instantiations of random key predistribution schemes, we have introduced a parameter $\Omega \leq k$, which acts as an upper bound on the number of shared keys which nodes can use to compute their link key. This allows us to derive a formula which describes the resilience of many different random KPSs, including the schemes described in Section 3.2. We show in Corollary 3.6 that our formula is equivalent to that of Scheme 2.1 in the special case when $q = \Omega = 1$.

We now present our generalised formula for fail_s, which applies to any key predistribution scheme where:

1. each node is allocated k keys, selected independently and uniformly at random without replacement from a pool of n keys;
2. the intersection threshold is $q \geq 1$, that is, nodes may only establish a link key if they share at least q keys;
3. the upper bound on the number of shared keys a pair of nodes may use is Ω, where $q \leq \Omega \leq k$; if two nodes share more than Ω keys then they should pick Ω of their keys at random to compute their link key;
4. suppose that a pair of nodes use ω shared keys to create a link key, where $q \leq \omega \leq \Omega$. We require that the function (such as hash, XOR, etc.) for producing the single link key is such that an adversary must know all of the ω shared keys between a pair of nodes to break the link; if the

adversary only knows at most $\omega - 1$ of the keys then the link remains secure.

Theorem 3.5. *For any random key predistribution scheme which fulfils conditions (1)–(4) above, the resilience is given by*

$$\begin{aligned}\mathsf{fail}_s &= \frac{1}{\mathsf{Pr}_1}\left(\sum_{\omega=q}^{\Omega}\left[1-\sum_{i=1}^{\omega}(-1)^{i-1}\binom{\omega}{i}\left(\frac{\binom{n-i}{k}}{\binom{n}{k}}\right)^s\right]p(\omega)\right)+ \\ &\quad \frac{1}{\mathsf{Pr}_1}\left(\left[1-\sum_{i=1}^{\Omega}(-1)^{i-1}\binom{\Omega}{i}\left(\frac{\binom{n-i}{k}}{\binom{n}{k}}\right)^s\right]\sum_{\omega=\Omega+1}^{k}p(\omega)\right) \quad . \quad (3.4.1)\end{aligned}$$

Proof. Consider a randomly-chosen pair of uncompromised nodes which share ω keys, where $q \leq \omega \leq \Omega$. For ease of notation and without loss of generality, we label these keys $\{1, 2, \ldots, \omega\}$. The probability that all of these ω keys are known to an adversary which has compromised s nodes is

$$\mathsf{Pr}[\{1,\ldots,\omega\} \subseteq \{X_1 \cup \cdots \cup X_s\}] = 1 - \mathsf{Pr}\left[\bigcup_{i=1}^{\omega} B_i\right] \quad ,$$

where B_i is the event that key $i \notin \{X_1 \cup \cdots \cup X_s\}$. Using inclusion-exclusion, we have

$$\begin{aligned}1-\mathsf{Pr}\left[\bigcup_{i=1}^{\omega} B_i\right] &= 1-\omega\mathsf{Pr}[1 \notin \{X_1 \cup \cdots \cup X_s\}] \\ &\quad +\binom{\omega}{2}\mathsf{Pr}[1,2 \notin \{X_1 \cup \cdots \cup X_s\}] + \cdots \\ &\quad (-1)^{i-1}\binom{\omega}{i}\mathsf{Pr}[1,\ldots,i \notin \{X_1 \cup \cdots \cup X_s\}] + \cdots \\ &= 1-\sum_{i=1}^{\omega}(-1)^{i-1}\binom{\omega}{i}\left(\frac{\binom{n-i}{k}}{\binom{n}{k}}\right)^s \quad .\end{aligned}$$

The probability of a randomly-chosen connected pair of uncompromised nodes sharing exactly ω keys ($q \leq \omega \leq \Omega$) is $\frac{p(\omega)}{\mathsf{Pr}_1}$. Therefore, for $q \leq \omega \leq \Omega$ we have

$$\mathsf{fail}_s = \frac{1}{\mathsf{Pr}_1}\left(\sum_{\omega=q}^{\Omega}\left[1-\sum_{i=1}^{\omega}(-1)^{i-1}\binom{\omega}{i}\left(\frac{\binom{n-i}{k}}{\binom{n}{k}}\right)^s\right]p(\omega)\right) \quad .$$

For $\Omega < \omega \leq k$, the probability of two connected nodes sharing ω keys is again $\frac{p(\omega)}{\mathsf{Pr}_1}$. However, only Ω of these keys will be used to secure the link, and the choice of these Ω is made a priori, uniformly at random, and so without loss of generality they can be labelled $1, 2, \ldots, \Omega$. Therefore the probability of the adversary knowing all Ω keys is

$$\mathsf{Pr}[\{1, \ldots, \Omega\} \subseteq \{X_1 \cup \cdots \cup X_s\}] = 1 - \sum_{i=1}^{\Omega} (-1)^{i-1} \binom{\Omega}{i} \left(\frac{\binom{n-i}{k}}{\binom{n}{k}} \right)^s ,$$

using the result above, and so for $\Omega < \omega \leq k$,

$$\mathsf{fail}_s = \frac{1}{\mathsf{Pr}_1} \left(\left[1 - \sum_{i=1}^{\Omega} (-1)^{i-1} \binom{\Omega}{i} \left(\frac{\binom{n-i}{k}}{\binom{n}{k}} \right)^s \right] \sum_{\omega=\Omega+1}^{k} p(\omega) \right) .$$

Adding these two results gives the final formula for fail_s. □

We now demonstrate that our formula agrees with that given in Lemma 3.2, in the case where $q = \Omega = 1$.

Corollary 3.6 (Eschenauer Gligor resilience revisited)**.** *The resilience of a random KPS which fulfils conditions (1)–(4) and where $q = \Omega = 1$ (such as the Eschenauer Gligor scheme), is given by*

$$\mathsf{fail}_s = 1 - \left(1 - \frac{k}{n}\right)^s .$$

Proof. Setting $q = \Omega = 1$ in Equation (3.4.1) gives

$$\begin{aligned} \mathsf{fail}_s &= \frac{1}{\mathsf{Pr}_1} \left(\left[1 - (-1)^0 \binom{1}{1} \left(\frac{\binom{n-1}{k}}{\binom{n}{k}} \right)^s \right] p(1) + \right. \\ &\qquad \left. \left[1 - (-1)^0 \binom{1}{1} \left(\frac{\binom{n-1}{k}}{\binom{n}{k}} \right)^s \right] \sum_{\omega=2}^{k} p(\omega) \right) \\ &= \frac{\sum_{\omega=1}^{k} p(\omega)}{\mathsf{Pr}_1} \left(1 - \left(\frac{\binom{n-1}{k}}{\binom{n}{k}} \right)^s \right) . \end{aligned}$$

Since Pr_1 is by definition the sum of the probabilities of having $1, 2, \ldots, k$ keys in common, the first fraction is equal to 1, and we have

$$\begin{aligned}\mathsf{fail}_s &= 1 - \left(\frac{(n-1)!}{(n-1-k)!k!} \Big/ \frac{n!}{(n-k)!k!}\right)^s \\ &= 1 - \left(1 - \frac{k}{n}\right)^s ,\end{aligned}$$

as required. □

It is now straightforward to derive the correct formula for the resilience of Scheme 3.1 from Theorem 3.5:

Corollary 3.7 (q-composite resilience)**.** *The resilience of a random KPS that fulfils conditions (1)–(4) and where $q > 1$ and $\Omega = k$ (such as the q-composite scheme from [20]) is given by*

$$\mathsf{fail}_s = \frac{1}{\mathsf{Pr}_1}\left(\sum_{\omega=q}^{k}\left[1 - \sum_{i=1}^{\omega}(-1)^{i-1}\binom{\omega}{i}\left(\frac{\binom{n-i}{k}}{\binom{n}{k}}\right)^s\right]p(\omega)\right) . \tag{3.4.2}$$

Proof. Using Equation (3.4.1), we observe that when $\Omega = k$ the summation from $\omega = \Omega + 1$ to $\omega = k$ vanishes, leaving the formula given above. □

3.5 Numerical examples

We now compare our corrected formula to the original expression for fail_s. In Tables 3.1 and 3.2 we contrast equations (3.3.1) and (3.4.2) for sample values within the q-composite scheme ($\Omega = k$). We fix $n = 1000$ and $k = 100$, and in Table 3.1 we fix $q = 10$ and vary s from 1 to 20. In Table 3.2 we fix $s = 10$

s	(3.3.1)	(3.4.2)	% difference
1	2.75×10^{-11}	1.79×10^{-11}	53.532634
2	1.85×10^{-8}	1.52×10^{-8}	21.345324
3	7.03×10^{-7}	6.25×10^{-7}	12.472689
4	8.27×10^{-6}	7.63×10^{-6}	6.037381
5	0.000051	0.000048	6.037381
6	0.000213	0.000204	4.544026
7	0.000669	0.000647	3.517733
8	0.001720	0.001674	2.776871
9	0.003794	0.003711	2.222941
10	0.007423	0.007292	1.797917
11	0.013196	0.013006	1.465379
12	0.021692	0.021434	1.201299
13	0.033414	0.033086	0.989159
14	0.048737	0.048342	0.817219
15	0.067871	0.067415	0.676889
16	0.090850	0.090342	0.561736
17	0.117530	0.116984	0.466844
18	0.147617	0.147046	0.388391
19	0.180696	0.180114	0.323365
20	0.216264	0.215683	0.269363

Table 3.1: Comparison of formulae when $n = 1000$, $k = 100$, $q = 10$, hence $\mathsf{Pr}_1 = 0.555019$

and vary q from 1 to 20. Differences are given as a percentage difference, that is, the final column is given by $\frac{(3.3.1)-(3.4.2)}{(3.4.2)} \times 100$.

We find that (3.3.1) gives higher values for fail_s, that is, it underestimates the resilience. As the differences are small, (3.3.1) can be thought of as an upper bound on the correct value. We note that an asymptotic approximation can be derived by routine approximation of (3.4.2) (using the basic techniques of Poisson approximation to the Binomial distribution); for example, it is a simple exercise to show that

$$\mathsf{fail}_s \approx \sum_{\omega=q}^{k} \frac{\omega!}{\mathsf{Pr}_1} \binom{k}{\omega}^2 e^{-\frac{1}{n}\left(k^2 - 2k\omega + \frac{\omega(\omega+1)}{2}\right)} \left(\frac{1 - e^{-\frac{sk}{n}}}{n}\right)^{\omega}$$

and numerically this presents as a lower bound. These approximations are

q	Pr_1	(3.3.1)	(3.4.2)	% difference
1	0.999985	0.027080	0.026874	0.765811
2	0.999802	0.026966	0.026760	0.769232
3	0.998681	0.026520	0.026314	0.782573
4	0.994211	0.025397	0.025191	0.816410
5	0.981134	0.023337	0.023133	0.881074
6	0.951193	0.020382	0.020183	0.983376
7	0.895315	0.016889	0.016701	1.126261
8	0.807913	0.013337	0.013164	1.310152
9	0.690967	0.010113	0.009960	1.534359
10	0.555019	0.007423	0.007292	1.797917
11	0.416034	0.005313	0.005204	2.099974
12	0.289839	0.003730	0.003641	2.439908
13	0.187255	0.002580	0.002510	2.817340
14	0.112090	0.001765	0.001710	3.232105
15	0.062167	0.001197	0.001154	3.684213
16	0.031964	0.000806	0.000774	4.173823
17	0.015250	0.000540	0.000516	4.701213
18	0.006759	0.000360	0.000342	5.266761
19	0.002786	0.000240	0.000226	5.870935
20	0.001070	0.000159	0.000149	6.514278

Table 3.2: Comparison of formulae when $n = 1000$, $k = 100$, $s = 10$

weakest when Pr_1 is very small (when n is large in comparison to k), but such low connectivity is unlikely to be used in practice. For more appropriate values of Pr_1 for network connectivity, the approximations become more accurate.

It can be seen that, using either formula, the value of fail_s increases in s and decreases in q. Conversely, the percentage difference decreases in s and increases in q. However, as the largest values of the percentage differences correspond to the smallest values of fail_s for both equations, the absolute error in (3.3.1) remains small.

We therefore conclude that whilst our contribution is of mathematical importance it has limited impact on applications, as the original formula from [20]

provides a close approximation to the true value.

Sample values for the formula from [75] are not given in the tables; for exact computation using Maple 15, the calculation did not terminate within an hour for input numbers of the magnitude given in the tables. By contrast, our formula can be evaluated within seconds on the same computer (AMD Phenom^{TM} II X4 970 CPU, 3.5 GHz, 16 GB RAM). Approximate calculation of (3.3.2) revealed answers appearing to converge to the results given by our formula, (3.4.2).

3.6 Conclusion

We have described two random key predistribution schemes and explained how the resilience of the q-composite scheme has been inaccurately presented in the literature. We have derived a formula for fail_s which is rigorously proven, practical to compute, and applicable to a wide range of random key predistribution schemes because of the parameter Ω. Notice that if we take a scheme with $\Omega = k$ and change Ω to be in the range $q \leq \Omega < k$, then connectivity remains unchanged but resilience is reduced. Whilst this may be undesirable for many applications, setting $\Omega < k$ may provide practical benefits such as reduced computation time, and an obstacle to an adversary in determining exactly which of the common keys have been hashed to create the key for a given link.

Correctly calculating resilience is important for accurately assessing and comparing KPSs; in particular, comparisons are often drawn between the performances of random and deterministic key predistribution schemes. It is there-

fore reassuring to know that the original equation in [20] produces probabilities which are similar to those given by the correct formula for fail_s. However, establishing the correct formula is of mathematical importance, and expressing it in a way which is computable is of practical importance.

Chapter 4

Expander graphs and key predistribution schemes

Contents

The content of this chapter is largely based on the paper [42].

4.1 Introduction

Since networks may be modelled as graphs, tools from graph theory have been used in both their design and analysis. In this chapter, we explore the role of expander graphs in KPSs. The expansion of a graph is a measure of how well connected it is, and how difficult it is to separate subsets of vertices; we will see the precise definition in Section 4.2. Roughly speaking, a graph has good expansion if every 'small' subset of vertices has a 'large' neighbourhood, and intuitively, expansion is a desirable property for graphs of networks. The term 'expander graphs' is used informally to refer to graphs with good expansion.

In 2006, expander graph theory was introduced to the study of KPSs from two perspectives. On the one hand, Çamtepe et al. [18] showed that a mathematical construction for an expander graph could be used to design a KPS, resulting in a network which is well connected under certain constraints. On the other hand, Ghosh [36] claimed that good expansion is a necessary condition for 'optimal' networks. We examine these claims and determine the role of expander graphs in KPSs for resource-constrained networks.

We show that constructions for KPSs based on expander graphs provide perfect resilience, but lower connectivity and expansion than many existing comparable KPSs. We argue that expansion is an important metric for assessing KPSs to be used alongside the other common metrics of key storage, connectivity and resilience for a given network size. However, we note the difficulty of finding the expansion coefficient of a graph and so propose estimating the expansion and using other graph-theoretical techniques to indicate potential weaknesses.

We begin by introducing expander graphs and the relevant terminology in Section 4.2. In Section 4.3 we outline Ghosh's claims and show by means of a counter-example that his conclusion is misdirected towards expansion in product graphs rather than intersection graphs. In Section 4.4 we discuss how to maximise the probability of a high expansion coefficient in the intersection graph, and in Section 4.5 we analyse the extent to which KPSs based on expander graph constructions achieve this, in comparison to other schemes from the literature. Finally, in Section 4.6 we suggest practical metrics for analysing and improving KPSs and the resulting intersection graphs.

4.2 Expander graphs

For a thorough survey of expander graphs and their applications, see [25, 39]. Here we introduce only the aspects of expander graphs which are relevant to our study, and in particular we restrict our attention to finite graphs.

4.2.1 Boundary properties and expansion

We begin with the definition of a subgraph, before defining boundary properties and isoperimetric inequalities. Definitions 4.2 and 4.3 are reproduced from [10, Chapter 16], to which the reader is referred for further details and examples.

Definition 4.1. A *subgraph* of a graph $G = (V, E)$ is a graph $G_S = (V_S, E_S)$ in which $V_S \subseteq V$ and $E_S \subseteq E$. If V_S or E_S is a proper subset (that is, $V_S \neq V$ or $E_S \neq E$), then the subgraph is a *proper subgraph* of G. If V_S or E_S is empty, the subgraph is called the *null graph*. A *vertex-induced subgraph* is a subset

$S \subseteq V$ together with the edge set $E_S := \{(N_i, N_j) \in E : N_i, N_j \in S\}$.

We can consider *boundary properties* of subgraphs. Two intuitive examples of boundary properties are the edge boundary and the vertex boundary:

Definition 4.2. For a graph $G = (V, E)$ and a vertex-induced subgraph on $S \subset V$, the *edge boundary*, denoted $E(S, \overline{S})$, is defined to be the set of edges incident to both a vertex in S and a vertex in $\overline{S} = G \setminus S$. Similarly, the *vertex boundary* is the set of vertices in $\overline{S}$ adjacent to at least one vertex in S, denoted by δS.

Definition 4.3. An *isoperimetric inequality* in a graph is a lower bound on the size of a boundary, in terms of the size of the subgraph. For example, an isoperimetric inequality for the edge boundary would be an explicit expression for

$$\min\{|E(S, \overline{S})| : S \subset G, |S| = i\}$$

for some number of interest $i \in \mathbb{N}$.

We are now ready to define the *expansion* of a graph:

Definition 4.4. A finite graph $G = (V, E)$ is an ϵ-*edge expander graph*, where the edge-expansion coefficient ϵ is defined by

$$\epsilon = \min_{S \subset V : |S| \leq \frac{v}{2}} \left(\frac{|E(S, \overline{S})|}{|S|} \right) .$$

We will explore this definition and the significance of the edge-expansion coefficient in Section 4.2.2. First we make some remarks about the nomenclature of expansion, which is not consistent throughout the literature.

Remark 4.1. *There is a related definition of an ϵ-vertex expander graph, which is defined in the same way but using the vertex boundary for the isoperimetric inequality. We have chosen to restrict our study here to edge expansion, as it is perhaps more intuitive for analysing key sharing, and is the measure used in the related literature on KPSs. Since we will be consistently referring to edge expansion properties, we will omit the word 'edge' when the context is clear, for ease of notation.*

Remark 4.2. *Although the expansion coefficient ϵ is defined for any graph, the phrase 'expander graph' is used informally to refer to graphs with good expansion, that is, graphs with a high value of ϵ, as we explain in Section 4.2.2. Definitions for ϵ vary slightly across the literature, in particular some definitions use the strict inequality $|S| < \frac{v}{2}$. Another name for the edge-expansion coefficient is the* isoperimetric number*, which is closely related to the* algebraic connectivity*, and in a weighted graph where every vertex has the same weight, ϵ is equivalent to the* Cheeger constant*; see [22] for further details.*

4.2.2 The implications of large ϵ for networks

We now explore what the definition of the edge-expansion coefficient ϵ means, and why a high value of ϵ is desirable, through the following observations:

- If $\epsilon = 0$ then we see from the definition that there exists a subset of vertices $S \subset V$ which is disconnected from the rest of the graph, ie. $E(S, \overline{S}) = \emptyset$. This implies that the graph is not connected.
- A graph is connected if and only if $\epsilon > 0$ (see proof of Lemma 4.1), hence

all connected graphs are ϵ-expander graphs for some positive value of ϵ.

- If ϵ is 'small', for example $\epsilon = \frac{1}{100}$, then there exists a set of vertices S which is only connected to the rest of the graph by one edge per 100 nodes in S. This is undesirable for a resource-constrained network for the following reasons:
  - The set S is vulnerable to being 'cut off' from the rest of the network by a small number of attacks or faults. If S contains $c \times 100$ nodes then there are only c edges between S and $\overline{S}$. A small number of compromises or failures amongst the nodes incident to these edges (of which there can be no more than $2c$) will render all communication between S and $\overline{S}$ insecure.
  - Since S is connected to the rest of the network by comparatively few edges, a higher communication burden is placed on the small set of $\leq 2c$ nodes, since a higher proportion of data needs to be routed through them. This will drain the batteries of the nodes nearest to the edges between S and $\overline{S}$ faster than those of an average node, so that after some period of time they will run out of energy, disconnecting S from the rest of the network even though many nodes in S may still have battery power remaining.
  - Reliance on a small number of edges to connect large sets of nodes may create bottlenecks in the transmission of data through the network, making data collection and/or aggregation less efficient.
- If ϵ is larger, particularly if $\epsilon > 1$, then there is no 'easy' way to disconnect large sets of nodes, and there is a more even spread of communication

burdens, battery usage and data flow.

We see from these observations that intersection graphs with higher values of ϵ are more desirable for resource-constrained networks. A graph with a 'large' value of ϵ is often said to have 'good expansion'. The size of ϵ is subject to the following bounds.

Lemma 4.1. *For any connected graph $G = (V, E)$ with $|V| \geq 2$,*

$$0 < \epsilon \leq \min_{x \in V} d(x) \ .$$

Proof. We begin by considering the lower bound. Suppose for a contradiction that $\epsilon = 0$. Then there exists a set $S \subset V$ such $|E(S, \overline{S})| = 0$. This contradicts the fact that G is connected. Since ϵ cannot be negative, we have that $\epsilon > 0$.

For the upper bound, consider the set $S = \{x\}$ where $x \in V$. It is clear that $|E(S, \overline{S})| = d(x)$, where $d(x)$ is the degree of x as defined in Section 2.3.1, and so $\frac{|E(S,\overline{S})|}{|S|} = \frac{d(x)}{1} = d(x)$. Since the definition of the edge-expansion coefficient ϵ uses the *minimum* value over all $S \subset V$ with $|S| \leq \frac{|V|}{2}$, we have that $\epsilon \leq \min_{x \in V} d(x)$. □

In addition to the observations made above, graphs with good expansion also have low diameter, logarithmic in the size of the network [39] and contain multiple short, disjoint paths between nodes [44], the many benefits of which we discussed in Section 2.4.4. These properties mean that key graphs with good expansion are particularly desirable for resource-constrained networks.

The papers by Çamtepe et al. [18] and Shafiei et al. [60] propose KPSs based on expander graph constructions. These methods of designing a KPS ensure that the key graph has good expansion, and we further examine these proposals in Section 4.5. Before we consider the claims made by Ghosh in [36] about the necessity of good expansion for 'optimal' networks, we introduce one more definition of expansion, to which we will refer briefly in Section 4.5, and which we will build upon in Chapter 5.

4.2.3 Spectral expansion

For a d-regular graph G, we define the *spectral expansion* of G using linear algebra. The following definitions are compiled with reference to [2, 22, 25, 39].

Definition 4.5. The *adjacency matrix* of a graph $G = (V, E)$ is defined to be a $|V| \times |V|$ matrix A, where each entry a_{ij} is the number of edges incident to both vertex i and vertex j.

It is easy to see that the adjacency matrix of a simple graph is a symmetric $0-1$ matrix, with zeroes on the main diagonal. For a d-regular graph, the sum of the entries in each row or column is d. Since A is a real symmetric $v \times v$ matrix, it has $|V| = v$ real eigenvalues, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_v$, where all $\lambda_i \in [-d, d]$. In fact, it can be shown that $\lambda_1 = d$, corresponding to eigenvector u_1 whose entries are all $\frac{1}{v}$. We also note that $\lambda_v = -d$ if and only if G is bipartite (two-colourable). We can now define the spectral gap as follows.

Definition 4.6. Let $\widetilde{\lambda}$ be the largest eigenvalue in absolute value with $|\widetilde{\lambda}| \neq d$.

Then the *spectral gap* of a d-regular graph G is defined to be $\lambda_1 - \widetilde{\lambda} =$

$$d - \widetilde{\lambda} \ .$$

The various definitions of expansion are closely related. Tanner [65] and independently Alon and Milman [1] proved that

$$\frac{d - \widetilde{\lambda}}{2} \leq \epsilon \leq \sqrt{2d(d - \widetilde{\lambda})} \tag{4.2.1}$$

where ϵ is the edge expansion coefficient defined in Definition 4.4. Thus, the spectral gap may be used as a measure of the *spectral expansion*: the larger the spectral gap, the better the expansion of G.

Finally, Alon and Boppana [55] proved that all large d-regular graphs satisfy $\widetilde{\lambda} \geq 2\sqrt{d-1} - o(1)$. A graph is said to be Ramanujan if this bound is tight, that is, if $\widetilde{\lambda} \leq 2\sqrt{d-1}$, and therefore Ramanujan graphs have asymptotically smallest possible $\widetilde{\lambda}$, making them very good spectral expanders [25].

4.3 Expansion in product graphs

In [36] Ghosh considers KPSs with large network size, low key storage per node, high connectivity and high resilience. He considers jointly 'optimising' these parameters, although exactly what this means is unclear, since different applications will prioritise them differently. Nevertheless, he argues that if a KPS is in some sense 'optimal', the product graph (Definition 4.7) of the key graph and communication graph must have 'good expansion properties'. We show by a counterexample that expansion in the product graph is not a helpful

measure because the product graph is almost inevitably an expander graph. Additionally, we show that the product graph is unable to capture the required detail to analyse a network, and that it is the intersection graph where such analysis is relevant.

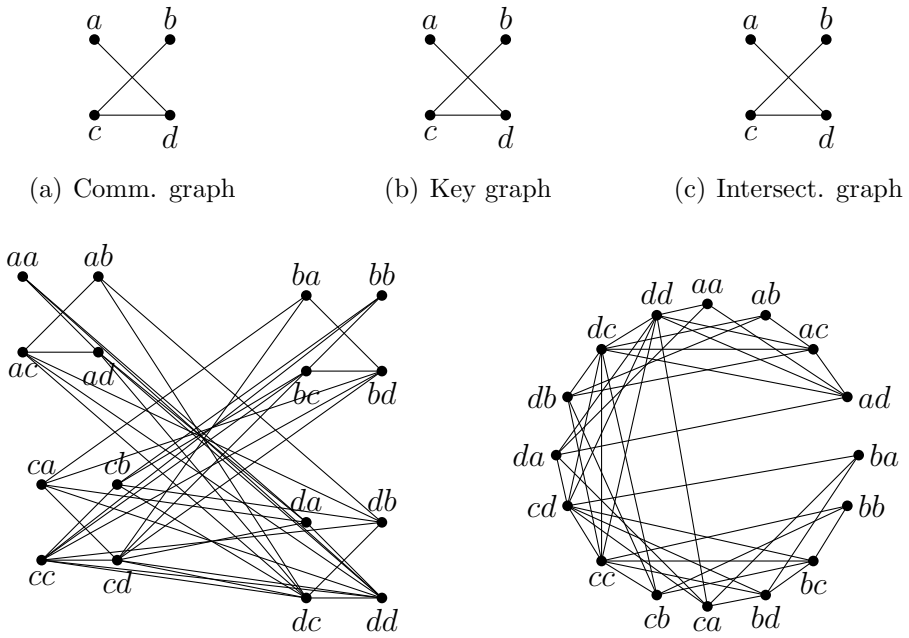
First, we define what is meant by the product graph in this context.

Definition 4.7. The (Cartesian) *product graph* of two graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$ is defined as $G.H = (V_G \times V_H, E_{G.H})$, where the set of edges $E_{G.H}$ is defined in the following way: for vertices $xy, x'y' \in V_G \times V_H$ with $xy \neq x'y'$, we have $(xy, x'y') \in E_{G.H}$ if

$$(x = x' \text{ or } (x, x') \in E_G) \quad \text{and} \quad (y = y' \text{ or } (y, y') \in E_H) \ .$$

The definition is perhaps easier to understand through examples. In Figures 4.1 and 4.2 we consider examples of product graphs and examine how they relate to their constituent communication and key graphs. Figure 4.1 shows a communication and a key graph, and their corresponding intersection and product graphs. The product graph is represented in Figure 4.1(d) in a way which demonstrates its construction, and redrawn in Figure 4.1(e) for clarity.

Figure 4.1(d) illustrates that the product graph construction results in four copies of the key graph, connected to each other in a way which resembles a large copy of the communication graph. We see that there is an edge in the product graph $(ac, ab) \in E_{G.H}$ because $a = a$ and $(c, b) \in E_H$. Similarly, $(ca, ba) \in E_{G.H}$ because $(c, b) \in E_G$ and $a = a$. However, we find that $(aa, ab) \notin E_{G.H}$ because whilst $a = a$, $(a, b) \notin E_H$.



(a) Comm. graph (b) Key graph (c) Intersect. graph

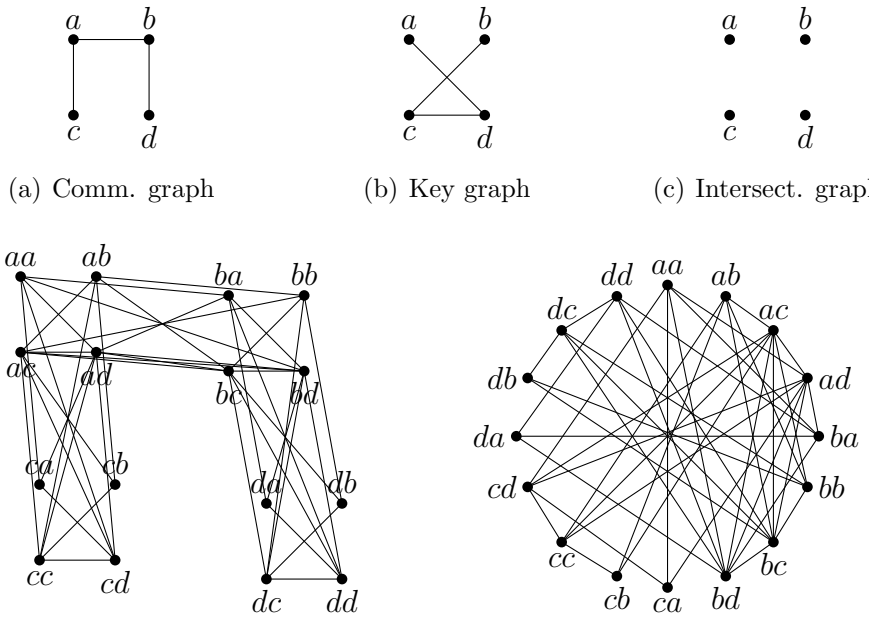
(d) Product graph (e) Product graph re-drawn

Figure 4.1: A product graph corresponding to an identical communication and key graph pair.

In Figure 4.1 the communication and key graphs are identical, giving the best possible case for intersection. We now calculate the expansion coefficient of the product graph. Consider sets S of $1, 2, \ldots, 8$ vertices (recall from the definition that we should consider subsets S with $|S| \leq \frac{|V|}{2}$, and here $|V| = 16$). We observe that any single vertex is connected to the rest of the graph by at least three edges, any set of two vertices is connected to the rest of the graph by at least six edges, etc., so that

$$\epsilon = \min \left\{ \frac{3}{1}, \frac{6}{2}, \frac{9}{3}, \frac{9}{4}, \frac{11}{5}, \frac{16}{6}, \frac{12}{7}, \frac{10}{8} \right\} \quad .$$

That is, $\epsilon = \frac{10}{8} = \frac{5}{4}$, so the product graph of Figure 4.1 has expansion coefficient $\epsilon = \frac{5}{4}$.



(a) Comm. graph (b) Key graph (c) Intersect. graph

(d) Product graph (e) Product graph re-drawn

Figure 4.2: A product graph corresponding to a communication and key graph pair with empty intersection.

Now consider Figure 4.2, where we have retained the same key graph but taken the complement of the communication graph. Thus, the communication graph has the same number of edges as that in Figure 4.1(a) but the intersection graph, shown in Figure 4.2(c), has no edges. Clearly, if this were to represent a network, it would mean that no secure communication would be possible.

However, the product graph does have edges, and indeed appears well connected. By observation, we find that it too has expansion coefficient $\epsilon = \frac{5}{4}$. Indeed, after some inspection, we find that the product graphs of Figures 4.1 and 4.2 are isomorphic, using a simple bijection to relabel vertices as follows:

Figure 4.1(e)		Figure 4.2(e)
$(a*)$	$\rightarrow$	$(c*)$
$(b*)$	$\rightarrow$	$(d*)$
$(c*)$	$\rightarrow$	$(b*)$
$(d*)$	$\rightarrow$	$(a*)$

This means that all graph-theoretic properties of connectivity, expansion, degree, diameter etc. are identical between the two product graphs. From this we see that a product graph with good expansion can occur when the key and communication graphs intersect 'fully', i.e. when $E_G \cap E_H = E_G = E_H$, and when there are no edges in the intersection, i.e. $E_G \cap E_H = \emptyset$. This shows that the expansion of the product graph certainly does not correspond to any degree of 'optimality' regarding the intersection graph and therefore the resulting network. In particular, it strongly suggests that expansion in the product graph is not a good tool for analysing the connectivity of networks without reference to the intersection graph. Ghosh's claim that an 'optimal' combination of key and communication graph will result in a product graph with good expansion tells us very little, since good expansion in the product graph is almost inevitable, as we will now explain.

Lemma 4.2. *A (Cartesian) product graph $G.H = (V_G \times V_H, E_{G.H})$ is connected if and only if both $G = (V_G, E_G)$ and $H = (V_H, E_H)$ are connected.*

Proof. Suppose that the product graph is connected. We want to show that for any pair of vertices $x_i, x_j \in V_G$ there exists a path from x_i to x_j, and similarly that there exists a path between every pair of vertices $y_i, y_j \in V_H$.

Pick a pair of vertices $x_{p1}, x_{pn} \in V_G$, fix an arbitrary vertex $y_1 \in V_H$ and consider $x_{p1}y_1, x_{pn}y_1 \in V_G \times V_H$. Since $G.H$ is connected, there exists a path from $x_{p1}y_1$ to $x_{pn}y_1$, say,

$$(x_{p1}y_1, x_{p2}y_1), (x_{p2}y_1, x_{p3}y_1), \ldots, (x_{p(n-1)}y_1, x_{pn}y_1) \ .$$

By the definition of the product graph, this means that either $x_{pi} = x_{p(i+1)}$ or

$(x_{pi}, x_{p(i+1)}) \in E_G$ for each $1 \leq i \leq n$. Thus we have found a path from x_{p1} to x_{pn} in G, and so G is connected. Similarly, H is connected.

Now suppose that G and H are each connected. We can construct a (not necessarily shortest) path between arbitrary vertices $x_{p1}y_{q1}, x_{pn}y_{qm} \in V_G \times V_H$ in the following way. There is a path in G from x_{p1} to x_{pn}, say

$$(x_{p1}, x_{p2}), (x_{p2}, x_{p3}), \ldots, (x_{p(n-1)}, x_{pn}) \ .$$

Then by definition, the following is a path in $G.H$:

$$(x_{p1}y_{q1}, x_{p2}y_{q1}), (x_{p2}y_{q1}, x_{p3}y_{q1}), \ldots, (x_{p(n-1)}y_{q1}, x_{pn}y_{q1}) \ . \tag{4.3.1}$$

Similarly, using a path $(y_{q1}, y_{q2}), (y_{q2}, y_{q3}), \ldots, (y_{q(m-1)}, y_{qm})$ in H, we have a path in $G.H$:

$$(x_{pn}y_{q1}, x_{pn}y_{q2}), (x_{pn}y_{q2}, x_{pn}y_{q3}), \ldots, (x_{pn}y_{q(m-1)}, x_{pn}y_{qm}) \ . \tag{4.3.2}$$

Since the path (4.3.2) begins at the vertex where path (4.3.1) ends, we can concatenate them to give a path from $x_{p1}y_{q1}$ to $x_{pn}y_{qm}$ in $G.H$.

□

Corollary 4.3. *If G and H are connected, the product graph $G.H$ has expansion coefficient $\epsilon_{G.H} > 0$.*

Proof. Recall from Lemma 4.1 that a connected graph is an expander graph for some value of ϵ. Therefore, if G and H are connected, the product graph will be an expander graph for some value of $\epsilon_{G.H} > 0$. □

We conjecture that with high probability, $\epsilon_{G.H} > \epsilon_G, \epsilon_H$ and $\epsilon_{G.H} \gg 0$. We justify this by considering the comparatively large degrees of nodes in the product graph, and the product graph's similarity to an expander graph construction.

For any node $xy \in V_G \times V_H$ with degrees $d_G(x), d_H(y)$ in the communication and key graphs respectively, we can compute its degree in the product graph as

$$d_{G.H}(xy) = d_G(x)d_H(y) + d_G(x) + d_H(y) \ . \tag{4.3.3}$$

Using Lemma 4.1, we have that

$$\epsilon_{G.H} \leq \min_{xy \in V_G \times V_H} (d_G(x)d_H(y) + d_G(x) + d_H(y)) \ ,$$

a much higher bound than for the constituent graphs. Since, on average, vertices of the product graph have higher degree than vertices in the constituent graphs, and since the construction of the product graph makes 'isolated' sets of vertices extremely unlikely, we see that $\epsilon_{G.H}$ is likely to be large, and in particular greater than either of ϵ_G and ϵ_H. By comparison, the expansion coefficient of the intersection graph $\epsilon_{G \cap H}$ is forced to be no more than those of the constituent graphs, ϵ_G and ϵ_H, as explained in the next section.

Additionally, the construction of the product graph is not dissimilar to that of the *zig-zag product* graph presented in [58] as an expander graph construction, and used by Shafiei et al in [60] to produce key graphs with good expansion. We see then that expansion in the product graph is inevitable if the constituent graphs are connected, is likely to be 'good', and does not imply anything about the quality of the connectivity or expansion in the intersection graph, where it is needed. Ghosh does not justify his choice of using the product graph as a

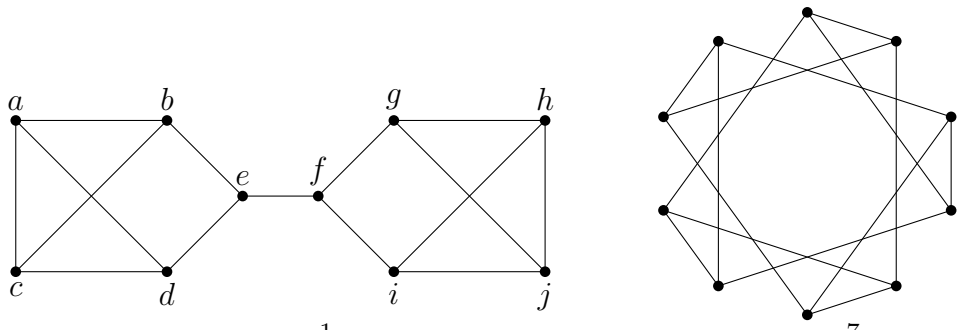
means of studying two graphs simultaneously, and we conclude that there are no benefits to doing so. In order to capture the relevant interaction between the key and communication graphs, the intersection graph is the relevant tool, and it is in the intersection graph where good expansion is desired.

4.4 Expansion in intersection graphs

We claim that when comparing two networks of the same size with identical key storage, connectivity and resilience parameters, the network represented by the intersection graph with higher expansion will be the more robust, with a more evenly distributed flow of data. We justify this using the following example.

Example 4.1. *Consider Figure 4.3 and suppose that these are two intersection graphs, representing different networks. Each graph is 3-regular on 10 nodes. We suppose that an Eschenauer Gligor KPS [32] (Scheme 2.1 in Section 2.4.1) has been used to construct the key graph, where each node stores three keys chosen randomly from a pool of 25 keys. Recall from Chapter 3 that* $\Omega = 1$*, that is, where nodes have more than one key in common, they select just one of them as their link key.*

Using Lemma 3.1, we find that $\mathsf{Pr}_1 = 1 - \frac{\binom{22}{3}}{\binom{25}{3}} \approx 0.33$*, and from Lemma 3.2 we have* $\mathsf{fail}_1 = \frac{3}{25}$ *and* $\mathsf{fail}_s = 1 - \left(1 - \frac{3}{25}\right)^s$ *for both graphs. We observe that in Figure 4.3(a) the expansion is* $\epsilon = \frac{1}{5}$*. This minimum value is achieved by (for example) picking the set of 5 vertices* $S = \{a, b, c, d, e\}$*, which is only connected to the rest of the graph by the single edge* (e, f)*. However, in Figure 4.3(b) we*



(a) $\epsilon = \frac{1}{5}$ (b) $\epsilon = \frac{7}{5}$

Figure 4.3: Examples of 3-regular graphs on 10 nodes with different expansion parameters.

find that $\epsilon = \frac{7}{5}$; any set of 5 vertices is connected by at least 7 edges to the rest of the graph.

For resource-constrained applications, the network represented by Figure 4.3(a) is less desirable, because:

- *it is more vulnerable to a listening adversary, who could decrypt a high proportion of communications through the network by the compromise of a single node e or f;*
- *nodes e and f are more vulnerable to battery failure;*
- *a battery failure or other problem on just one of the two nodes e and f would disconnect the network;*
- *communication bottlenecks are likely to occur around nodes e and f, making communication through the network less efficient.*

Conversely, in Figure 4.3(b) the communication burdens are distributed evenly

across the nodes so that battery power will be used more evenly and there are no weak spots for an adversary to target in order to quickly damage the rest of the network. The graph can only be split into disjoint sets by the removal of 4 or more nodes, that is, almost half of the network. It is clear then that Figure 4.3(b) represents the less vulnerable network.

From this example we see that some strengths and weaknesses of the 'layout' of the network are hidden if we only consider the size, key storage, connectivity and resilience, and in Section 4.6 we discuss the practicality of using expansion as another metric for assessing networks. Before that, we consider how best to probabilistically maximise the expansion in an intersection graph.

Lemma 4.4. *The expansion of an intersection graph $G \cap H = (V, E_G \cap E_H)$ is bounded above:*

$$\epsilon_{G\cap H} \leq \min\{\epsilon_G, \epsilon_H\} \ .$$

Proof. We begin by considering the degree of a vertex in the intersection graph, which is

$$d_{G\cap H}(x) \leq \min\left(d_G(x), d_H(x)\right)$$

because for each vertex $y \in V$ adjacent to x, the edge $(x, y) \in E_G$ will be removed in the intersection unless $(x, y) \in E_H$ also. Using Lemma 4.1, we have that $\epsilon_{G\cap H} \leq \min_{x\in V}\{d_{G\cap H}(x)\}$.

Without loss of generality, suppose that $\epsilon_G \leq \epsilon_H$. Consider a set S of vertices in G which achieves the minimum $\frac{|E(S,\overline{S})|}{|S|} = \epsilon_G$. If every edge of $E(S, \overline{S})$ remains in the intersection then $\epsilon_{G\cap H} \leq \epsilon_G$, otherwise $\epsilon_{G\cap H} < \epsilon_G$, since no edges are added elsewhere in the intersection. □

We see that it is necessary that G and H have high expansion coefficients for $G \cap H$ to be a good expander. If the communication graph is complete then the expansion of the key graph will be preserved in the intersection. If information about the locations of the nodes is known a priori or if there is some control over the communication graph, then keys can be assigned to nodes in a more efficient manner; see [52] for a survey of KPSs for such scenarios.

However, we usually assume that there is little or no control over the communication graph and model it using a random graph such as the random geometric model, as in [18]. If the communication graph is random, all that can be done to aid good expansion in the intersection graph is to design the KPS so that the key graph has as high expansion as possible for a particular network size and for given levels of key storage, connectivity and resilience.

4.5 Analysing the expansion of existing KPSs

Many KPSs produce key graphs with large expansion coefficients for chosen levels of key storage and resilience, as we will now demonstrate. In particular, we will discuss and compare the expansion of KPSs based on expander graph constructions (Section 4.5.1), random KPSs (Section 4.5.2) and KPSs based on combinatorial designs (Section 4.5.3).

4.5.1 KPSs based on expander graph constructions

Çamtepe et al. [18] and Shafiei et al. [60] propose KPSs based on expander graph constructions and demonstrate that these schemes compare favourably to other well-regarded KPS approaches. Çamtepe et al. [18] use a construction for a Ramanujan graph, which, as we saw in Section 4.2, is an asymptotically optimal spectral expander graph. The construction they use is for network size $v = p + 1$ and key storage $k = q + 1$, where p and q are primes congruent to 1 mod 4 (see [39]). Shafiei et al. [60] use the zig-zag construction for an expander graph, which has the benefit of being more flexible to produce key graphs for any sizes of v and k. Both papers use the following method:

1. construct an expander graph G for the appropriate network size and degree (and, in the case of [18], remove any self-loops or multiple edges. The authors suggest that these be replaced with randomly-selected edges such that all nodes have the same degree, though they omit this step from their example.);
2. assign a unique pairwise key to every edge of G;
3. preload each node with the set of keys which correspond to its set of edges.

The key graph then has good expansion. However, we claim that it is possible to achieve higher expansion in a KPS for the same network size and key storage, as we will now demonstrate.

4.5.2 Random KPSs

Random graphs are good expanders with high probability [39]. As noted in Remark 3.1, the random KPS of Eschenauer and Gligor [32] gives a key graph which is more highly connected than the Erdös-Rényi graph $G(v, \mathsf{Pr}_1)$ [31]. It therefore seems likely that this and other random KPSs (discussed in Section 2.4.1 and Chapter 3) produce key graphs with good expansion.

With the exception of the random pairwise scheme (Scheme 2.3) the key graphs of these random KPSs are also likely to be *better* expanders than those based on expander graph constructions, since for comparable key storage, random KPSs have larger average node degree. That is, in the KPSs based on expander graph constructions, the node degree is the same as the key storage because unique pairwise keys are used. In random KPSs we usually expect that $d(N_i) > k$ for all vertices N_i, as illustrated by the following example.

Example 4.2. *In the Eschenauer Gligor random KPS [32], the degree $d(N_i)$ of each node $N_i \in V$ in the key graph is almost certainly larger than the key storage k. For example, if nodes store 50 keys randomly selected from a pool of 1000 keys, then the expected degree of any node is*

$$(v-1) \times \left(1 - \frac{\binom{950}{50}}{\binom{1000}{50}}\right) \quad .$$

If the network has 1000 nodes, this means that the expected degree is ≈ 71.905. This implies that for practical values of k, n and v, the connectivity Pr_1 is greater in the Eschenauer Gligor scheme than in KPSs produced by expander graph constructions. Random graphs are known to be good expanders with

high probability and so, perhaps contrary to intuition, a key graph based on an expander graph construction is likely to be a worse expander than a key graph generated by the Eschenauer Gligor scheme.

It is clear that a benefit of the KPSs based on expander graph constructions is that each key is only used for one edge, meaning that the graphs have perfect resilience: $\mathsf{fail}_s = 0$ for all $1 \leq s \leq v - 2$. Therefore, in comparison to other KPSs with perfect resilience, the KPSs based on expander graphs have an additional benefit of guaranteeing a known expansion parameter. However, where perfect resilience is not required, for fixed values of k, n and v, random KPSs exist with slightly lower resilience but much higher connectivity than the schemes based on expander graphs from [18] and [60], and in many cases they also have comparable, if not better, expansion.

Similarly, most schemes based on combinatorial designs also reuse keys so that $d(N_i) > k$, and therefore produce key graphs with higher average degree than those based on expander graph constructions. We will now demonstrate that the key graphs of many combinatorial designs are also likely to be good expanders.

4.5.3 Combinatorial designs

Many combinatorial designs have been suggested for use as KPSs [56]. We will now analyse a subset of these constructions and show that they provide good expansion in the key graph, which is a previously unstated benefit of the combinatorial design approach to KPSs.

4.5.3.1 Finding the expansion of a design

Given any particular design, we can find the adjacency matrix of the corresponding key graph, and hence its spectral gap, which provides bounds on the expansion coefficient. We may even be able to find ϵ by inspection if v is small. However, we would like to analyse the expansion of designs in general.

Most of the designs proposed for use in KPSs have the property of being configurations (Definition 2.4). More information on configurations can be found in [24, 47]. We note here that the key graph of a configuration is regular:

Lemma 4.5. *(from [47, Lemma 1.1]) The key graph of a (n, v, r, k)-configuration is regular of degree $k(r-1)$. This is the maximum possible degree of a graph of a (n, v, r, k)-design.*

We restrict our study of the expansion of designs to configurations, where the regularity of the key graph will simplify some of the analysis.

The following theorem gives an estimation of the expansion of a regular, uniform configuration.

Theorem 4.6. *For an (n, v, r, k)-configuration, the edge-expansion coefficient can be estimated by*

$$\epsilon \approx \frac{k(r-1)}{2} \quad .$$

Proof. In the key graph of an (n, v, r, k)-configuration, the degree of a node is $k(r-1)$ (Lemma 4.5). Thus, for a set of nodes $S \subset V$,

$$E(S, \overline{S}) = |S|k(r-1) - 2E(S, S),$$

where $E(S,S)$ counts the number of edges whose endpoints are both in S. The factor of two is needed because the '$|S|k(r-1)$' term has counted each of these edges twice.

We now need to find an expression for $E(S,S)$. Let N_a and N_b be nodes in S, and let $\mathcal{K}_a$ and $\mathcal{K}_b$ be their respective key sets, each of size k. For simplicity of notation, label the keys in $\mathcal{K}_a$ as $K_1, \ldots, K_k$ and define random variables X_i for $1 \leq i \leq k$ so that

$$X_i = \begin{cases} 1 & \text{if key } K_i \in \mathcal{K}_b \\ 0 & \text{otherwise} \end{cases} .$$

Then $X := \sum_{i=1}^{k} X_i$ counts the number of edges between N_a and N_b.

The expected value of X_i, which we will write as $Ex[X_i]$ to avoid confusion with the edge notation, is given by

$$Ex[X_i] = \frac{r-1}{v-1} ,$$

since there are $r-1$ other nodes which know key K_i, out of a total of $v-1$ other nodes in the network. By linearity of expectation,

$$Ex[X] = kEx[X_i] = \frac{k(r-1)}{v-1} .$$

Thus the expected number of edges amongst $|S|$ nodes is

$$\binom{|S|}{2} \frac{k(r-1)}{v-1} ,$$

hence

$$\begin{aligned} Ex[E(S,\overline{S})] &= |S|k(r-1) - 2\binom{|S|}{2}\frac{k(r-1)}{v-1} \\ &= |S|k(r-1)\left[1 - \frac{|S|-1}{v-1}\right] . \end{aligned}$$

Finally, since $\epsilon = \min_{1 \leq |S| \leq \frac{v}{2}} \frac{|E(S,\overline{S})|}{|S|}$, this gives

$$\begin{aligned} \epsilon &\approx \min_{1 \leq |S| \leq \frac{v}{2}} \left\{ k(r-1) \left[1 - \frac{|S|-1}{v-1} \right] \right\} \\ &= k(r-1) \left[1 - \frac{\lfloor \frac{v}{2} \rfloor - 1}{v-1} \right] \\ &\approx \frac{k(r-1)}{2} \end{aligned}$$

for large v. □

Theorem 4.6 tells us that the *expected value* of the expansion of a configuration-based KPS is 'good', since in particular $\frac{k(r-1)}{2} > 1$ for practical values of k and r. However, Theorem 4.6 does not guarantee good expansion. As a counter example, notice that the design $(\mathcal{X}, \mathcal{B})$ where $\mathcal{X} = \{1,2,3,4,5,6\}$, $\mathcal{B} = \{\{1,2\},\{2,3\},\{1,3\},\{4,5\},\{5,6\},\{4,6\}\}$ is a $(6,6,2,2)$-configuration but it certainly does not have expansion parameter $\epsilon = 1$ as estimated by Theorem 4.6; on the contrary, it is disconnected and has expansion $\epsilon = 0$.

To successfully construct a KPS from a configuration, we clearly require the graph of the configuration to be connected. In addition to being k-uniform and r-regular, configurations which are proposed as constructions for KPSs generally have further properties which guarantee connectedness. We demonstrate examples of these properties in Sections 4.5.3.2 and 4.5.3.3, where we give a brief overview of two classes of configurations which have been proposed as constructions for KPSs, namely μ-common intersection designs and strongly regular graphs, and we present lower bounds for their expansion parameters.

4.5.3.2 μ-common intersection designs

Recall from Definition 2.5 that a KPS constructed from a μ-common intersection design has the property that any two nodes which are non-adjacent have at least μ common neighbours. In [47], the motivation given for using μ-common intersection designs as KPSs is that if two nodes N_i and N_j wish to communicate but do not share a common key, then they can communicate via 'two hops' if they share at least one common neighbour. In Section 2.4.4 we considered the various ways in which common neighbours can be beneficial. We now show that expansion provides another reason to support the choice of μ-common intersection designs for constructing KPSs.

We have already seen in Section 4.2 that a property of expander graphs is low diameter, which is beneficial for all of the protocols given in Section 2.4.4. Having good expansion and having low diameter are related concepts, and we use this to show that in general, a μ-common intersection design has a large expansion coefficient:

Lemma 4.7. *Let $G = (V, E)$ be a graph with diameter 2. Then for any subset S, where $\emptyset \neq S \subset V$,*

$$|E(S, \overline{S})| \geq \min\{|S||\overline{S}|\}.$$

Proof. Without loss of generality, suppose that $|S| \leq |\overline{S}|$. Now, suppose for a contradiction that $|E(S, \overline{S})| < \min\{|S||\overline{S}|\}$, that is, suppose $|E(S, \overline{S})| < |S|$. Then there exists a node $N_i \in S$ which is not adjacent to any node in $\overline{S}$. Denote the set of nodes adjacent to N_i by V_{N_i}. We have that $V_{N_i} \subseteq S$, and

so $E(V_{N_i}, \overline{S}) \subseteq E(S, \overline{S})$. Thus $|E(V_{N_i}, \overline{S})| \leq |E(S, \overline{S})| < |S| \leq |\overline{S}|$, and so there exists a node $N_j \in \overline{S}$ which is not adjacent to any node in V_{N_i}. This contradicts the property that the graph has diameter 2, since N_i and N_j do not have a common neighbour. □

Corollary 4.8. *The graph $G = (V, E)$ of a μ-common intersection design is an ϵ-expander graph, where $\epsilon \geq 1$.*

Proof. Since the graph of a μ-common intersection design has diameter 2, this is a simple consequence of Lemma 4.7. By definition, the expansion coefficient is given by

$$\begin{aligned} \epsilon &= \min_{S \subset V : |S| \leq \frac{v}{2}} \left\{ \frac{|E(S, \overline{S})|}{|S|} \right\} \\ &\geq \min_{S \subset V : |S| \leq \frac{v}{2}} \left\{ \frac{|S|}{|S|} \right\} \\ &\geq 1. \end{aligned}$$

□

Therefore we have shown that μ-common intersection designs are a natural choice for KPSs, not only because of the 'two-hop paths' property mentioned in [47], but also because they are good expanders, which implies the other beneficial properties given in Section 4.2.

4.5.3.3 Strongly regular graphs

This bound also holds for strongly regular graphs (Definition 2.6), which may be regarded as a special type of μ-common intersection design. We also have

another lower bound:

Lemma 4.9. *For a connected* $(v, k(r-1), \lambda, \mu)$ *strongly regular graph,*

$$\epsilon \geq \frac{k(r-1)}{2} - \frac{\lambda - \mu + \sqrt{(\lambda-\mu)^2 + 4(k(r-1)-\mu)}}{4} \ .$$

Proof. In [12] it is shown that the non-trivial eigenvalues of a strongly regular graph are the solutions of the equation

$$x^2 - (\lambda - \mu)x + (k(r-1) - \mu) = 0 \ .$$

Thus the larger root is given by

$$\widetilde{\lambda} = \frac{\lambda - \mu + \sqrt{(\lambda-\mu)^2 + 4(k(r-1)-\mu)}}{2}$$

and, using Equation (4.2.1), we have our result. □

In Chapter 5 we will see an example of a strongly regular graph as a KPS, and we will use Lemma 4.9 to show that its expansion is in fact significantly greater than 1.

4.6 Using expansion as a metric

We have seen that for two networks with the same size, key storage, connectivity and resilience, the network represented by the intersection graph with the higher expansion coefficient is the more robust, with the more evenly distributed flow of data. Therefore we suggest that expansion is an important metric to be considered alongside the usual metrics of key storage, connectivity and resilience, when designing KPSs and assessing their suitability for use

in resource-constrained networks. However, we now state some drawbacks to the use of expansion as a metric, and explain the extent to which they can be overcome.

Difficulty of determining the expansion coefficient. Determining the expansion coefficient of a given graph is known to be co-NP-complete [7], and so testing KPSs for their expansion coefficient is not an easy task. Additionally, even if the expansion coefficient of the key graph is known, the expansion of the intersection graph will not be known a priori if the communication graph is modelled as a random graph.

Nevertheless, the spectral gap can be used to find upper and lower bounds on the expansion of the key graph (Lemma 4.2.1). If the intersection graph is known, that is, if it is possible to determine the locations of the nodes after deployment by using an online base station or GPS, its expansion coefficient could also be approximated using the spectral gap. This is likely to be relevant if post-deployment key management protocols are available such as key refreshing [4] or key redistribution (Chapter 6), for which it could be useful to know as much as possible about the vulnerability of the network. Some key management protocols are able to provide targeted improvements to specific weak areas of the network, and we explain below how best to identify such weaknesses.

Limitations of the expansion coefficient. We note that the expansion coefficient alone does not claim to fully describe the structure of the graph,

giving only a 'worst case' assessment. That is, the value of ϵ only reflects the weakest point of the graph and tells us nothing about the structure of the graph elsewhere.

For example, consider an intersection graph on v nodes which is effectively partitioned into two sets: a set of $v - 1$ nodes with good expansion, and a final node which is disconnected from the rest of the graph, as demonstrated in Figure 4.4(b). We would find that $\epsilon = 0$, and we would suspect that the graph is less than desirable for network applications.

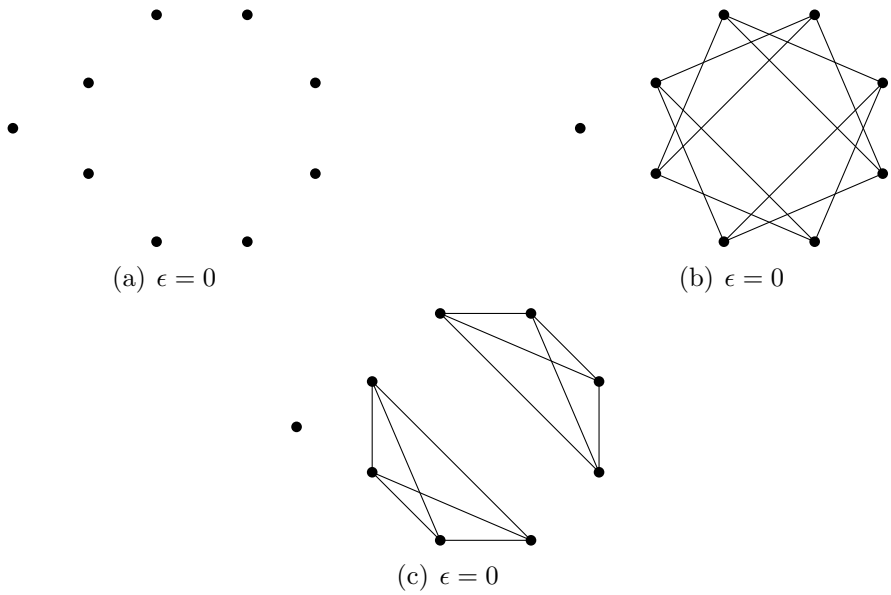


Figure 4.4: Distinguishing between cases where $\epsilon = 0$

However, particularly in a network of thousands of nodes, the disconnection of one node is unlikely to be severely detrimental to the network; indeed, loss of some nodes due to poor positioning or battery failure may be expected. Knowing only that $\epsilon = 0$ does not distinguish between the following cases:

1. the graph is completely disconnected (Figure 4.4(a));

2. a single node is disconnected from the rest of the graph, which otherwise has good expansion (Figure 4.4(b));

3. the disconnected graph is a union of smaller graphs, some with good expansion (Figure 4.4(c)).

If an intersection graph falls into Case 1 then it is likely that the key graph has low connectivity, i.e. a low value of Pr_1. However, for the same values of network size, key storage, connectivity and resilience, knowing only that $\epsilon = 0$ in the intersection graph cannot distinguish between the Cases 2 and 3, though Case 2 is likely to be much better for network applications.

Therefore, we suggest some graph-theoretic tools which also serve as indicators of whether the structure of a graph is suitable for a network. These may be used alone or in conjunction with (an estimate of) the expansion coefficient in order to analyse a proposed KPS, and where possible to analyse the resulting intersection graph.

4.6.1 Components

We note that to distinguish between the cases in Figure 4.4 it is relevant to know the number of *components*. A *component* of a graph is a connected vertex-induced subgraph containing the *maximal* number of edges [21], that is, a subset S of one or more vertices of the graph, where the vertices of S are connected but $E(S, \overline{S}) = \emptyset$. Hence Figure 4.4(a) has nine components, Figure 4.4(b) has two, and Figure 4.4(c) has three. For applications where

data must be routed throughout the network, it is desirable to minimise the number of components.

Unlike finding the expansion coefficient of a graph, calculating the number of components can be done in linear time using depth-first search, as described in [40]. The *global connectivity* of a graph is the number of nodes in its largest component divided by the total number of nodes. We wish the global connectivity to be as close to one as possible.

4.6.2 Cut-edges

A *cut-edge* (also known as a *bridge*) is an edge whose deletion increases the number of components. Equivalently, an edge is a cut-edge if it is not contained in any cycle of the graph. This is illustrated in Figure 4.3, where the edge (e, f) is a cut-edge.

As we have seen, cut-edges in the intersection graph of a network are undesirable because they can cause bottlenecks, increase communication burdens on the nodes at their endpoints, and create weak points in the network where a small fault or compromise by an adversary creates significant damage. Therefore, one of the reasons why intersection graphs with high expansion are desirable for networks is because they are less likely to have cut-edges:

- If $\epsilon > \frac{1}{\lfloor \frac{v}{2} \rfloor}$ then we know that there is no cut-edge which, if removed, would separate the graph into two components, each of size $\frac{v}{2}$ (or $\lfloor \frac{v}{2} \rfloor$ and $\lfloor \frac{v}{2} \rfloor + 1$ if v is odd).

- If $\epsilon = \frac{1}{2}$ then it is possible that there are cut-edges which, if removed, would disconnect at most two nodes from the network.

- If $\epsilon > 1$ then for all $S \subset V$ with $|S| \leq \frac{v}{2}$,

$$|E(S, \overline{S})| > |S| \geq 1 \ ,$$

  and so there can be no cut-edges in the graph.

Determining whether a graph contains cut-edges can also be achieved by a linear time algorithm [66].

4.6.3 Cutpoints

There is also a related notion of *cutpoints* in graphs; here we reproduce the definition from [21].

Definition 4.8. Consider a simple connected graph $G = (V, E)$, and a vertex-induced subgraph $G_S = (S, E_S)$ on a subset of the vertices $\emptyset \neq S \subset V$. Let $E_{\overline{S}} = E \setminus E_S$, and let $V_{\overline{S}}$ be the set of vertices which are incident to edges in $E_{\overline{S}}$. Whenever there exists such a subgraph G_S so that $|S \cap V_{\overline{S}}| = 1$, then the single node v at which they intersect is called a *cutpoint* of G. In an unconnected graph, a node is called a cutpoint if it is a cutpoint of one of its components. Since G has no self-loops, a cutpoint is a node whose removal increases the number of components by at least one.

We see then that in Figure 4.3(a), nodes e and f are cutpoints, and that graphs with good expansion will have few cutpoints. If a graph contains no cutpoints

it is said to be *nonseparable* or *biconnected*, which again is clearly desirable for an intersection graph representing a network.

The website [76] gives examples of Java algorithms which find the nonseparable components of given graphs and can even add edges to make graphs nonseparable. Tools such as this can provide simple, effective ways to analyse an intersection graph of a deployed network and indeed, wherever the post-deployment key management protocols allow, to make improvements to the structure of the intersection graph.

4.7 Conclusion

We have shown that if we fix levels of key storage, network size, connectivity and resilience, then the larger the value of the expansion coefficient ϵ in the intersection graph, the better suited it will be for resource-constrained networks. This is because graphs with good expansion are well connected with low diameter and do not have the vulnerabilities of cut-edges and cutpoints. We have shown that the expansion coefficient of the product graph is not a relevant metric; rather, it is the intersection graph where a high expansion coefficient is desirable.

In a setting where there is control over the communication graph, the expansion of the intersection graph should be an important consideration in the design of the key graph. If there is no control over the communication graph, then after choosing levels of network size, key storage, connectivity and resilience, the best choice of KPS is the one with the highest expansion, since it will maximise

the probability of achieving good expansion in the intersection graph.

The key graphs of the KPSs proposed in [18] and [60] have good expansion, and the use of unique pairwise keys gives perfect resilience. However, many existing KPSs, including random KPSs and those based on μ-common intersection designs and strongly regular graphs, are able to achieve better expansion for the same key storage and network size, at the cost of lower resilience.

Finally, we have suggested that expansion is an important metric for comparing KPSs proposed for resource-constrained networks, and a useful parameter for analysing intersection graphs after deployment in order to improve weak parts of the network. Determining the expansion of a graph is co-NP-complete and gives only a worst-case assessment of the graph. Therefore we have proposed the use of linear time algorithms to estimate the expansion, and introduced related graph-theoretic properties which could be used to analyse the key and intersection graphs of networks.

Chapter 5

Hypergraphs, expansion and KPSs

Contents

5.1 Introduction

In this chapter we discuss the role of hypergraphs in the representation and construction of KPSs. We show that there are benefits to using hypergraphs rather than graphs to represent KPSs, extend ideas from Chapter 4 to con-

struct further KPSs with good expansion, and draw connections with the literature on KPSs based on combinatorial designs.

We begin in Section 5.2 by defining hypergraphs and arguing for their use in representing KPSs, as they are able to demonstrate key storage and resilience as well as connectivity. We present motivating examples, and discuss the connection between hypergraphs and designs. In Section 5.3 we define expansion in hypergraphs, explain why hypergraphs with good expansion have advantages over graphs with good expansion for the construction of KPSs, and outline various constructions. In particular, we give an example of a Cayley hypergraph and demonstrate its effectiveness as a basis for a KPS.

5.2 Hypergraph representations of KPSs

We now define hypergraphs, which can be considered as a generalisation of graphs, where each edge may be incident to more than two vertices.

Definition 5.1. A *hypergraph* $H = (V, E)$ is a set of vertices $V = \{N_1, \ldots, N_v\}$ and a set of *hyperedges* E. A hyperedge is a subset of V of cardinality ≥ 2, written as

$$(N_{i1}, \ldots, N_{ir}) \in E$$

where $r \geq 2$. If every edge contains r vertices, we say that the hypergraph is r-uniform. Thus, a simple graph can be thought of as a 2-uniform hypergraph.

In Section 2.3.2 we introduced the use of graphs to represent KPSs, which is common in the literature. As far as we are aware, hypergraphs have not pre-

viously been used to represent KPSs. We suggest that representing a KPS by a hypergraph has advantages over a graph representation, namely by demonstrating the key storage and resilience of the KPS in addition to the connectivity, as we now explain.

5.2.1 Representing a KPS with a hypergraph

We consider again the example key graph from Section 2.3.2.1 which we replicate here in Figure 5.1(a), without the key labels. We can represent the same KPS with a hypergraph by letting each key correspond to a hyperedge, so the hyperedge is incident to all the nodes which know the corresponding key.

There are various equivalent ways to draw hyperedges: Figure 5.1(b) demonstrates perhaps the more intuitive method, where a hyperedge 'contains' its vertices; Figure 5.1(c) uses the convention of drawing a hyperedge as 'spokes' from a midpoint between the set of incident vertices. The latter will be used throughout the remainder of this chapter as it is less likely to lead to ambiguity.

Both graph and hypergraph representations demonstrate the connectivity of the KPS. The benefit of using a hypergraph representation rather than a graph representation is two-fold. Without the need to use key labels, we can deduce the key storage and the resilience of the KPS from the hypergraph using the following observations:

1. the degree of a node is equal to the number of keys stored by that node,

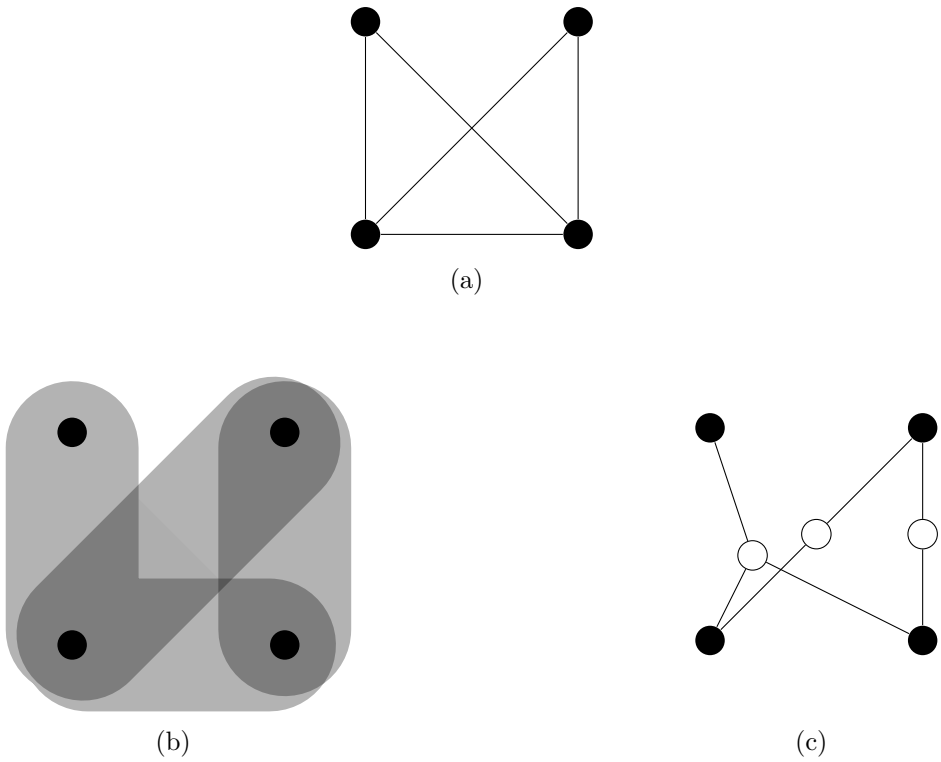


Figure 5.1: Graph and hypergraph representations of a simple KPS

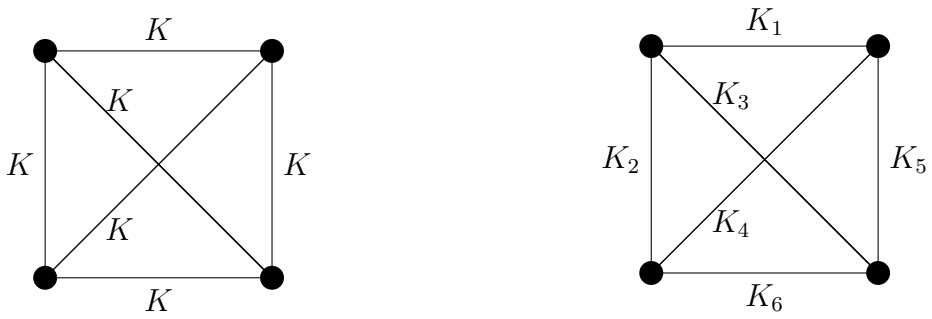
that is, $d = k$

2. the number of vertices incident to each hyperedge, r, demonstrates the number of nodes which store each key, allowing us to calculate the resilience.

Without key labels, it is impossible to determine the exact key storage and resilience from the graph representation.

5.2.2 Trivial KPS examples

To take an extreme example, consider Figure 5.2, which shows (with labels) the key graphs of two of the trivial KPS examples from Section 2.2.3 on four nodes. That is, Figure 5.2(a) is the KPS where every node stores a single key, K, and Figure 5.2(b) is the KPS where each pair of nodes is assigned a unique key. Notice that both key graphs are complete, despite the difference in their resilience. Without the edge labels (which would be infeasible to draw on a large graph) the graphs would be isomorphic, and thus the important metrics of key storage and resilience would not be represented.

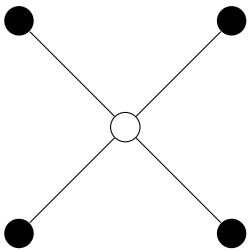


(a) Each node is assigned a single key, K

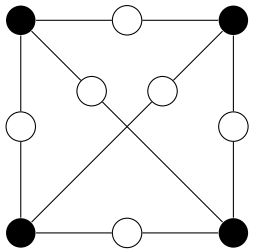
(b) Every pair of nodes is assigned a unique key

Figure 5.2: Trivial key predistribution schemes represented by graphs

By comparison, without the need for key labels, the corresponding hypergraph representations given in Figure 5.3 clearly show the key storage and the number of nodes which store each key: we observe from the nodes' degrees that $k = 1$ in Figure 5.3(a) and $k = 3$ in Figure 5.3(b), and from the magnitude of the hyperedge(s) that $r = 4$ in Figure 5.3(a) and $r = 2$ in Figure 5.3(b)), which immediately tells us that the resilience is worst possible in Figure 5.3(a) and



(a) Each node is assigned a single key, K



(b) Every pair of nodes is assigned a unique key

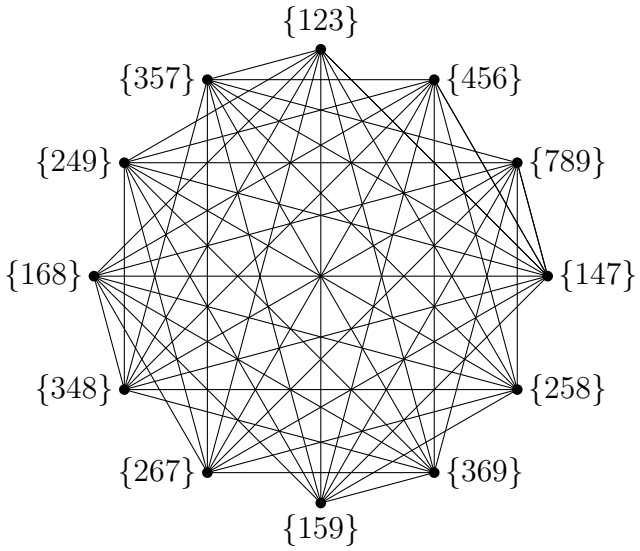
Figure 5.3: Trivial key predistribution schemes represented by hypergraphs

best possible in Figure 5.3(b).

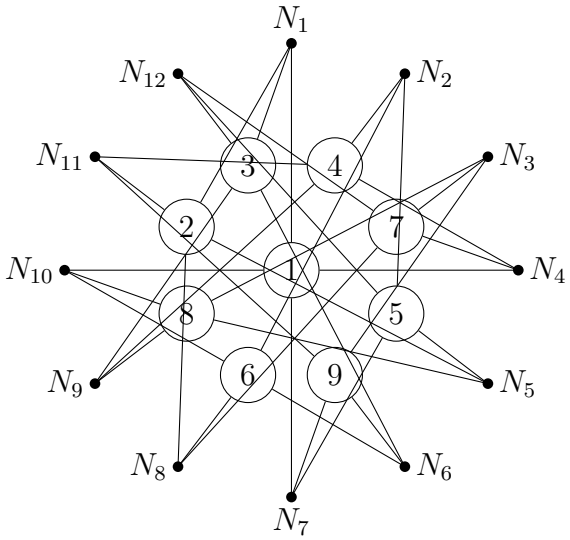
5.2.3 Design-based KPS example

We now consider another, less extreme example. Recall our $2-(9,3,1)$ design from Example 2.4 in Section 2.3.3: $\mathcal{X} = \{1,2,3,4,5,6,7,8,9\}$ and $\mathcal{B} = \{\{123\}, \{456\}, \{789\}, \{147\}, \{258\}, \{369\}, \{159\}, \{267\}, \{348\}, \{168\}, \{249\}, \{357\}\}$, which can be considered as a KPS where points represent keys and each block corresponds to a node's key set. The graph representation of this KPS is reproduced in Figure 5.4(a) and, for comparison, Figure 5.4(b) gives a hypergraph representation of the same KPS. We show in Section 5.2.4 that it is straightforward to construct a hypergraph representation of a design, and indeed that it is an intuitive representation.

Figure 5.4(a) clearly demonstrates the connectivity: we can see that each node is connected to nine other nodes, hence $\mathsf{Pr}_1 = \frac{9}{11} = 0.8181\ldots$. The labels tell us the set of keys known to each node, and from this we deduce that $k = 3$. After a little careful study, we can also observe that $r = 4$. However, without



(a) Graph representation of KPS from Example 2.4



(b) Hypergraph representation of KPS from Example 2.4

Figure 5.4: Graph and hypergraph representations of a $2-(9,3,1)$ design

the key labels it would not be possible to determine k and r with certainty. By contrast, Figure 5.4(b) demonstrates clearly that $k = 3$ (node degree) and $r = 4$ (hyperedges uniformly incident to four vertices). A simple calculation of $k(r-1) = 3 \times 3 = 9$ tells us that each node is connected to nine other nodes,

and hence the connectivity of the KPS is $\mathsf{Pr}_1 = \frac{9}{11}$. Key labels are included in Figure 5.4(b) for clarity later, but notice that they are not needed for these calculations.

In summary, without key labels, a graph only provides an exact representation of the connectivity of a KPS, whereas a hypergraph unambiguously represents the connectivity, key storage and resilience. Even with key labels, it is easier to observe the re-use of keys (the value of r) from the hypergraph representation.

5.2.4 Hypergraphs and designs

We now demonstrate a method for constructing a hypergraph representation of a design, and explain how these two combinatorial objects are closely linked by their *incidence matrices*.

Definition 5.2. The *incidence matrix* M of a design is a matrix where the columns represent blocks, rows represent points, and whose entries are given by:

$$m_{ij} = \begin{cases} 1 & \text{if point } i \text{ is in block } j \\ 0 & \text{otherwise} \end{cases} .$$

Hence, the incidence matrix of the $2-(9,3,1)$ design from Example 2.4 in

Section 2.3.3 is

$$M = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

It is then straightforward to see that key K_1 is given to nodes N_1, N_4, N_7, N_{10}, key K_2 is given to nodes N_1, N_5, N_8, N_{11}, and so on, and hence to construct a hypergraph representation, as shown in Figure 5.4(b).

Finally, we make a remark about the relationship between hypergraphs and designs, to which we will refer in Section 5.3.3.

Remark 5.1. *Since a (hyper)graph $G = (V, E)$ is defined by its vertex set V and (hyper)edge set E, a (hyper)graph is a set system. The two concepts are equivalent. In particular, a graph is a set system which is uniform of rank 2. An r-uniform hypergraph is a set system which is uniform of rank r. Further, as we now demonstrate, the hypergraph representation of a combinatorial design can be a regular, uniform design itself. The hypergraph $H = (V, E)$ from Figure 5.4(b) is given by $V = \{1, 2, \ldots, 12\}$, and*

$$\begin{aligned} E &= \{\{1,4,7,10\},\{1,5,8,11\},\{1,6,9,12\},\{2,4,9,11\},\{2,5,7,12\}, \\ &\quad \{2,6,8,10\},\{3,4,8,12\},\{3,5,9,10\},\{3,6,7,11\}\}. \end{aligned}$$

Regarding $H = (V, E)$ as a combinatorial design, that is, regarding the hyperedges as blocks, we see that this design is regular of degree 3, uniform of rank 4, and has incidence matrix M^t, the transpose of the incidence matrix M above.

5.3 Expansion in hypergraphs

In Chapter 4 we noted that good expansion is a desirable property for key predistribution schemes. That is, it is desirable to have good expansion in the intersection graph, and therefore in the key graph. However, attempts to create a key predistribution scheme from known expander graph constructions (as in [18, 60]) have provided an extreme in the trade-off between connectivity and resilience: they provide perfect resilience at the expense of low connectivity (in comparison to KPSs without perfect resilience, given fixed key storage k). Here, we show how to adapt existing hypergraph constructions which are known to provide good expansion, to create KPSs with good expansion and without perfect resilience.

5.3.1 Expansion in KPSs without perfect resilience

The constructions in [18, 60] take an expander graph construction, remove any self-loops and multi-edges, and then assign a unique key to each edge. Thus, each key is used to secure exactly one link, and so perfect resilience is achieved. However, this also means that if a node stores k keys then it only has degree k in the key graph. Other KPS constructions tend to re-use keys, such that a node which stores k keys will often have degree significantly greater than k, and hence the network has higher connectivity.

Therefore, if perfect resilience is the main priority, then these expander graph constructions are appropriate. However, if perfect resilience is not required,

then considerably higher connectivity can be achieved. We now address how to retain the focus of obtaining good expansion in a KPS without perfect resilience.

A naïve approach would be to take an existing expander graph construction for KPSs, such as those in [18, 60], and simply assign the same key to multiple edges. However, it is not entirely clear how best to do this. One way would be to create the expander graph, pick a node at random, and assign a key K_1 to two of its edges, repeat for a key K_2, etc. However, such an approach could lead to unnecessarily high key storage for some nodes, whilst others receive comparatively few keys. In addition, an expander with higher connectivity than in [18, 60] should be picked in the first place, as otherwise we are reducing the resilience whilst maintaining the same connectivity. It is not immediately obvious how many edges the initial expander construction should have in order to maintain a similar magnitude of key storage and maintain a desirable trade-off between connectivity and resilience.

Therefore, whilst this method may have merit, we demonstrate a 'tidier' solution, which harnesses the properties of hypergraphs to achieve exactly what we seek: a KPS with good expansion which has a deterministic construction, enabling detailed analysis, and where each key is known to $r > 2$ nodes. We propose that if we could find a 'nice' hypergraph with $r > 2$ and, in some sense, good connectivity and expansion, then we could still assign a unique key per edge, without resulting in having perfect resilience.

5.3.2 Constructions for hypergraphs with good expansion

The literature on hypergraphs with good expansion (sometimes called 'expanding hypergraphs') is limited. In [35], the notion of eigenvalues of a graph is extended to hypergraphs, and thus Friedman et al. are able to analyse the second eigenvalue of a hypergraph. This allows them to carry over the concept from graph theory of a spectral expander graph being one with large spectral gap (Section 4.2). They prove many theorems about the properties of such expanding hypergraphs, and provide a construction, namely Cayley hypergraphs. Whilst they demonstrate that Cayley hypergraphs have good expansion, they also note that the expansion of a general Cayley hypergraph is far from the optimal bounds which are proven in the paper.

There is currently no known method for generating uniform hypergraphs at random [39]. If there were, they would likely provide a very promising way of constructing KPSs, since it is proven in [35] that a random uniform hypergraph has good expansion with high probability.

In [57] two further approaches to constructions of hypergraphs with good expansion are given: explicit constructions from Gower's Theorem, and semi-explicit constructions from Fourier Analysis. However, in order to provide a simple demonstration of the method of constructing a KPS from a hypergraph, we will use the Cayley hypergraph construction, which has good (though far from optimal) expansion, and we will compare its effectiveness as a construction for a KPS to the construction from [18] based on a Ramanujan expander graph.

5.3.3 Cayley hypergraphs

We use the Cayley expanding hypergraph construction from [34] for our analysis because it is straightforward to define and construct, whilst noting that Cayley hypergraphs have far from optimal expansion when considering the second eigenvalue.

Definition 5.3. Let V be a group, and $W \subset V$. The 3-*uniform Cayley hypergraph on V and W* is the hypergraph whose vertices are the elements of V and whose hyperedge set is given by

$$E = \{(x, y, z) : x, y, z \text{ distinct}, xyz \in W\}.$$

Remark 5.2. *We note that, to the best of our knowledge, the earliest definition of a Cayley hypergraph is given in [34], as an extension of the widely-known definition of a Cayley graph (or 2-hypergraph). It is stated as the definition of a '3-regular' Cayley hypergraph on V and W. However, the word 'regular' is more usually used to mean that the number of (hyper)edges incident to each node is constant. This is not necessarily the case in Cayley hypergraphs (as we will demonstrate below) and indeed there will generally be more than three hyperedges incident to each node. We therefore use the word 'uniform' for consistency with the recent literature.*

In addition, for the hypergraph to be 3-uniform we require that $x \neq y \neq z \neq x$. Otherwise, hyperedges of the form (x, x, z) and (x, x, x) would be included, which are not usually considered to be 3-edges. This is not stated in the definition in [34].

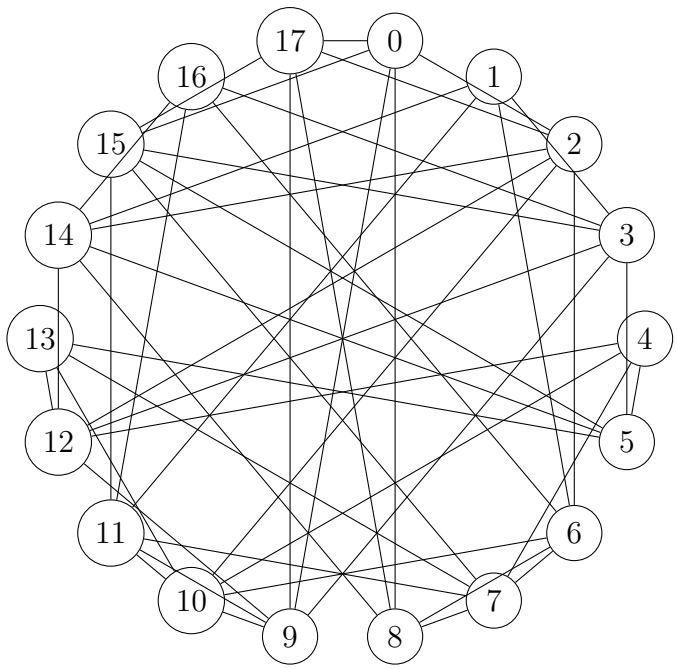


Figure 5.5: KPS from Ramanujan expander graph construction, from [18]

5.3.4 Comparing Ramanujan expander graphs and Cayley hypergraphs as constructions for KPSs

In the example given in [18], a KPS is created for 18 nodes from a Ramanujan expander graph. We reproduce their key graph in Figure 5.5.

Each node stores 4, 5 or 6 keys, and there are 46 different keys used in total, each securing exactly one link. Thus we have $\mathsf{fail}_s = 0$ for all $s \leq 16$ and

$$\mathsf{Pr}_1 = \frac{46}{\binom{18}{2}} = \frac{46}{153} \approx 0.3.$$

For a direct comparison, we now construct a KPS for 18 nodes using a 3-uniform Cayley hypergraph. The Cayley hypergraph $H = (V, E)$ where $V =$

$\{0, 1, \ldots, 17\}$ and

$$E = \{(x, y, z) : x, y, z \text{ distinct}, x + y + z \equiv 0 \mod 18\}$$

also has 46 hyperedges, but each hyperedge connects three nodes. Thus there are $46 \times 3 = 138$ 'pairs' of links, and so

$$\mathsf{Pr}_1 = \frac{138}{\binom{18}{2}} =\approx 0.902 \ ; \tag{5.3.1}$$

we have tripled the connectivity. This has not been at the cost of a large reduction in resilience:

$$\mathsf{fail}_1 = \frac{1}{3}\frac{7}{124} + \frac{2}{3}\frac{8}{122} \approx 0.0625 \tag{5.3.2}$$

because:

- One third of the nodes store 7 keys. On compromising one of these nodes, the adversary learns 7 keys. There are then $138 - (7 \times 2)$ uncompromised links remaining, of which 7 use a key known to the adversary.
- Similarly, two thirds of the nodes store 8 keys, and compromising one of these leaves $138 - (8 \times 2)$ uncompromised links, of which 8 are vulnerable.

We will provide a general formula for fail_s at the end of this section, after making a simplifying assumption.

We also note that there is a more equal spread of key storage per node in this construction than in the Ramanujan construction from [18], that is, the key storage for node N_i is $k_{N_i} \in \{7, 8\}$, whereas in the Ramanujan construction $k_{N_i} \in \{4, 5, 6\}$. However, for large networks the Cayley construction does not

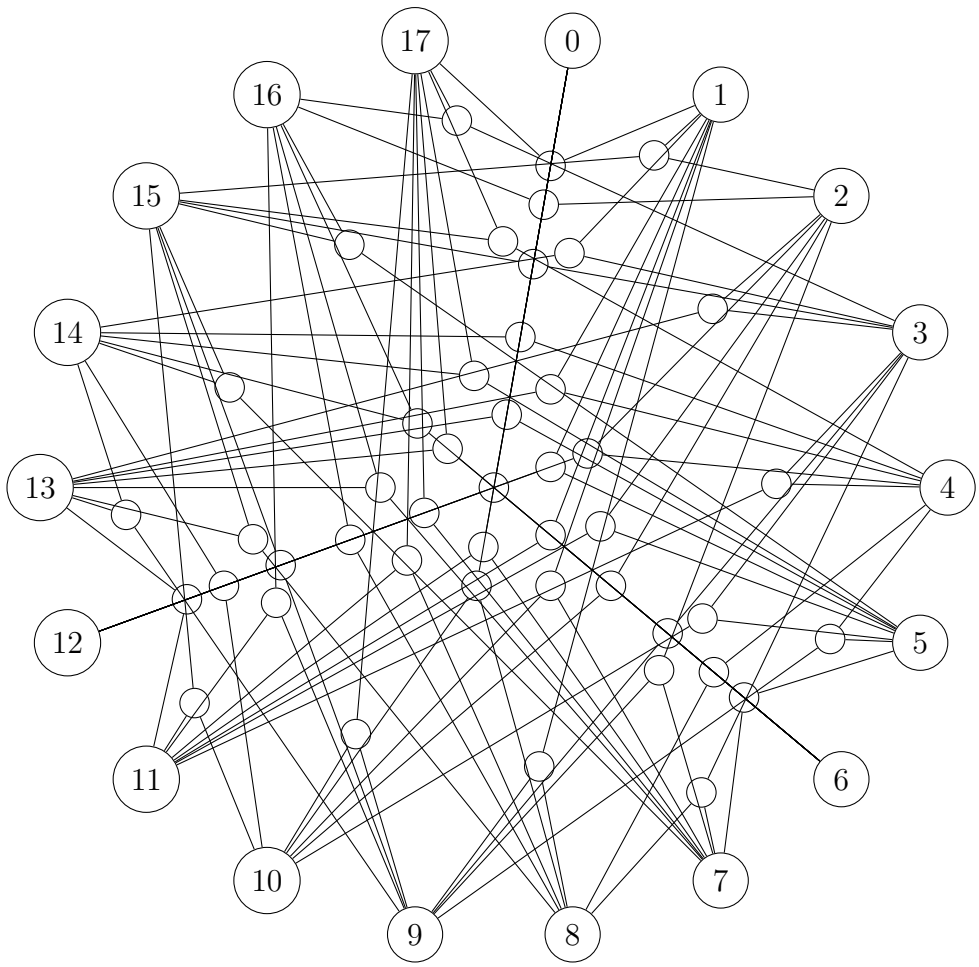


Figure 5.6: Cayley hypergraph on 18 nodes

scale well: the key storage is $\approx \frac{v-1}{2}$. Nevertheless, it serves as a simple example of the use of a hypergraph to construct a KPS.

We consider the expansion of our Cayley hypergraph. By inspection, we can find a subset of vertices S with $|S| = 9$, and $|E(S, \overline{S})| = 72$, giving

$$\epsilon \leq 8 \ . \tag{5.3.3}$$

After making some further observations we will show that, in fact, the addition of two hyperedges gives $\epsilon = 8$.

Although our particular choice of Cayley hypergraph is far from optimal, it serves as a simple comparison to the Ramanujan construction, and the comparison is favourable: it seems that KPSs with practical trade-offs between connectivity and resilience, as well as the benefits of good expansion, can be constructed from hypergraphs with good expansion.

But of course this is not entirely surprising. Recall from Remark 5.1 in Section 5.2.4 that a hypergraph is a set system, and that an r-uniform, k-regular hypergraph $H = (V, E)$ can be regarded as an r-uniform, k-regular design in itself, or as the representation of a k-uniform, r-regular design, whose incidence matrix is the transpose of the incidence matrix of H. Thus, our construction bears a strong resemblance to a combinatorial design, and the effectiveness of designs as constructions for KPSs is well established.

To recover the set system to which our Cayley hypergraph on 18 nodes corresponds, we generate the 46×18 incidence matrix M, where each row corresponds to a hyperedge and each column corresponds to a key, as shown in Figure 5.7.

Now, by reading down each column, we find the key set of each node and recover the set system $(\mathcal{X}, \mathcal{B})$ where $\mathcal{X} = \{01, 02, \ldots, 46\}$ and the blocks of $\mathcal{B}$ are:

$$\begin{array}{lll}
\{01,02,03,04,05,06,07,08\}, & \{01,09,10,11,12,13,14,15\}, & \{02,09,16,17,18,19,20\}, \\
\{03,10,16,21,22,23,24,25\}, & \{04,11,17,21,26,27,28\}, & \{05,12,18,22,26,29,30,31\}, \\
\{06,13,19,23,27,29,32,33\}, & \{07,14,20,24,29,34,35,36\}, & \{08,15,24,27,37,38,39\}, \\
\{15,20,23,26,40,41,42,43\}, & \{08,14,19,22,40,44,45\}, & \{07,13,18,21,37,41,44,46\}, \\
\{06,12,17,34,38,42,45,46\}, & \{05,11,16,32,35,39,43,46\}, & \{04,10,30,33,36,43,45\}, \\
\{03,09,28,31,36,39,42,44\}, & \{02,25,31,33,35,38,41\}, & \{01,25,28,30,32,34,37,40\}.
\end{array}$$

Then $(\mathcal{X}, \mathcal{B})$ is a design, regular of degree three, with rank eight. Notice that it

$$M = \begin{bmatrix}
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
1&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0\\
1&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0\\
1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0\\
1&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0\\
0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&1&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&1&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0\\
0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0\\
0&1&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0\\
0&0&1&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&1&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&1&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0\\
0&0&0&1&1&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&1&0&0&1&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&1&1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&1\\
0&0&0&0&1&1&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&1&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&1\\
0&0&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&1\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&1&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&1\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&1&0\\
0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1\\
0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&0&0\\
0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&1&0&0\\
0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&0
\end{bmatrix}$$

Figure 5.7: Incidence matrix for Cayley hypergraph on 18 nodes

is not uniform, as some blocks have seven points. However, if we simply added the hyperedges $\{2, 8, 14\}$ and $\{4, 10, 16\}$ to our Cayley hypergraph, we would then have a 3-uniform, 8-regular hypergraph, corresponding to an 8-uniform, 3-regular design. (These additional hyperedges are picked using the six nodes which have degree seven, and assigning hyperedges so that no pair of nodes is incident to more than one hyperedge.) Explicitly, we would have the design $(\mathcal{X}, \mathcal{B})$ where $\mathcal{X} = \{01, 02, \ldots, 48\}$ and the blocks of $\mathcal{B}$ are:

$$\begin{array}{lll}
\{01,02,03,04,05,06,07,08\}, & \{01,09,10,11,12,13,14,15\}, & \{02,09,16,17,18,19,20,47\}, \\
\{03,10,16,21,22,23,24,25\}, & \{04,11,17,21,26,27,28,48\}, & \{05,12,18,22,26,29,30,31\}, \\
\{06,13,19,23,27,29,32,33\}, & \{07,14,20,24,29,34,35,36\}, & \{08,15,24,27,37,38,39,47\}, \\
\{15,20,23,26,40,41,42,43\}, & \{08,14,19,22,40,44,45,48\}, & \{07,13,18,21,37,41,44,46\}, \\
\{06,12,17,34,38,42,45,46\}, & \{05,11,16,32,35,39,43,46\}, & \{04,10,30,33,36,43,45,47\}, \\
\{03,09,28,31,36,39,42,44\}, & \{02,25,31,33,35,38,41,48\}, & \{01,25,28,30,32,34,37,40\}.
\end{array}$$

Notice that this is not a $2-(48, 8, 1)$ design, since, for example, it can easily be checked that points 1 and 16 do not appear in the same block. However, this design is a configuration (Definition 2.4). Notice that for each block B_i there is exactly one block B_j with which it does not share any keys, i.e. $B_i \cap B_j = \emptyset$. Thus, B_i and B_j have 16 common neighbours, and so this is a 16-common intersection design. Further, for every B_i and B_j with $B_i \cap B_j \neq \emptyset$, there are $\lambda = 14$ common neighbours, and thus this is a $(18, 16, 14, 16)$ strongly regular graph. We have already established that strongly regular graphs make good constructions for KPSs in Section 4.5.3.3, and in particular, Lemma 4.9 gives

$$\epsilon \geq 8 - \frac{(-2) + \sqrt{(-2)^2 + 4 \times 0}}{4} = 8 \ .$$

Together with Equation (5.3.3), this gives $\epsilon = 8$, which shows that our proposed KPS construction does have good expansion.

Finally, we can find the connectivity and resilience of this KPS based on an

extended Cayley hypergraph. By observation,

$$\mathsf{Pr}_1 = \frac{144}{\binom{18}{2}} \approx 0.941, \tag{5.3.4}$$

and

$$\mathsf{fail}_1 = \frac{8}{128} = \frac{1}{16} = 0.0625. \tag{5.3.5}$$

Thus, by adding the extra two edges, the connectivity has increased from approximately 0.902 to 0.941 (Equations (5.3.1) and Equation (5.3.4)), and the resilience has fractionally improved: the difference between Equation (5.3.2) and Equation (5.3.5) is ≈ 0.000033.

Calculating fail_2 using conditional probabilities, we find that

$$\mathsf{fail}_2 = \frac{1}{17}\left(\frac{8}{128} + \frac{8}{112}\right) + \frac{16}{17}\left(\frac{8}{128} + \frac{7}{113}\right) = 0.125005 \tag{5.3.6}$$

because: suppose the adversary has compromised one node, which without loss of generality we will call N_1, and so the adversary has learned key set $\mathcal{K}_{N_1}$. Now there is one node N_i out of the remaining 17 nodes for which $\mathcal{K}_{N_1} \bigcap \mathcal{K}_{N_i} = \emptyset$, that is, one node which would reveal 8 further keys to the adversary. Compromising any of the other 16 nodes reveals exactly 7 new keys, as $|\mathcal{K}_{N_1} \bigcap \mathcal{K}_{N_j}| = 1$ for all $2 \leq j \leq 18$, $j \neq i$.

In [46] the following formula is given as an estimate for the resilience of a KPS composed of l copies of a strongly regular graph:

$$\mathsf{fail}_s = 1 - \frac{\binom{v-l-2}{s}}{\binom{v-2}{s}} \quad . \tag{5.3.7}$$

Table 5.1 gives the exact values of Equation (5.3.7) for our extended Cayley hypergraph ($l = 1$) for $s = 1, \ldots, 15$, and we see that it agrees exactly with Equation (5.3.5) and approximately with Equation (5.3.6).

s	1	2	3	4	5	6	7	8
fail_s	0.0625	0.125	0.1875	0.25	0.3125	0.375	0.4375	0.5

s	9	10	11	12	13	14	15
fail_s	0.5625	0.625	0.6875	0.75	0.8125	0.875	0.9375

Table 5.1: Resilience of the extended Cayley hypergraph

Table 5.1 shows that fail_s increases steadily with s. In particular, $\mathsf{fail}_s = 0.5$ when $s = \frac{v}{2} - 1$. As a point of comparison, recall from Lemma 3.2 that the resilience of an Eschenauer Gligor scheme is given by $\mathsf{fail}_s = 1 - \left(1 - \frac{k}{n}\right)^s$. Thus an Eschenauer Gligor scheme which also has $\mathsf{fail}_1 = 0.0625$ has parameters $\frac{k}{n} = \frac{1}{16}$, and this proportion gives $\mathsf{fail}_s > 0.5$ for $s \geq 11$. Indeed, for each s, the Eschenauer Gligor value of fail_s with $\frac{k}{n} = \frac{1}{16}$ is smaller than the approximate value of fail_s for the extended Cayley hypergraph KPS given by Equation (5.3.7).

However, if we make a comparison based on key storage and size of key pool, that is, fixing $k = 8$ and $n = 48$, we find that the KPS based on an extended Cayley hypergraph performs better, since the Eschenauer Gligor KPS has lower connectivity: $\mathsf{Pr}_1 \approx 0.796$, and poorer resilience: $\mathsf{fail}_1 \approx 0.167$ and $\mathsf{fail}_s > 0.5$ for $s \geq 4$.

In summary, although the extended Cayley hypergraph construction for a KPS does not scale well, it does provide good expansion, a lower bound for which can be easily found, and there are combinations of parameters k and n for which it outperforms the Eschenauer Gligor KPS.

5.4 Conclusion

Our exploration of hypergraphs and their expansion has led to some conclusions and some further questions.

Firstly, we have shown that using hypergraphs rather than graphs to represent KPSs has the benefits of demonstrating the key storage and resilience in addition to the connectivity, and we have therefore recommended using hypergraphs for representing and analysing KPSs.

After noting that existing expander graph constructions for KPSs have perfect resilience, we have demonstrated that, where perfect resilience is not required, connectivity and expansion can be improved by re-using keys. A hypergraph is a natural tool for constructing such schemes, where each key is known to more than two nodes. We therefore propose looking to the literature on hypergraphs with good expansion for KPS constructions with low key storage, good connectivity, high expansion and good - though not perfect - resilience.

There are currently few constructions for hypergraphs which approach the optimal proven bounds for expansion. We have reasoned that any future constructions proposed for hypergraphs with good expansion are likely to correspond to good constructions for KPSs since they would have the following properties:

- Re-use of keys, providing a trade-off between connectivity and resilience which is more suitable for many applications than the low connectivity given by a scheme with perfect resilience (and equal key storage).

- The benefits of good expansion: low diameter and average path length, more equal sharing of communication burdens, efficient data flow due to lack of bottlenecks, and few vulnerabilities (cutpoints and cut-edges).
- If the construction is deterministic, the KPS will also benefit from non-probabilistic metrics for key storage, connectivity and resilience, and, typically, more efficient shared key discovery.

However, we have noted that hypergraphs are set systems, and it is possible that any such construction may already be established in the literature as a combinatorial design. Indeed, since combinatorial designs are hypergraphs, our results in Section 4.5 about the expansion of configurations, μ-common intersection designs and strongly regular graphs support the claims of this chapter.

We used a Cayley hypergraph as an example of constructing a KPS from a hypergraph. Although far from optimal in terms of hypergraph expansion, we demonstrated that a hypergraph approach such as this could be used to construct a KPS with low key storage, good resilience and considerably higher connectivity and expansion than that of the Ramanujan expander graph from [18]. We also showed that the (extended) construction is equivalent to a strongly regular graph with expansion $\epsilon = 8$. It seems possible that the problem of finding a construction for random uniform hypergraphs with good expansion may be related to the problem of finding random strongly regular graphs, as discussed in [34] and [13] respectively, and that solving either problem may lead to a good construction for a KPS.

Chapter 6

Broadcast-enhanced key predistribution schemes

Contents

The work in this chapter is joint with Keith M. Martin, Siaw-Lynn Ng, Maura B. Paterson and Douglas R. Stinson, and appears as a paper on the Cryptology ePrint Archive [43].

6.1 Introduction

We present a formalisation of a category of schemes which we call *broadcast-enhanced key predistribution schemes* (BEKPSs). These schemes are suitable for networks with access to a trusted base station and an authenticated broadcast channel. We demonstrate that the access to these extra resources allows for the creation of BEKPSs with advantages over key predistribution schemes such as flexibility and more efficient revocation. There are many possible ways to implement BEKPSs, and we propose a framework for describing and analysing them.

We begin in Section 6.2 by introducing the ideas and terminology of a closely-related concept, namely broadcast encryption, before defining BEKPSs themselves and suggesting motivations for their study. In Section 6.3 we then give an example of a BEKPS and present in more detail the model, setting and metrics for analysing BEKPSs.

In [23], Cichoń, Gołębiewski and Kutyłowski propose a scheme for 'redistribut-

ing' keys to a wireless sensor network using a broadcast channel after an initial key predistribution. We classify this as a BEKPS and analyse it in that context. Section 6.4 provides some simpler versions of proofs from [23] and discusses modifications of the Cichoń et al. BEKPS based on defining a suitable intersection threshold.

Finally, we present two families of BEKPSs designed for particular scenarios: in Section 6.5 we propose a construction for BEKPSs designed for efficient revocation of nodes, and in Section 6.6 we propose BEKPSs which harness some of the advantages of hierarchical networks in a homogeneous network of nodes. We include these two families of schemes as examples of the many applications of BEKPSs, and to demonstrate that explicit formulae can be derived for the connectivity, resilience and broadcast cost, allowing for precise and efficient analysis of BEKPSs. Our analysis demonstrates their effectiveness in achieving their aims in resource-constrained networks. We conclude in Section 6.7 and present ideas for future work.

6.2 Motivation and definitions

6.2.1 Broadcast encryption

There are many applications where it is possible for a trusted base station to use a broadcast channel to communicate with nodes during the operational phase of the network. This broadcast channel can be used not only to distribute content, but also to conduct key management operations. Such applications

have been widely studied in the context of *broadcast encryption*. The first broadcast encryption schemes were given in [3, 33]. A classic example of a broadcast encryption application is pay-TV systems, where a key predistribution scheme is used to install keys into set-top boxes during the initialisation phase. The access to content is then managed by broadcasting the encrypted content along with a key management 'header' whose purpose is to provide an additional key 'layer' of content keys. The combined use of the predistributed keys and content keys defines the set of users that are able to decrypt and hence view the content. Note that whilst a pair of users may at times share keys, there is no motivation in the design of the scheme for users to be able to communicate with each other; the purpose of a broadcast encryption scheme is to control access to content.

6.2.2 Broadcast-enchanced key predistribution

Recall that key predistribution is a technique particularly suited to resource-constrained environments where public key cryptography is infeasible and there is no method for distributing symmetric keys once the network is operational, in particular for networks where a secure channel cannot be established between the network and base station after deployment. A major drawback of KPSs is that once the keys have been predistributed, subsequent key management operations are challenging to conduct [4]. We define a *broadcast-enhanced key predistribution scheme* to be a key distribution scheme designed for a network where a trusted base station and an authenticated broadcast channel will be available. We distinguish between the *underlying keys* which are pre-

distributed to the nodes, and the *temporal keys* which are broadcast by the base station and which the nodes may use for communication until the next broadcast. The base station broadcasts the temporal keys by encrypting them using underlying keys, as we will describe in Section 6.3.

Broadcast encryption can be regarded as a type of BEKPS, but with fundamentally different design goals: in broadcast encryption, temporal key sharing between nodes is incidental. In this paper we consider BEKPSs for applications where communication *between* nodes is important, for example in networks of data-gathering nodes, and so temporal key sharing is one of the primary design goals. Such differences of purpose create substantial differences between the designs of typical BE schemes and BEKPSs; in Section 6.5 we see how a naïve approach to designing a BEKPS from an LKH broadcast encryption scheme would not provide sufficient resilience.

6.2.3 Advantages of BEKPSs over KPSs

Where a base station and broadcast channel are available, there are several advantages of deploying broadcast-enhanced key predistribution as opposed to the use of a basic KPS, including:

- **Flexibility:** Underlying keys may be allocated in a way which is undesirable, either because there was a lack of control over the initial deployment, or because the purpose or priorities of the network have changed over time. For example, as node batteries become drained, it may be desirable to reduce the burdens on remaining nodes by distributing fewer

temporal keys and maintaining a connected network. A BEKPS enables the base station flexibility to ensure that some undesirable properties of the underlying key predistribution do not persist in the temporal key distribution. This is clear when we consider that the number of temporal keys shared by two nodes can be greater than, equal to, or less than the number of underlying keys which they share. Changes may be temporary, and hence only effective between a small number of updates, or permenantly sustained by future updates.

- **Ease of revocation:** In any BEKPS it is possible to revoke a node by ensuring it does not receive any future temporal keys. This can be done simply by omitting that node's underlying keys from the set of underlying keys used to encrypt all future temporal key broadcasts. This straightforward approach benefits from reducing the size of future broadcasts for the base station. However, it has the potential to reduce the connectivity and resilience of the remaining network (we define these terms in Section 6.3), and repeated revocations may lead to rapid degeneration of the remaining network [23]. In Section 6.5 we discuss a practical way to design a BEKPS for efficient revocation, where repeated revocations increase the broadcast cost but do not lead to network degeneration.

- **Creating hierarchy in the temporal key distribution:** The distribution of the temporal keys may coincidentally or deliberately feature 'imbalances', for example, certain nodes may store more keys than average. Nodes which store extra keys may be desirable for efficient routing of information through the network, and indeed KPSs have been proposed for heterogenous networks; see [17, 49] for a brief survey. In ho-

mogeneous networks (where all nodes have identical hardware), having comparatively more keys brings with it the disadvantages of increased communication burdens and quicker battery drainage. One way to reduce the damage that this causes to the network is to change at regular intervals the nodes which are required to store extra keys, as in the election of cluster heads in a network - see [62].

In any network where some nodes store extra keys, the compromise of such a node will be more detrimental to the resilience of the network than the compromise of the average node. In Section 6.6 we propose a family of BEKPSs which provide the benefits of efficient routing found in hierarchical networks, whilst frequent temporal key updates reduce the resilience risks and battery drainage.

6.3 Framework

In this section we propose the framework for a BEKPS protocol by describing our model and setting, defining the relevant notation and metrics for our analysis, and providing examples.

6.3.1 BEKPS model

We propose BEKPSs for networks of v nodes, $N_1, N_2, \ldots, N_v$, a trusted authority that preloads the underlying keys onto the nodes, and a (not necessarily distinct) trusted base station that can broadcast to all nodes using a broadcast channel.

A BEKPS protocol is comprised of the following phases:

1. **Underlying key predistribution:** Each node N_i is allocated a set of *underlying keys* from the underlying key pool $\mathcal{K}_v = \{u_1, u_2, \ldots, u_n\}$ before deployment, according to a key predistribution scheme. Underlying keys are solely for the purpose of encrypting and decrypting temporal keys, and should not be used for node to node communication. Once the nodes are deployed, we assume that the underlying keys are fixed and cannot be altered by the base station or overwritten by the nodes. (We justify this assumption by noting that if it were possible to securely supply nodes with new underlying keys, then the resulting system could simply be considered as an entirely new BEKPS and analysed within our model.)

2. **Temporal key distribution:** After the nodes are deployed, the base station broadcasts *temporal keys* to them in order for them to communicate. Each node is allocated a set of temporal keys from a temporal key pool $\mathcal{K}_\tau = \{t_1, t_2, \ldots, t_m\}$. The temporal keys are broadcast to the nodes encrypted by underlying keys, so that a node learns a temporal key if and only if the temporal key is encrypted by an underlying key known to that node.

3. **Shared key discovery:** Once the temporal keys have been broadcast, a shared key discovery protocol such as one of those given in [17, 73] can be used so that each node establishes the set of other nodes with which it shares keys. As in KPSs, this can be executed more efficiently if the temporal keys are assigned in a deterministic or publicly known way,

such that the broadcast of a node identifier is sufficient for other nodes to deduce that node's key identifiers.

All BEKPSs described in this chapter use variations on the Eschenauer Gligor KPS [32], as described in Scheme 2.1, Section 2.4.1. Shared key discovery therefore requires all nodes to broadcast their key identifiers, unless the assignment of keys is made public. Since this does not vary throughout the chapter, we will generally omit a description of this phase when defining our BEKPSs.

4. **Temporal key update:** A new temporal key pool may be generated and new sets of temporal keys broadcast as often as desired, according to the constraints of the network.

We now present an example of a BEKPS: the 'key redistribution' scheme of Cichoń et al. [23].

Scheme 6.1. *Underlying keys are distributed randomly, as in an Eschenauer Gligor KPS [32]: a key pool of underlying keys* $\mathcal{K}_v = \{u_1, u_2, \ldots, u_n\}$ *is generated, and each node is allocated a random* k*-subset of* $\mathcal{K}_v$*. A temporal key pool* $\mathcal{K}_\tau = \{t_1, t_2, \ldots, t_m\}$ *of* $m = n/c$ *temporal keys is generated, where* c *is a small constant. Each temporal key is encrypted using* c *underlying keys, and the base station then broadcasts the encrypted temporal keys to the network.*

In general, the choice of which underlying keys should encrypt each temporal key can be made randomly or deterministically. In the Cichoń et al. scheme, the underlying keys which should be used to encrypt each temporal key are chosen in a pseudorandom way. That is, we take a pseudorandom bijec-

tion π *between* $\{1,\ldots,m\} \times \{1,\ldots,c\}$ *and* $\{1,\ldots,n\}$ *and encrypt* t_i *using* $u_{\pi(i,1)},\ldots,u_{\pi(i,c)}$. *Presumably, a new* π *is chosen before each update. To simplify the notation, we relabel the underlying keys so that* $u_1 = u_{\pi(1,1)}, u_2 = u_{\pi(1,2)},\ldots,u_n = u_{\pi(n/c,c)}$ *so that the first* c *underlying keys encrypt* t_1 *and so on. Then the base station broadcasts:*

$$\begin{array}{cccc} E_{u_1}(t_1), & E_{u_2}(t_1), & \ldots, & E_{u_c}(t_1) \\ E_{u_{(c+1)}}(t_2), & E_{u_{(c+2)}}(t_2), & \ldots, & E_{u_{2c}}(t_2) \\ \vdots & & & \\ E_{u_{(n-c)}}(t_{n/c}), & E_{u_{(n-c+1)}}(t_{n/c}), & \ldots, & E_{u_n}(t_{n/c}) \end{array}$$

6.3.2 Setting

We now describe in more detail the setting for which we design BEKPSs.

Communication range: We consider static, homogeneous networks, as introduced in Section 2.2.1. As described in Sections 2.3.2 and 4.4, in many applications the nodes will not all be within communication range of each other, and therefore to fully analyse the deployed network we must consider the intersection of the key graph and the communication graph. However, our contributions in this chapter relate to the properties of the key graph, and so this is where we perform our analysis. Our results can be applied to a particular scenario with its corresponding communication graph in the same way as for any key predistribution scheme; for an example, see [18] where a random geometric graph is used to model the nodes' locations.

Adversary model: We assume the existence of a strong adversary who is able to compromise nodes to learn both temporal and underlying keys, and to keep records of all previous transmissions. It should be noted that a BEKPS does not provide backwards and forwards security against such an adversary, that is, exposure of an underlying key reveals all future and past temporal keys. It would be possible to provide such security against a weaker adversary who was only able to obtain temporal keys, for example in networks where underlying keys are stored in tamper-resistant hardware.

We suppose that the adversary compromises each node with equal probability. As discussed in Section 2.2.4, it is credible to imagine an adversary which compromises nodes in a carefully-targetted order so as to expose the greatest possible number of keys through the smallest number of compromises. However, the random node compromise model which we use allows us to compare many different schemes by providing a lower bound on fail_s.

Resource constraints: As with KPSs, we consider BEKPSs for resource-constrained environments where asymmetric cryptography is infeasible. If there were no other constraints on resources then it would be trivial to design a BEKPS with almost any properties:

- If there were no limit to the number of keys a node could store, then every pair of nodes could share a unique underlying key, or indeed every possible subset of nodes could share a unique underlying key. This would make it possible to achieve temporal key sharing across any ar-

bitrary group of nodes, though with the potential for high broadcast requirements.

- If the broadcast size were unlimited, each node could store a single, unique underlying key. The base station could then individually target nodes when broadcasting temporal keys, and achieve any desired combination of shared temporal keys amongst the nodes.

However, such a high use of resources will not always be feasible. Our focus in this chapter will be on BEKPSs for resource-constrained networks, where key storage and broadcast capability are limited, and where it is desirable for the longevity of the network to minimise the communication and computational requirements of the nodes.

6.3.3 Metrics

As in KPSs, the resource constraints dictate that there is a trade-off to be made between minimising key storage and maximising connectivity and resilience. For BEKPSs these metrics need to be defined in a little more detail, and we identify two further metrics to consider.

Key storage: We have previously denoted the number of keys that a node can store by k. When considering BEKPSs, we will use k to denote the number of underlying keys which nodes are required to store, and we will use κ to denote the number of temporal keys broadcast to each node. We denote the total number of keys which each node stores by σ, where $\sigma = k + \kappa$. As in

KPSs, key storage should be minimised. We note that in BEKPSs the number of temporal keys κ that a node is required to store is not necessarily constant over time.

Connectivity: As with KPSs, we measure connectivity by Pr_1, the probability that a randomly picked pair of nodes is connected. That is, the probability that they share at least q temporal keys, where q is the number of temporal keys required to compute a link key, as specified by the scheme. It is usually desirable to maximise Pr_1 in BEKPSs. Connectivity in the underlying key predistribution is not necessarily required.

Resilience: As with KPSs, resilience is measured by fail_s, the probability that a temporal link between two uncompromised nodes N_i, N_j is insecure after s other nodes are compromised. In this chapter we confine our analysis to the computation of fail_1 for the ease of comparing schemes.

Broadcast cost: We define the broadcast cost b to be the number of encrypted temporal keys broadcast by the base station at each update. That is, b gives the number of temporal keys being broadcast, multiplied by the number of underlying keys used to encrypt each temporal key. For example, in Scheme 6.1 we have $b = cn/c = n$. We consider ways to minimise the broadcast cost.

Revocation efficiency: Since nodes may develop faults and we assume the presence of an adversary, the ability to revoke keys and/or nodes adds robustness to a network. We will describe nodes which are to be revoked, that is, nodes suspected to be compromised by an adversary, displaying irregularities, or otherwise weakening the network, as 'compromised nodes'. We will refer to the remaining nodes which (as far as the base station can tell) have not been compromised and are functioning as they should, as 'uncompromised'.

We analyse a BEKPS's capability to revoke compromised nodes by the metrics:

- broadcast cost b_r for the revocation of r nodes during a temporal key update;
- number of uncompromised nodes which lose keys because of the revocation of r compromised nodes.

We note that for many subsets of these metrics it is trivial to devise a BEKPS which optimises them. For example, storage, connectivity and broadcast cost can be optimised by all nodes storing a single underlying key u_1, with which a single temporal key t_1 is encrypted and broadcast. Nodes would be connected with probability $\mathsf{Pr}_1 = 1$, but resilience would be minimised and revocation of a strict subset of nodes would be impossible. Therefore we are interested in schemes which provide suitable trade-offs between all of these metrics.

6.3.4 Expansion

We note that we have not listed expansion as a metric for analysing BEKPSs. This is because for some BEKPSs it is not an important design goal: in particular, the hierarchical BEKPS proposed in Section 6.6 deliberately creates a key graph where 'secondary' nodes are not well connected to the rest of the graph. In Chapter 4 we explained that such a key graph is undesirable for a KPS, creating unnecessary burdens on certain, more connected nodes. In contrast, in a BEKPS we can exploit the flexibility provided by the base station and broadcast channel to regularly redistribute these burdens and create a more efficient network in the long-term.

6.4 The BEKPS of Cichoń et al.

Much of the work in this section was conducted by Douglas R. Stinson and appeared in the preprint [64]. It was later incorporated into the paper [43], and we include our interpretation of it here for completeness in the discussion of BEKPSs.

We noted in Section 6.1 that Cichoń, Gołębiewski and Kutyłowski present a technique for 'key redistribution' in sensor networks [23], which we classify as a BEKPS. The details of their scheme are given in Scheme 6.1. In this section we provide simpler proofs of some of their results (Section 6.4.1), refine the estimates for the expected number of shared underlying and temporal keys between two nodes (Section 6.4.2) and give a precise analysis of the resilience

(Section 6.4.3). In Section 6.4.4 we present some numerical values of our formulae, and finally in Section 6.4.5 we discuss a modification to the scheme based on defining a suitable intersection threshold.

6.4.1 Simplifying proofs from [Cichoń et al. 2010]

In this section we give some simplified proofs of results from [23]. We begin by establishing a combinatorial framework.

We have noted that each node is assigned a k-subset of underlying keys from $\mathcal{K}_v$. The indices of these keys form a k-subset of $X = \{1, \ldots, n\}$ that we term a *block*. For the purposes of our analysis, each node can be identified with its corresponding key block; henceforth we will use the terms 'node' and 'block' interchangeably. Note that every block is a k-subset of $\{1, \ldots, n\}$ that is chosen independently and uniformly at random from the set of all $\binom{n}{k}$ possible k-subsets.

In [23, Theorem 1], formulae are proven for the expected number of shared underlying keys and the expected number of shared temporal keys between a pair of nodes. The proofs given in [23] use some heavy machinery involving generating functions. However, this theorem has a quick, simple proof based on the linearity of expectation of random variables.

First we consider [23, Theorem 1 (part 2)], which asserts that the expected number of temporal keys shared by two nodes is $\frac{n}{c}\left(1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right)^2$. Suppose that $G_1, \ldots, G_{n/c}$ partition the n-set $\{1, \ldots, n\}$ into $m = n/c$ disjoint c-sets. Let A

and B be random blocks. The number of temporal keys shared by A and B is

$$\omega_{A,B} = |\{i : A \cap G_i \neq \emptyset \text{ and } B \cap G_i \neq \emptyset\}|.$$

For $1 \leq i \leq n/m$, define a random variable $\tilde{\mathsf{X}}_i = 1$ if $A \cap G_i \neq \emptyset$ and $B \cap G_i \neq \emptyset$, and define $\tilde{\mathsf{X}}_i = 0$, otherwise. Let $\tilde{\mathsf{X}} = \sum_{i=1}^{n/c} \tilde{\mathsf{X}}_i$. Then $\tilde{\mathsf{X}}$ computes $\omega_{A,B}$ and $E[\tilde{\mathsf{X}}]$ is the expected value of $\omega_{A,B}$. It is obvious that

$$\mathsf{Pr}[A \cap G_i \neq \emptyset] = \mathsf{Pr}[B \cap G_i \neq \emptyset] = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}$$

and hence

$$E[\tilde{\mathsf{X}}_i] = \mathsf{Pr}[A \cap G_i \neq \emptyset \text{ and } B \cap G_i \neq \emptyset] = \left(1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right)^2.$$

By linearity of expectation,

$$E[\tilde{\mathsf{X}}] = \frac{n}{c}\left(1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right)^2, \tag{6.4.1}$$

which proves [23, Theorem 1 (part 2)].

To prove [23, Theorem 1 (part 1)] which states that the expected number of underlying keys shared between two nodes is $\frac{k^2}{n}$, we just set $c = 1$ in the formula derived above. We have

$$\begin{aligned} E[\text{number of shared underlying keys}] &= \frac{n}{1}\left(1 - \frac{\binom{n-1}{k}}{\binom{n}{k}}\right)^2 \\ &= n\left(1 - \frac{n-k}{n}\right)^2 \\ &= \frac{k^2}{n}, \end{aligned}$$

which proves the desired result.

6.4.2 Refining estimates

We have reproved the exact formula for the expected number of shared temporal keys. In [23, Corollary 1], an estimate for Equation (6.4.1) is given when k is roughly $\sqrt{n}$. However, we can also estimate Equation (6.4.1) when $k \neq \sqrt{n}$.

First, we estimate

$$\frac{\binom{n-c}{k}}{\binom{n}{k}} \approx \frac{(n-c)^k}{n^k} = \left(1 - \frac{c}{n}\right)^k,$$

so

$$E[\tilde{\mathsf{X}}] \approx \frac{n}{c}\left(1 - \left(1 - \frac{c}{n}\right)^k\right)^2.$$

Next,

$$\left(1 - \frac{c}{n}\right)^k \approx 1 - \frac{kc}{n} + \frac{k^2c^2}{2n^2},$$

so

$$E[\tilde{\mathsf{X}}] \approx \frac{n}{c}\left(\frac{kc}{n} - \frac{k^2c^2}{2n^2}\right)^2 = \frac{k^2c}{n}\left(1 - \frac{kc}{2n}\right)^2.$$

Finally, if we expand the square and ignore the last term, we get

$$E[\tilde{\mathsf{X}}] \approx \frac{k^2c}{n}\left(1 - \frac{kc}{n}\right). \tag{6.4.2}$$

If $k = \sqrt{n}$, then our estimate (6.4.2) is

$$\frac{k^2c}{n} - \frac{k^3c^2}{n^2} = \frac{k^2c}{n} - \frac{c^2}{\sqrt{n}}.$$

The estimate given in [23] is

$$\frac{k^2c}{n} + O\left(\frac{1}{\sqrt{n}}\right).$$

However, in [23], c is assumed to be fixed and the big-oh hides an unspecified constant that depends on c. To demonstrate this, we provide some example

values of the estimates:

n	k	c	exact $E[\tilde{\mathsf{X}}]$	estimate (6.4.2)	estimate from [23]
10000	50	8	1.933	1.920	2.000
10000	50	16	3.718	3.680	4.000
10000	100	8	7.466	7.360	8.000
10000	100	16	13.810	13.440	16.000
10000	150	16	28.876	27.360	36.000

6.4.3 Refining the calculation of resilience

For the analysis in this section we consider the resilience of the Cichoń et al. BEKPS during a single broadcast phase, that is, during a time period where each node's set of temporal keys is not updated. Thus we are concerned with the compromise of temporal keys; an adversary's knowledge of underlying keys is irrelevant to the analysis.

Cichoń et al. [23] study the resilience of their BEKPS but they make several simplifying assumptions. Here we give a much more general analysis and we derive general formulae for resilience. In [23, Theorem 2], it is assumed that two nodes A and B have exactly c temporal keys in common. In view of the estimates provided in the last section, this is roughly the expected number of common temporal keys when $k = \sqrt{n}$. Under this assumption, [23, Theorem 2] estimates the probability that a random node C contains these c common temporal keys to be $(kc/n)^c$. We calculate the resilience when $k \neq \sqrt{n}$.

6.4.3.1 Temporal key sets

As before, suppose that $G_1, \ldots, G_{n/c}$ partition an n-set $X = \{1, \ldots, n\}$ into $m = n/c$ disjoint c-sets. Suppose A is a random block (i.e., a k-subset of X) and define

$$\mathcal{I}(A) = \{i : A \cap G_i \neq \emptyset\}.$$

$\mathcal{I}(A)$ is the set of indices of the temporal keys held by A. Then let

$$\kappa_A = |\mathcal{I}(A)|;$$

κ_A is the number of temporal keys held by A.

Fix any i-subset $I \subseteq \{1, \ldots, m\}$. Define

$$M(i) = |\{A : \mathcal{I}(A) = I\}|.$$

Note that $M(i)$ counts the number of possible nodes whose set of temporal keys is equal to I. The value $M(i)$ does not depend on the particular i-subset I that was chosen.

It is easy to see that

$$|\{A : \mathcal{I}(A) \subseteq I\}| = \binom{ic}{k}. \tag{6.4.3}$$

We can derive a formula for $M(i)$ from (6.4.3) by applying the principle of inclusion-exclusion.

Lemma 6.1. *For $i \geq 1$, we have*

$$M(i) = \sum_{j=0}^{i-1} (-1)^j \binom{(i-j)c}{k} \binom{i}{j}. \tag{6.4.4}$$

Next, define

$$N(i) = |\{A : \kappa_A = i\}|.$$

$N(i)$ is the number of possible nodes holding exactly i temporal keys. The following is an immediate consequence of (6.4.4).

Lemma 6.2. *For $i \geq 1$, we have*

$$N(i) = \binom{m}{i} M(i) = \sum_{j=0}^{i-1} (-1)^j \binom{m}{i} \binom{(i-j)c}{k} \binom{i}{j}. \tag{6.4.5}$$

6.4.3.2 Intersection of two blocks

Next, we consider intersections of two blocks. For $\omega \geq 1$, define a ω-*link* to be an ordered pair of two nodes that contain exactly ω common temporal keys. Let $P(\omega)$ denote the number of possible ω-links; then

$$P(\omega) = |\{(A, B) : |\mathcal{I}(A) \cap \mathcal{I}(B)| = \omega\}|.$$

We have the following formula for $P(\omega)$:

Lemma 6.3. *For $\omega \geq 1$, we have*

$$P(\omega) = \sum_{i=\omega}^{k} \sum_{j=\omega}^{k} \binom{m-i}{j-\omega} \binom{i}{\omega} N(i) M(j). \tag{6.4.6}$$

For $\omega = 0$, we have

$$P(0) = \sum_{i=1}^{k} \sum_{j=1}^{k} \binom{m-i}{j} N(i) M(j). \tag{6.4.7}$$

Proof. Let $i = \kappa_A$ and $j = \kappa_B$. We can choose A in $N(i)$ ways. For each choice of A, choose ω indices in $\mathcal{I}(A)$ and choose $j - \omega$ indices in $\{1, \ldots, m\} \backslash \mathcal{I}(A)$.

Let the set of the j chosen indices be denoted by J. Then choose B such that $\mathcal{I}(B) = J$; there are $M(j)$ ways to do this. □

Remark 6.1. *We can verify the formulae (6.4.6) and (6.4.7) by checking that the following equations hold for various values of n, c and k:*

$$\sum_{\omega=0}^{k} P(\omega) = \binom{n}{k}^2$$

and

$$\frac{\sum_{\omega=1}^{k} \omega P(\omega)}{\binom{n}{k}^2} = \frac{n}{c}\left(1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right)^2.$$

6.4.3.3 Compromised links and resilience

We can now find expressions for the number of nodes which will compromise a given link, and derive the formula for fail_1. Suppose that (A, B) is a ω-link. Then define

$$S(\omega) = |\{C : \mathcal{I}(A) \cap \mathcal{I}(B) \subseteq \mathcal{I}(C)\}|.$$

$S(\omega)$ denotes the number of possible nodes that will compromise the ω-link (A, B), and it does not depend on the particular choices of A and B.

Lemma 6.4. *For any $\omega > 0$, we have*

$$S(\omega) = \sum_{i=\omega}^{k} \binom{m-\omega}{i-\omega} M(i). \tag{6.4.8}$$

Proof. Let $i = \kappa_C$. Choose $i - \omega$ indices in

$$\{1, \ldots, m\} \backslash (\mathcal{I}(A) \cap \mathcal{I}(B)).$$

Let J denote the i-set consisting of the $i - \omega$ chosen indices along with $\mathcal{I}(A) \cap \mathcal{I}(B)$. Then choose C such that $\mathcal{I}(C) = J$; there are $M(i)$ ways to do this. □

Finally, define

$$T(\omega) = |\{(A, B, C) : |\mathcal{I}(A) \cap \mathcal{I}(B)| = \omega \text{ and } \mathcal{I}(A) \cap \mathcal{I}(B) \subseteq \mathcal{I}(C)\}|.$$

$T(\omega)$ counts triples (A, B, C) where (A, B) is a ω-link compromised by C. It is clear, applying (6.4.8), that the following formula holds.

Lemma 6.5. *For any $\omega > 0$, we have*

$$T(\omega) = P(\omega)S(\omega) = \sum_{i=\omega}^{k} \binom{m-\omega}{i-\omega} M(i)P(\omega).$$

Now we are in a position to compute some resilience parameters. Recall that fail_1 denotes the probability that a random link (A, B) is compromised by a random node C.

Theorem 6.6. *The resilience after the compromise of one node is given by*

$$\mathsf{fail}_1 = \frac{\sum_{\omega=1}^{k} T(\omega)}{\sum_{\omega=1}^{k} P(\omega)\binom{n}{k}}. \tag{6.4.9}$$

Proof. The total number of possible ω-links with $\omega \geq 1$ is

$$\sum_{\omega=1}^{k} P(\omega),$$

so the total number of triples (A, B, C) where (A, B) is a link is

$$\sum_{\omega=1}^{k} P(\omega)\binom{n}{k}.$$

The total number of triples (A, B, C) where (A, B) is a link and C compromises this link is

$$\sum_{\omega=1}^{k} T(\omega).$$

The resilience is just the quotient of these two quantities. □

Define $\mathsf{fail}_1(\omega)$ to denote the probability that a random ω-link (A, B) is compromised by a random node C. We have the following obvious result.

Lemma 6.7. *For any $\omega \geq 1$, we have*

$$\mathsf{fail}_1(\omega) = \frac{S(\omega)}{\binom{n}{k}}. \tag{6.4.10}$$

Lemma 6.7 provides another way to derive the formula (6.4.9) for fail_1. Let λ_ω denote the probability that a random link is a ω-link. It is clear that

$$\lambda_\omega = \frac{P(\omega)}{\sum_{i=1}^{k} P(i)} \tag{6.4.11}$$

and

$$\mathsf{fail}_1 = \sum_{\omega=1}^{k} \lambda_\omega \mathsf{fail}_1(\omega). \tag{6.4.12}$$

Then, from (6.4.10), (6.4.11) and (6.4.12), we have

$$\begin{aligned} \mathsf{fail}_1 &= \sum_{\omega=1}^{k} \lambda_\omega \mathsf{fail}_1(\omega) \\ &= \sum_{\omega=1}^{k} \frac{P(\omega)S(\omega)}{\sum_{i=1}^{k} P(i)\binom{n}{k}} \\ &= \frac{\sum_{\omega=1}^{k} T(\omega)}{\sum_{\omega=1}^{k} P(\omega)\binom{n}{k}}, \end{aligned}$$

agreeing with (6.4.9).

6.4.4 Numerical examples

We now provide some numerical examples of our formulae. First, we give an example to illustrate the computation of resilience parameters.

Example 6.1. *Suppose* $n = 1000$, $c = 4$ *and* $k = 31$. *Then the expected number of temporal keys shared by a pair of nodes, given by (6.4.1), is* $\omega = 3.511857771$, *which is a bit less than* $\omega = 4$.

[23, Theorem 2] estimates $\mathsf{fail}_1(4)$ *by computing the quantity*

$$\left(\frac{kc}{n}\right)^c = 0.0002364213760.$$

A more accurate estimate for $\mathsf{fail}_1(4)$ *based on the analysis in [23], would be*

$$\frac{\binom{m-c}{k-c}}{\binom{m}{k}} = 0.0001980391200.$$

However, from (6.4.10), the exact value of $\mathsf{fail}_1(4) = 0.0001651542962$.

We find that the overall resilience of the scheme determined from (6.4.9) is $\mathsf{fail}_1 = 0.01330121549$. *This is quite a bit higher than* $\mathsf{fail}_1(4)$, *primarily because links consisting of fewer than four temporal keys (which occur frequently) are compromised with higher probability. This can be seen in the following tabulation of values* λ_ω *and* $\mathsf{fail}_1(\omega)$:

ω	λ_ω	$\mathsf{fail}_1(\omega)$
1	0.08756777557	0.1185218591
2	0.1843995070	0.01364696407
3	0.2407996311	0.001524883082
4	0.2188569817	0.0001651542962
5	0.1472998707	0.00001731603382
6	0.07626527018	0.000001755184555

Our next example considers the effect of varying the parameter k.

Example 6.2. *Suppose* $n = 1000$ *and* $c = 4$. *We compute the values of* fail_1

for various choices of k:

k	fail_1
5	0.01925413575
10	0.03349126556
15	0.03904935504
20	0.03548705708
25	0.02588255435
30	0.01518790238
35	0.007187785428
40	0.002776219702
45	0.0008938567010
50	0.0002464139425

It is interesting to observe that fail_1 *at first increases, and then decreases, as k increases. The higher values of* fail_1 *for small values of k reflect the fact that the network has fewer links and the links that do exist are more easily compromised.*

Our next example considers the effect of varying the parameter c.

Example 6.3. *Suppose $n = 1000$ and $k = 25$. We compute the values of* fail_1 *for various choices of c:*

c	fail_1
2	0.02636458442
3	0.02785890369
4	0.02588255435
5	0.02240961738
6	0.01861362594
7	0.01509874645
8	0.01211001320
9	0.009692483706
10	0.007795858957

The interesting thing to note here is that fail_1 *decreases as c increases beyond 3, but the decrease is gradual and not very dramatic.*

6.4.5 Intersection thresholds

We introduced the idea of an intersection threshold in the q-composite scheme (Scheme 3.1, Section 3.2). As q increases, the resilience increases and the connectivity decreases. We now develop formulae for these metrics which depend on the intersection threshold of the scheme.

Theorem 6.8. *For a scheme with intersection threshold q, we have that*

$$\mathsf{Pr}_1 = 1 - \frac{\sum_{i=0}^{q-1} P(i)}{\binom{n}{k}^2}. \tag{6.4.13}$$

Proof. There are $\binom{n}{k}^2$ possible pairs of nodes, of which $\sum_{i=0}^{q-1} P(i)$ are not connected. □

The formula for resilience (6.4.9) is generalised as follows.

Theorem 6.9. *For a scheme with intersection threshold q, the resilience is given by*

$$\mathsf{fail}_1 = \frac{\sum_{\omega=q}^{k} T(\omega)}{\sum_{\omega=q}^{k} P(\omega)\binom{n}{k}}. \tag{6.4.14}$$

Proof. The proof is a straightforward modification of the proof of Theorem 6.6. □

We now revisit Example 6.1.

Example 6.4. *Suppose $n = 1000$, $c = 4$ and $k = 31$, as in Example 6.1. We*

compute the connectivity and resilience for various values of q.

q	Pr_1	fail_1
1	0.9809852766	0.01330121549
2	0.8950825780	0.003202999469
3	0.7141893766	0.0005577036219
4	0.4779684839	0.00007970558807
5	0.2632730072	0.00001002335465

The use of an intersection threshold allows us to choose a suitable trade-off between connectivity and resilience. Observe that resilience increases substantially as q increases; however, connectivity decreases at the same time. For $q > 5$, the connectivity is too low to be practical. In this example, $q = 2$ or 3 provides a good way to 'balance' connectivity and resilience.

6.5 Revocation

In this section we consider how to design a BEKPS for a resource-constrained network where it is a high priority to be able to revoke nodes efficiently. As noted in Section 6.3.3, revocation of compromised nodes can be achieved in any BEKPS simply by avoiding using their underlying keys to encrypt new temporal keys. However, this can have the undesired effect of reducing the connectivity amongst uncompromised nodes, because they will receive fewer temporal keys if some of their underlying keys are no longer used. In general, it is possible to recover the level of connectivity Pr_1 after revocation by selecting future temporal keys from a smaller pool, so that each temporal key will be known to a higher proportion of the nodes. However, this lowers the resilience. We therefore design a BEKPS which enables the revocation of compromised nodes whilst retaining the connectivity and resilience in the remaining network,

and keeping key storage and broadcast cost low.

Clearly, the most precise way to be able to revoke individual nodes without causing any damage to the rest of the network is to assign a unique underlying key to each node of the network. If each node is given a single, unique underlying key, then this also has the benefit of achieving minimum key storage per node. However, an update requires a broadcast of $(v-r)\kappa$ temporal keys when r nodes have been revoked, which is infeasibly large for many applications.

If it is not the case that each node stores a unique underlying key, then revocation cannot be precise: uncompromised nodes will also be increasingly affected as the number of revocations increases. For example, if each node stores k underlying keys, then when a single node is revoked, k underlying keys are taken out of use by the base station. We denote this as $R(1) = k$ and derive a general formula for the number of redundant underlying keys after i revocations, when underlying keys are distributed using Scheme 2.1, the Eschenauer Gligor random KPS [32].

Lemma 6.10. *Suppose that each node stores k underlying keys, selected randomly from a key pool of n underlying keys. Let $R(i)$ denote the expected number of underlying keys removed from use when i nodes have been revoked. Then*

$$R(i) = n\left(1 - \left(1 - \frac{k}{n}\right)^i\right) \quad . \tag{6.5.1}$$

Proof. Let the i nodes which have been revoked be denoted by $N_1, \ldots, N_i$. For

$1 \leq j \leq n$ define a random variable

$$X_j = \begin{cases} 1 & \text{if key } k_j \text{ is known to at least one of } N_1, \ldots, N_i \\ 0 & \text{otherwise} \end{cases}$$

and let $X = \sum_{j=1}^{n} X_j$. We want to find $R(i) = E[X]$.

The expected value of X_1 is $E[X_1] = 1 - \mathsf{Pr}[k_1 \text{ is in none of } N_1, \ldots, N_i]$, so

$$E[X_1] = 1 - \left(1 - \frac{k}{n}\right)^i \quad .$$

Linearity of expectation gives $E[X] = nE[X_1]$, which gives the result. □

This means that an uncompromised node is unintentionally revoked with probability $\frac{\binom{R(i)}{k}}{\binom{n}{k}}$ as it can no longer learn any temporal keys in future broadcasts, and so after $i \geq 1$ revocations the network size $v(i)$ is

$$v(i) = (v - i)\left(1 - \frac{\binom{R(i)}{k}}{\binom{n}{k}}\right) \quad , \tag{6.5.2}$$

where v is the original network size.

We propose a BEKPS where precise revocation is possible, that is, $v(i) = v - i$, and which provides a choice of trade-offs between key storage and broadcast cost which are likely to be suitable for a wide range of network scenarios. To achieve this, we use *LKH schemes* (Section (6.5.1)) for the underlying key distribution and random key predistribution for the temporal keys.

Figure 6.1 shows the deterioration of the size of the network, $v(i)$ from Equation (6.5.2), in comparison to the straight line $(v - i)$, after i revocations. We see that for a small number of revocations, $i \leq 0.05v$, we have $v(i) \approx v - i$. However, there is then a rapid deterioration in the size of the network for

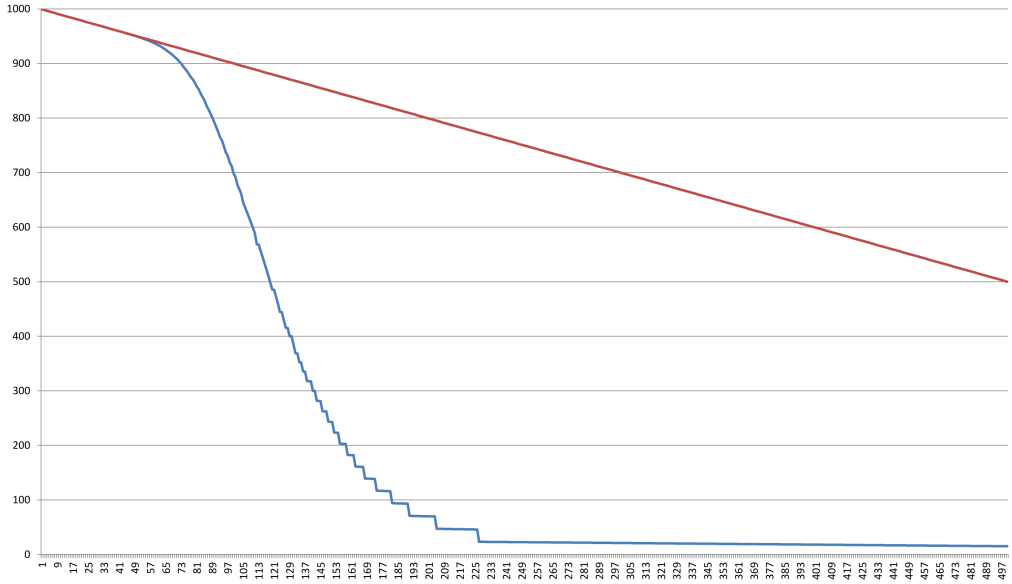


Figure 6.1: Plot of the deterioration of $v(i)$ (Equation (6.5.2)) in comparison to the straight line $(v - i)$ for an example network of $v = 1000$ nodes, where $n = 1000$ and $k = 30$.

$0.05v \leq i \leq 0.2v$, by which stage there are very few nodes remaining in the network. This demonstrates that, if only a small proportion of revocations are anticipated, a naïve approach using Scheme 2.1 for the underlying layer may be sufficient. However, for larger numbers of revocations, it highlights the importance of our proposed BEKPS for revocation, where the size of the network is always $v - i$ after i revocations.

6.5.1 LKH

Logical key hierarchy (LKH) schemes [38, 69, 72] are used in the literature of broadcast encryption for effective revocation. Each of the $v = 2^{d-1}$ nodes is allocated d keys, one of which is unique, and the other keys are known to $2^1, 2^2, \ldots, 2^{d-1}$ nodes respectively. In Figure 6.2 we demonstrate this on a

network of $v = 16 = 2^{5-1}$ nodes. Each node stores five keys: a unique key, a key δ_i shared with another node, a key γ_i shared with 3 other nodes, a key β_i shared with 7 other nodes, and the key α known to all nodes, called the *root* key.

If a message is to be broadcast to all nodes it can be encrypted using the root key. If a set of r nodes is to be revoked, then the message should be broadcast using the smallest set of keys known only to the $v - r$ uncompromised nodes. The size of a broadcast is then logarithmic in the size of the network.

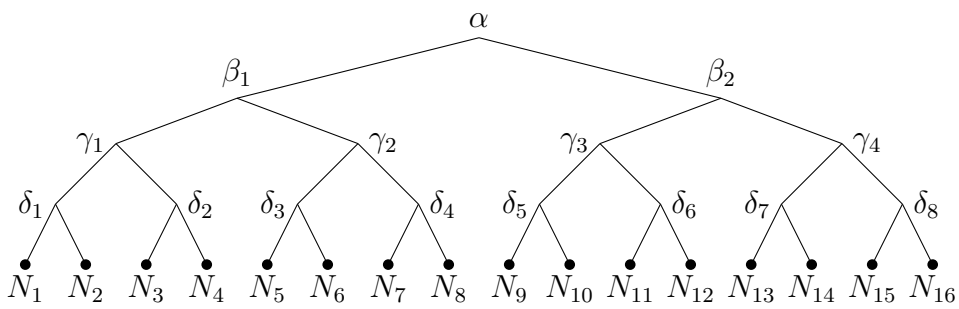


Figure 6.2: LKH tree on 16 nodes

6.5.2 BEKPS for revocation

We use LKH for the underlying layer of our BEKPS because it allows fine-grained revocation with low key storage and logarithmic broadcast cost. Other revocation schemes may also be adapted to form the underlying layer for specific BEKPS scenarios, however, to prevent our analysis from becoming unwieldy, we restrict our focus to using LKH as a basis for a BEKPS and varying the distribution of LKH trees in the underlying layer. For other broadcast encryption or revocation schemes the analysis will remain broadly similar to

that given here, and any benefits they provide over LKH would likely be reflected in the resulting BEKPS. For example, subset-cover revocation schemes such as the two proposed in [54] require $r \log v$ and $2r$ broadcasts, respectively, for the revocation of r nodes, whereas LKH requires $2r \log v$ broadcasts. The second of these schemes also reduces the key storage from $\log v$ (as in LKH) to $\frac{1}{2} \log^2 v$, and they additionally provide traitor tracing mechanisms. It seems likely, therefore, that similar improvements would be reflected in a BEKPS based on these schemes.

Similarly, we demonstrate a distribution of temporal keys which is broadly based on the Eschenauer Gligor random KPS, but in theory any KPS could be adapted for the temporal key distribution. We have chosen random key distribution because it is defined for any size of temporal key storage κ and key pool $\mathcal{K}_\tau$, making it highly adaptable, and it provides a relatively simple platform upon which to perform our analysis.

In Section 6.5.2.2 we will define exactly how the temporal keys are distributed. However, to motivate our choice of underlying key distribution, we note here that before any nodes are revoked, temporal keys will be broadcast using the root key of the LKH tree. Since the compromise of a node therefore exposes the temporal keys known to all other nodes, we propose using multiple, smaller trees to lessen this resilience risk. In this way our BEKPS is fundamentally different from an LKH broadcast encryption scheme, where a single LKH tree is used, but the base station may broadcast temporal keys to any chosen subset of nodes.

We propose the following BEKPS for scenarios where revocation is a high priority.

6.5.2.1 Underlying key predistribution

We assume that for a given application, each node can store σ keys in total, where σ is constant. To distribute underlying keys, we partition the v nodes into sets of size $\lambda = 2^{d-1}$, and we do this μ times. For ease of analysis, we will assume that $\lambda | v$ and that these partitions are chosen such that any tree from partition Π_i intersects any tree from partition Π_j in at most one node.

Within each set in each partition, nodes are allocated keys according to an LKH scheme, where the LKH tree has depth d. Thus each node belongs to μ different LKH trees of depth d and therefore must store $k = \mu d$ underlying keys, where μ and d are chosen so that $k < \sigma$. The total number of LKH trees is $L = \frac{\mu v}{\lambda}$.

Example 6.5. *Figure 6.3 illustrates an example of allocating $v = 16$ nodes into $\frac{v}{\lambda} = 4$ trees in each of $\mu = 2$ partitions. That is, each tree of $\lambda = 4$ nodes is represented by a shaded loop around the nodes; partition Π_1 is represented by the vertical loops, and partition Π_2 is represented by the horizontal loops. Each tree has been labelled $T_{i,j}$, where i denotes the partition to which it belongs, and j denotes the tree number within the partition. There are $L = \frac{\mu v}{\lambda} = 8$ trees in total, and each pair of trees intersects in at most one node.*

To construct a partition into $\mu\lambda$ trees which fulfills the above conditions will only be possible for certain values of the parameters v, k, λ and μ. This is a

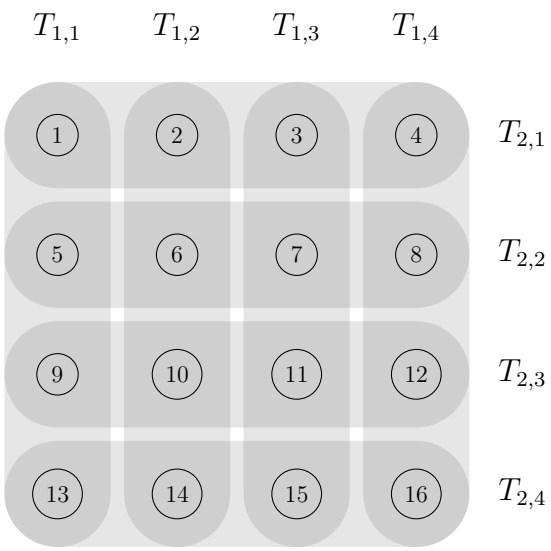


Figure 6.3: Example of partitioning nodes into $L = \frac{\mu v}{\lambda}$ trees

well-studied problem in the literature on *resolvable designs* [26, 53]. Briefly, a design is said to be resolvable if the blocks can be partitioned into μ sets or *parallel classes*, each of which forms a partition of the set of points. Thus, a resolvable $(v, \mu\lambda, \mu, \lambda)-$configuration provides a suitable construction for our underlying key predistribution. Resolvable designs have been widely studied; see [24] for existence results and constructions.

6.5.2.2 Temporal key distribution

From our assumption that total key storage is constant, it follows that each node can store at most $\kappa = \sigma - \mu d$ temporal keys. For each partition Π_i $(1 \leq i \leq \mu)$, a temporal key pool $\mathcal{K}_{\tau_i}$ of m keys is generated. We require that these key pools are disjoint, that is $\mathcal{K}_{\tau_i} \cap \mathcal{K}_{\tau_j} = \emptyset$ for all $1 \leq i < j \leq \mu$, so that we have μ independent Eschenauer Gligor schemes. Relaxing this requirement would allow for improvements in connectivity at the cost of decreased resilience.

A random set of temporal keys is allocated to every tree in the following way. For each partition i and each underlying LKH tree $T_{i,j}$ belonging to that partition (where $1 \leq i \leq \mu$ and $1 \leq j \leq \frac{v}{\lambda}$), a set of $\lfloor \frac{\kappa}{\mu} \rfloor$ temporal keys is chosen at random from the key pool $\mathcal{K}_{\tau_i}$ and encrypted using the underlying root key for $T_{i,j}$.

We assume that nodes require only one temporal key in common in order to establish a link, that is, the intersection threshold is $q = 1$, and a pair of nodes should use a single key to encrypt their communications, that is, $\Omega = 1$. For ease of analysis, we also create the following rules:

- If two nodes are in the same tree $T_{i,j}$ in any partition Π_i then they will share the set of temporal keys broadcast to $T_{i,j}$. The single temporal key which they use for communication should be randomly selected from this set. Any other keys which they may coincidentally share should be ignored. Note that there is no ambiguity because it is not possible for two nodes to be in more than one common tree.
- If two nodes are not in the same tree in any partition, they may form a link if they have at least one key in common; they should select just one of these keys at random to secure the link.

For example, we consider the possible connections between pairs of nodes in Figure 6.3:

- Nodes 1 and 9 both belong to the tree $T_{1,1}$ in partition Π_1. This means that they will share the set of temporal keys from key pool $\mathcal{K}_{\tau_1}$ which

are broadcast to tree $T_{1,1}$. They must pick one of these keys at random to encrypt their communications.

- Nodes 1 and 11 are not in a common tree in any partition. However, they may have one or more keys in common if
  - the set of keys broadcast to $T_{1,1}$ has non-empty intersection with the set of keys broadcast to $T_{1,3}$,

    and/or
  - the set of keys broadcast to $T_{2,1}$ has non-empty intersection with the set of keys broadcast to $T_{2,3}$.

  They should pick a single one of these keys at random with which to encrypt their communications.

Notice that if the set of keys broadcast to $T_{2,1}$ has non-empty intersection with the set of keys broadcast to $T_{2,3}$ then nodes 1 and 9 will have keys in common from key pool $\mathcal{K}_{\tau_2}$. However, they must use a key from key pool $\mathcal{K}_{\tau_1}$ to encrypt their communications because they belong to a common tree in partition Π_1.

In summary, if a pair of nodes are in the same tree then they have probability $\mathsf{Pr}_1 = 1$ of being connected. If not, the probability of them being connected is proportional to the Eschenauer Gligor connectivity probability (Lemma 3.1), as we explain below. If these rules were relaxed and a pair of nodes could use any combination of their shared temporal keys to secure their link, the connectivity and resilience of the network would increase, and so our analysis produces lower bounds.

6.5.2.3 Temporal key update

New temporal keys may be broadcast to the network as often as desired. To revoke a node N_i, the base station should broadcast new temporal keys to all nodes that are not in a common tree to N_i. For any nodes that are in a common tree to N_i, the base station can broadcast new temporal keys encrypted using the smallest set of LKH keys so that all uncompromised nodes receive the new temporal keys but N_i is unable to decrypt any temporal keys from the broadcast.

For example, to revoke the single node 6 in Figure 6.3, the base station should do the following:

- for each tree other than $T_{1,2}$ and $T_{2,2}$, broadcast a new set of temporal keys using the tree's root key;
- for tree $T_{1,2}$ broadcast a new set of temporal keys using the smallest number of underlying keys known to nodes 2, 10 and 14 but unknown to node 6;
- for tree $T_{2,2}$ broadcast a new set of temporal keys using the smallest number of underlying keys known to nodes 5, 7 and 8 but unknown to node 6.

To later revoke another node N_j the base station must repeat this process whilst continuing to ensure that node 6 receives no further temporal keys. Thus it can be seen that the broadcast load for the second revocation is greatest if

node N_j is not in either tree $T_{1,2}$ or $T_{2,2}$, and the broadcast load is actually lessened for the second revocation if node N_j is in either tree $T_{1,2}$ or $T_{2,2}$ and was receiving keys using its unique underlying key after the first revocation.

6.5.3 Analysis

We begin by calculating b_r, the broadcast cost of revoking r nodes.

Lemma 6.11. *For all $\mu, r \geq 1$, the broadcast cost of revoking r nodes is given by*

$$b_r \leq (\sigma - \mu d)\left(\frac{v}{\lambda} + rd - 2r\right) \quad . \tag{6.5.3}$$

Proof. We begin by calculating b_1, the broadcast cost required to revoke a single node N_i. Notice that N_i belongs to μ trees. Calculating b_1 requires the number of temporal keys per tree $\frac{\sigma}{\mu} - d$, the broadcast to the $L - \mu$ trees of uncompromised nodes, and the LKH revocation for the μ trees containing the compromised node, which requires $d - 1$ broadcasts per tree. Thus if the number of nodes per tree is greater than 1,

$$b_1 = \left(\frac{\sigma}{\mu} - d\right)(L - \mu + \mu(d-1)) = (\sigma - \mu d)\left(\frac{v}{\lambda} + d - 2\right) \quad .$$

If the number of nodes per tree is 1 (i.e. $d = 1$) then we assume that $\mu = 1$ so that the number of trees L equals the number of nodes v. Then $b_1 = (\sigma - 1)(L - 1)$.

The value of b_r for $r > 1$ will depend upon whether any of the r nodes are in the same tree(s). However, we have observed that the broadcast cost is largest

when each of the nodes to be revoked is in a different tree, hence

$$b_r \leq \left(\frac{\sigma}{\mu} - d\right)(L - r\mu + r\mu(d-1)) = (\sigma - \mu d)\left(\frac{v}{\lambda} + rd - 2r\right) \quad .$$

□

To analyse the connectivity and resilience of this BEKPS for revocation, we study separately the cases $\mu = 1$ and $\mu = 2$, and consider the effect of varying d and hence λ, the size of each LKH tree.

6.5.3.1 LKH trees where $\mu = 1$

We begin by considering $\mu = 1$, that is each node is in exactly 1 tree, and the total number of LKH trees is $L = \frac{v}{\lambda}$.

Lemma 6.12. *When $\mu = 1$, the connectivity is given by*

$$\mathsf{Pr}_1 = \frac{\lambda - 1}{v - 1}.1 + \frac{v - \lambda}{v - 1}\left(1 - \frac{\binom{m-(\sigma-d)}{\sigma-d}}{\binom{m}{\sigma-d}}\right) \quad .$$

Proof. Fix a single node N_i, belonging to a single LKH tree T_j (since $\mu = 1$). We consider the probability of N_i being connected to another of the $v - 1$ nodes, which fall into two categories:

- N_i will share temporal keys with the other $\lambda - 1$ nodes of its tree T_j with probability 1
- N_i will share at least one key with the remaining $v - \lambda$ nodes with the Eschenauer Gligor connectivity probability (Lemma 3.1), and $\kappa = \sigma - d$.

□

We now calculate the resilience metric fail_1. We must consider the probability of a random link being compromised when it is a link between nodes of the same tree, and when it is a link between nodes which are not in a common tree.

Lemma 6.13. *When $\mu = 1$, the resilience is given by*

$$\mathsf{fail}_1 = \frac{f_1\left[\frac{\lambda-2}{v-2}.1 + \frac{v-\lambda}{v-2}\left(\frac{\sigma-d}{m}\right)\right] + f_2\left[\frac{2\lambda-2}{v-2}.1 + \frac{v-2\lambda}{v-2}\left(\frac{\sigma-d}{m}\right)\right]}{f_1 + f_2} \quad ,$$

where $f_1 = \binom{\lambda}{2}\frac{v}{\lambda}$ *and* $f_2 = \left(1 - \frac{\binom{m-(\sigma-d)}{\sigma-d}}{\binom{m}{\sigma-d}}\right)\left(\binom{v}{2} - \binom{\lambda}{2}\frac{v}{\lambda}\right)$.

Proof. There are $f_1 = \binom{\lambda}{2}\frac{v}{\lambda}$ pairs of nodes in the network where both nodes are in the same tree. After compromising a single node, the adversary can break a link between one of these pairs with probability

$$\mathsf{fail}_{1,f1} = \frac{\lambda-2}{v-2}.1 + \frac{v-\lambda}{v-2}\left(\frac{\sigma-d}{m}\right) \quad ,$$

since for a given link, there are $\lambda-2$ nodes which, if compromised, would break that link with certainty by virtue of being in the same tree. A compromise of one of the remaining $v-\lambda$ nodes would reveal the desired key with probability $\frac{\sigma-d}{m}$.

The total number of pairs of nodes which are in different trees is $\binom{v}{2} - \binom{\lambda}{2}\frac{v}{\lambda}$, and each pair is connected with the Eschenauer Gligor connectivity probability (Lemma 3.1). Therefore we have that the expected number of links between pairs of nodes from different trees is

$$f_2 = \left(1 - \frac{\binom{m-(\sigma-d)}{\sigma-d}}{\binom{m}{\sigma-d}}\right)\left(\binom{v}{2} - \binom{\lambda}{2}\frac{v}{\lambda}\right)$$

and for these,

$$\mathsf{fail}_{1,f2} = \frac{2\lambda - 2}{v - 2}.1 + \frac{v - 2\lambda}{v - 2}\left(\frac{\sigma - d}{m}\right) \quad .$$

This is because, if we fix a link between uncompromised nodes N_i and N_j from different trees, there are $2(\lambda - 1)$ nodes which are in a common tree with either N_i or N_j and therefore know their shared key with certainty. This leaves $v - 2\lambda$ nodes which each have a probability of $\frac{\kappa}{m}$ of knowing their shared key. □

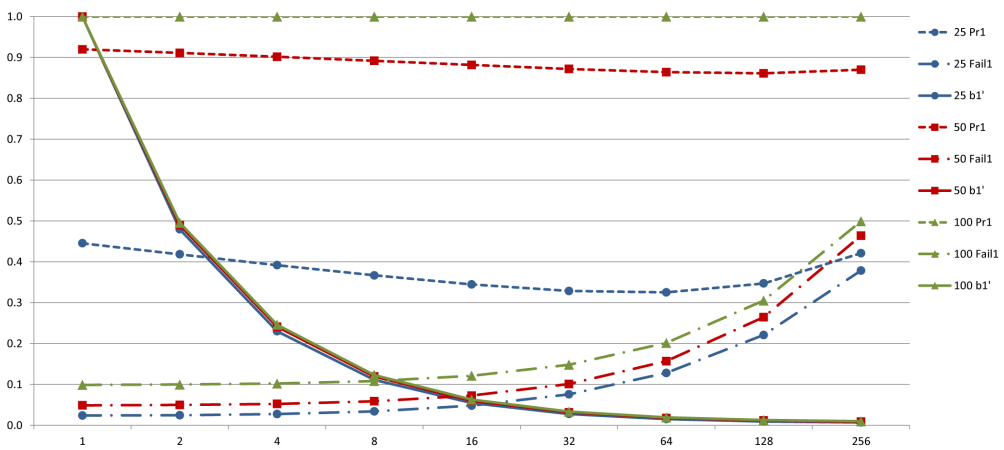


Figure 6.4: Plot of the values of Pr_1, fail_1 and b_1' when $\mu = 1$ and there are $1, 2, 2^2, \ldots, 2^8$ nodes per tree for key storage $\sigma = 25, 50$ and 100 respectively

We demonstrate these formulae in Figure 6.4. For comparison, we consider a fixed network size of $v = 1024$ nodes, and temporal key pool size of $m = 1000$ keys. We consider nodes which can store 25, 50 or 100 keys respectively, and plot the corresponding values of Pr_1 and fail_1 for underlying LKH layers of $1, 2, 4, \ldots, 256$ nodes per tree. In order to plot the broadcast cost (Equation (6.5.3)) on the same axes, we plot b_1 as a fraction of the number of keys to be broadcast when there is only one node per tree ($d = 1$), i.e. we plot $b_1' = \frac{b_1}{(\sigma - \mu)\left(\frac{v}{\lambda} - 1\right)}$.

We see that if nodes can store 100 keys then $\mathsf{Pr}_1 \approx 1$. If nodes can store

only 50 or 25 keys then the connectivity decreases significantly, but this has the advantage of lowering fail_1. Finally, we note that the broadcast cost b'_1 decreases exponentially as the number of nodes per tree increases. That is, for fixed network size v and key storage σ, the broadcast cost can be decreased by increasing the number of nodes per tree. The plots show that b'_1 is almost identical across different values of σ, however, as we know from the formula, the actual broadcast size b_1 increases with σ. For example, when there are 8 nodes per tree, the broadcast to revoke one node is $b_1 = 2730$ when $\sigma = 25$, $b_1 = 5980$ when $\sigma = 50$ and $b_1 = 12480$ when $\sigma = 100$.

We make some final remarks to justify the design of our BEKPS for revocation. The plot does not include the case where there is exactly one underlying LKH tree ($\lambda = v$), as in LKH broadcast encryption. It is clear from the formulae that whilst the broadcast cost would be minimised and the connectivity maximised, the resilience would be minimised, making it inadvisable for use as a BEKPS. Whilst a single LKH scheme is appropriate for many broadcast encryption applications, it is not appropriate for BEKPS because of the different design goals, and the fact that the base station always broadcasts temporal keys encrypted by the root key (or the smallest set of keys unknown to revoked nodes). Notice that if this restriction on the base station were removed and a single LKH tree were used for the underlying layer, this would be similar to our BEKPS except that nodes would have to store more underlying keys and therefore fewer temporal keys, restricting connectivity.

6.5.3.2 LKH trees where $\mu = 2$

We now consider the case where $\mu = 2$, that is, each node is a member of two trees, one from each partition. Each node therefore stores $2d$ underlying LKH keys, leaving space for it to store $\sigma - 2d$ temporal keys. The base station may broadcast a set of at most $\lfloor \frac{\sigma}{2} \rfloor - d$ temporal keys to each tree. Indeed, in general the base station may broadcast at most $\lfloor \frac{\sigma}{\mu} \rfloor - d$ to the root of each tree. For ease of notation we will omit the floor symbols.

Lemma 6.14. *When $\mu = 2$, the connectivity is given by*

$$\mathsf{Pr}_1 = \frac{2(\lambda - 1)}{v - 1}.1 + \frac{v - 1 - 2(\lambda - 1)}{v - 1}\left(1 - \left[\frac{\binom{m-(\frac{\sigma}{2}-d)}{\frac{\sigma}{2}-d}}{\binom{m}{\frac{\sigma}{2}-d}}\right]^2\right) .$$

The proof follows in the same way as that of Lemma 6.12. The Eschenauer Gligor probability contains a squared term because the probability of two nodes from different trees *not* being connected is the probability of them not sharing any keys from partition Π_1 multiplied by the probability of them not sharing any keys from partition Π_2.

Lemma 6.15. *When $\mu = 2$, the resilience is given by*

$$\mathsf{fail}_1 = \frac{f_1\left[\frac{\lambda-2}{v-2}.1 + \frac{v-\lambda}{v-2}\left(\frac{\frac{\sigma}{2}-d}{m}\right)\right] + f_2\left[\frac{2\lambda-2}{v-2}.1 + \frac{v-2\lambda}{v-2}\left(\frac{\frac{\sigma}{2}-d}{m}\right)\right]}{f_1 + f_2} ,$$

where $f_1 = \binom{\lambda}{2}L$ *and* $f_2 = \left(1 - \left[\frac{\binom{m-(\frac{\sigma}{2}-d)}{\frac{\sigma}{2}-d}}{\binom{m}{\frac{\sigma}{2}-d}}\right]^2\right)\left(\binom{v}{2} - \binom{\lambda}{2}L\right)$.

Proof. As in the proof of Lemma 6.13, we calculate fail_1 by considering the two cases:

1. If the link is between two nodes in a common tree in partition Π_i, $\lambda - 2$ other nodes from that tree can break the link with probability 1, and $v - \lambda$ nodes can each break the link with probability $\frac{\frac{\sigma}{2}-d}{m}$ using their knowledge of keys from the key pool $\mathcal{K}_{\tau_i}$. There are $f_1 = \binom{\lambda}{2}L$ such links.

2. If the link is between two nodes which are not in a common tree, $2\lambda - 2$ other nodes in their respective trees can break the link, and $v - 2\lambda$ other nodes can break the link with probability $\frac{\frac{\sigma}{2}-d}{m}$. The expected number of such links is $f_2 = \left(1 - \left[\frac{\binom{m-(\frac{\sigma}{2}-d)}{\frac{\sigma}{2}-d}}{\binom{m}{\frac{\sigma}{2}-d}}\right]^2\right)\left(\binom{v}{2} - \binom{\lambda}{2}L\right)$.

□

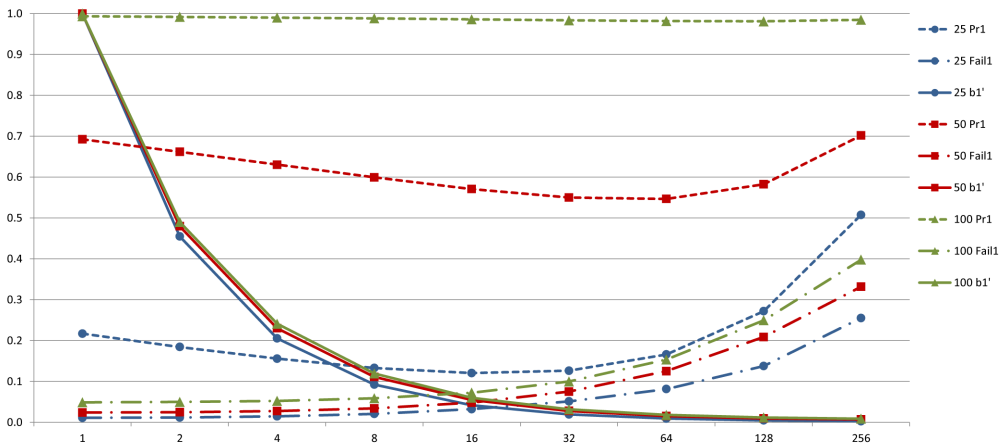


Figure 6.5: Plot of the values of Pr_1, fail_1 and b'_1 when $\mu = 2$ and there are $1, 2, 2^2, \ldots, 2^8$ nodes per tree for key storage $\sigma = 25, 50$ and 100 respectively

In Figure 6.5 we demonstrate some numerical values using these formulae. As in Figure 6.4, we set $v = 1024$ and $m = 1000$ for each of the key pools. Again, we see that Pr_1 is highest when 100 keys are stored per node, at the cost of a slightly increased value of fail_1. Comparing Figures 6.4 and 6.5 we find that,

when all other variables are fixed, the higher value of μ gives lower values of Pr_1 and fail_1, with little effect on b'_1. For comparison, we note that when there are 8 nodes per tree, the broadcast cost to revoke one node is $b_1 = 2080$ when $\sigma = 25$, $b_1 = 5460$ when $\sigma = 50$ and $b_1 = 11960$ when $\sigma = 100$, that is, a little lower than when $\mu = 1$. We therefore suggest that if the lower value of Pr_1 can be tolerated for the network's purposes, then $\mu = 2$ should be chosen to give higher resilience and lower broadcast for revocation.

When $\mu > 2$ the analysis for the connectivity and resilience becomes increasingly complex, and it remains an open problem to determine whether there are any advantages to higher values of μ. It seems likely that as μ increases Pr_1 will decrease, because nodes within the same tree will always be connected (unless revoked), but nodes which are not in a common tree can only be connected if they know keys from the same Eschenauer Gligor scheme, of which there are μ different schemes. Since each node can only store a fixed number of keys σ, as μ increases the number of temporal keys per Eschenauer Gligor scheme will decrease, and so Pr_1 will decrease accordingly. By the same argument, it seems likely that fail_1 would also decrease, giving higher resilience against an adversary

We have thus constructed an effective BEKPS protocol which allows efficient revocation and where, given key storage σ, there is some freedom to choose an appropriate trade-off between the parameters b_r, Pr_1 and fail_1, not only by varying the size of the key pool (as with any KPS), but also by varying the size λ of LKH trees in the underlying layer, and the number of trees μ to which each node belongs.

6.6 Hierarchical temporal key distribution

In Section 6.2.3 we introduced the idea of using a BEKPS to create hierarchy in the temporal layer, by broadcasting extra keys to certain nodes. This can provide more efficient routing of information through a network. The flexibility which a BEKPS provides to *change* which nodes have the extra keys reduces both the damage caused by extra battery usage and the risk posed to the resilience of the network. We will refer to nodes which are allocated extra keys as *primary nodes*, whilst the remaining *secondary nodes* have fewer keys.

Regularly changing the set of nodes that are primary will mean that the burdens of being a primary node are spread across the network over time. Random allocation of primary nodes reduces the risk of an adversary launching a targeted attack to reveal a high number of keys through a small number of node compromises.

6.6.1 BEKPS for hierarchical temporal key distribution

We now consider the question of how to create a BEKPS so that any node can be chosen as a primary node, and so that at any time period between broadcasts there should be p primary nodes and $v - p$ secondary nodes. (Note that the number of primary nodes p may be changed at any broadcast, so that there are p_i primary nodes after update i. However, since each update can be analysed without reference to the number of primary nodes which have gone before, we simply write p in the analysis which follows, for ease of notation.)

6.6.1.1 Underlying key predistribution

We propose that the best choice of KPS for the underlying keys is again one based on a revocation scheme such as LKH. We justify this with the following observations. Suppose that a node N_i with underlying key set U_i is to be chosen as a primary node. The base station must broadcast a higher proportion of temporal keys to it than to secondary nodes.

1. If at least one of the underlying keys in U_i is known uniquely to node N_i, then the base station can simply use this key to encrypt the extra temporal keys.

2. If none of N_i's underlying keys is known uniquely to N_i, that is, for each $u_j \in U_i$ there exists a node N_k with $u_j \in U_k$, then in broadcasting extra temporal keys, it will happen that some other nodes learn some extra temporal keys too. This will have the effect of creating a multiple-layered hierarchical network, where p nodes are primary nodes but amongst the remaining $v - p$ nodes there is variety in how many temporal keys are received. Whilst this may be desirable for some applications, in others it would cause some unnecessary battery drainage amongst the $v - p$ nodes and complicate routing protocols. We therefore restrict our study to a strictly two-layer hierarchy of primary and secondary nodes.

We conclude that to efficiently create primary and secondary nodes and to avoid burdening non-primary nodes unnecessarily, it is desirable that each node stores a unique underlying key. For similar reasons to those given in

Section 6.5.1, we propose an underlying layer based on LKH. As in Section 6.5, using a single LKH scheme minimises broadcast cost but maximises underlying key storage, and so we partition the nodes into several underlying LKH trees, each of size $\lambda = 2^{d-1}$.

6.6.1.2 Temporal key distribution

A straightforward way to allocate temporal keys in order to create p primary nodes is to use a slight modification of the Eschenauer Gligor KPS [32], where each primary node is allocated κ_1 temporal keys and each secondary node is allocated κ_2 temporal keys from a key pool $\mathcal{K}_\tau$ of m temporal keys. We will demonstrate that this allows the connectivity and resilience parameters to be easily altered with each broadcast, though of course many other KPSs would be suitable. For ease of analysis, we choose an intersection threshold of $q = 1$, that is, two nodes may form a link if they have one key in common. We will also assume that if two nodes have more than one key in common then they randomly select one of those keys to secure the link, that is, $\Omega = 1$. Relaxing this assumption would increase the resilience of the scheme.

The choice of primary nodes could be made deterministically or randomly, as desired. The benefits of choosing them deterministically are:

- more efficient shared key discovery
- the possibility of node identity authentication
- a node will not be required to be a primary node twice until necessary,

i.e. when all other nodes have been used as primary nodes at least once.

On the other hand, choosing the primary nodes at random may increase the difficulty for an adversary to target them for compromise. Given the increased risk to the resilience of the network which primary nodes cause, the unpredictability of the choice of primary nodes is an important security consideration. In our analysis we assume that the adversary compromises nodes at random, and therefore our analysis is applicable to deterministic and random allocations of primary nodes.

The base station may choose how to broadcast the temporal keys to secondary nodes in order to achieve a particular trade-off between connectivity, resilience and broadcast cost. We consider this in more detail in Section 6.6.2.2.

6.6.2 Analysis

6.6.2.1 Connectivity

We now derive formulae for the connectivity probabilities in terms of the size m of the temporal key pool and the number of temporal keys assigned to primary and secondary nodes, κ_1 and κ_2 respectively. We use $\mathsf{Pr}_{1,1}$ to denote the probability of two primary nodes being connected, $\mathsf{Pr}_{1,2}$ for the probability of a primary node and secondary node being connected, and finally $\mathsf{Pr}_{2,2}$ for the connectivity probability between a pair of secondary nodes.

Using the Eschenauer Gligor probability of connectivity given in Scheme 2.1,

we have that

$$\mathsf{Pr}_{1,1} = 1 - \frac{\binom{m-\kappa_1}{\kappa_1}}{\binom{m}{\kappa_1}}$$

and

$$\mathsf{Pr}_{1,2} = 1 - \frac{\binom{m-\kappa_1}{\kappa_2}}{\binom{m}{\kappa_2}} \quad .$$

Similarly, it can be seen that

$$\mathsf{Pr}_{2,2} \geq 1 - \frac{\binom{m-\kappa_2}{\kappa_2}}{\binom{m}{\kappa_2}}$$

when we consider that $1 - \binom{m-\kappa_2}{\kappa_2}/\binom{m}{\kappa_2}$ is the probability that two secondary nodes with different temporal key sets are connected. Two secondary nodes which are given the same set of temporal keys because they were encrypted with a shared LKH key will certainly be connected, and this is why a lower bound for $\mathsf{Pr}_{2,2}$ is given. The exact value of $\mathsf{Pr}_{2,2}$ will depend on choices which the base station makes regarding how to use the LKH tree(s) to distribute the temporal keys, as we describe in Section 6.6.2.2. In Section 6.6.2.3 we derive an estimate for $\mathsf{Pr}_{2,2}$ using an assumption about the temporal key distribution.

m	κ_1	κ_2	$\mathsf{Pr}_{1,1}$	$\mathsf{Pr}_{1,2}$	$\mathsf{Pr}_{2,2}$
500	50	10	0.9962	0.6548	$\geq$ 0.1844
500	50	15	0.9962	0.7990	$\geq$ 0.3709
1000	85	15	0.9996	0.7388	$\geq$ 0.2041
1000	85	25	0.9996	0.8945	$\geq$ 0.4731
1000	60	30	0.9783	0.8481	$\geq$ 0.6045
5000	100	50	0.8701	0.6377	$\geq$ 0.3965

Table 6.1: Examples of connectivity parameters (to four decimal places) for different key pool sizes m and the number of temporal keys given to primary and secondary nodes, κ_1 and κ_2 respectively.

Thus the base station can choose the parameters m, κ_1 and κ_2 to achieve different levels of the connectivity probabilities. Some example values are given in

Table 6.1; the values were chosen to demonstrate a variety of achievable trade-offs between the three connectivity parameters. Observe that connectivity between secondary nodes may not be necessary or even desirable; for example, to conserve resources whilst maintaining a connected network, it may be preferable to have a very low value of $\mathsf{Pr}_{2,2}$ as long as $\mathsf{Pr}_{1,2}$ is high enough to ensure that almost every secondary node is connected to at least one primary node, and $\mathsf{Pr}_{1,1}$ is high enough to ensure that almost every primary node is connected to all other primary nodes. Finally, we note that m, κ_1 and κ_2 are independent of the network size v, and can be changed at each broadcast if desired.

As with any random KPS, higher connectivity in this BEKPS results in lower resilience. In particular, the compromise of a primary node will reveal κ_1 of the total m keys. This risk will be reduced by dynamically changing the choice of primary nodes to lower the risk of their compromise, and by choosing $\mathsf{Pr}_{2,2}$, $\mathsf{Pr}_{1,2}$ and $\mathsf{Pr}_{1,1}$ to be as small as possible whilst retaining functional connectivity across the network. We calculate fail_1 in Section 6.6.2.3 after considering the different options available for the base station for the broadcast.

6.6.2.2 Broadcast cost

The following example considers how the temporal keys could be distributed to the secondary nodes.

Example 6.6. *Suppose we have a network of $v = 16 = 2^{5-1}$ nodes arranged in an LKH tree so that each node has to store $d = 5$ keys, as illustrated in Figure 6.6.*

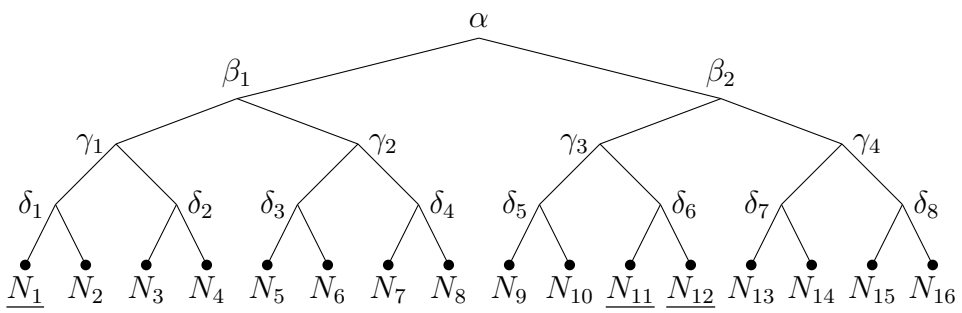


Figure 6.6: LKH tree on 16 nodes

Suppose that we wish to create $p = 3$ primary nodes, and at random we pick these to be nodes N_1, N_{11} and N_{12} (underlined in Figure 6.6). The base station would broadcast κ_1 temporal keys to each of these primary nodes, using their unique underlying keys. For the secondary nodes, there is a choice to be made about the temporal key broadcast.

1. *The key centre could broadcast a separate temporal key set to each of the secondary nodes using their underlying keys. This creates the maximum broadcast cost, the highest resilience, and $\mathsf{Pr}_{2,2}$ achieves its lower bound:*

$$\mathsf{Pr}_{2,2} = 1 - \frac{\binom{m-\kappa_2}{\kappa_2}}{\binom{m}{\kappa_2}} \quad .$$

2. *The key centre could minimise the broadcast by using the smallest set of LKH keys not known to the primary nodes, that is, by using the LKH keys associated with the minimal covering set of the secondary nodes. In this example, temporal keys would be broadcast to N_5, N_6, N_7, and N_8 encrypted by their shared key γ_2; to nodes N_3 and N_4 using δ_2; and to N_2 using its unique underlying key. Similarly, the broadcast to nodes N_{13}, N_{14}, N_{15} and N_{16} would be encrypted by their shared key γ_4, and*

nodes N_9 and N_{10} would be broadcast temporal keys encrypted by underlying key δ_5. The number of temporal key sets to be broadcast is then reduced from 16 to 8. The probability that a pair of secondary nodes have at least one common key is then

$$\mathsf{Pr}_{2,2} = \frac{2\binom{4}{2} + 2\binom{2}{2} + (1 \times 2 \times 4 \times 2 \times 4)\left(1 - \frac{\binom{m-\kappa_2}{\kappa_2}}{\binom{m}{\kappa_2}}\right)}{\binom{13}{2}} ,$$

which is greater than if unique underlying keys were used, at the cost of reduced resilience.

3. *In order to find a trade-off between the above options, the key centre could choose not to use the smallest set of LKH keys unknown to the primary nodes, for example by using the six δ_i keys with $i \in \{1, 2, \ldots, 8\} \setminus \{1, 6\}$, plus the unique key known to N_2. Then*

$$\mathsf{Pr}_{2,2} = \frac{6\binom{2}{2} + (1 \times 12 + 2 \times (10 + 8 + 6 + 4 + 2))\left(1 - \frac{\binom{m-\kappa_2}{\kappa_2}}{\binom{m}{\kappa_2}}\right)}{\binom{13}{2}} .$$

This example illustrates that there are choices to be made about the broadcast within each LKH tree, as well as about the number of underlying LKH trees $\frac{v}{\lambda}$. In our analysis we will assume that, on average, each set of temporal keys is broadcast to x secondary nodes, where $x < \lambda$ and $x \to \lambda$ as $p \to 0$ if the base station is using the minimum broadcast cost. Then to broadcast a set of κ_1 temporal keys to each primary node and κ_2-sets of keys to each secondary node requires a broadcast of size

$$b \approx \kappa_1 p + \kappa_2 \frac{(v - p)}{x} .$$

Using our assumption that each set of temporal keys is broadcast to x secondary nodes, we can revisit the expression we derived for $\mathsf{Pr}_{2,2}$ and use

weighted probability to derive the estimate

$$\mathsf{Pr}_{2,2} \approx \frac{x-1}{v-p-1} + \frac{v-p-x}{v-p-1}\left(1 - \frac{\binom{m-\kappa_2}{\kappa_2}}{\binom{m}{\kappa_2}}\right) \quad .$$

6.6.2.3 Resilience

We can make an estimate of fail_1 using Equation 3.2.1 from Section 3.2 with a weighted probability for primary and secondary nodes: the expected number of keys known to an adversary after the compromise of one node is $\kappa_1 \frac{p}{v} + \kappa_2 \frac{(v-p)}{v}$, and so we have that

$$\mathsf{fail}_{1,est} = \frac{\kappa_1 p + \kappa_2 (v-p)}{vm} \quad ,$$

since there is exactly one key securing each link. However, this method does not take into account the proportions of the three different types of links.

We now extend the definition of fail_1 to the hierarchical network setting. We retain our assumption that the adversary compromises all nodes with equal probability, and give each type of link in the network the same weight. In Table 6.2 we see comparisons between the approximation $\mathsf{fail}_{1,est}$ and our more detailed calculation of fail_1.

Lemma 6.16. *The resilience of a hierarchical BEKPS is given by*

$$\begin{aligned}
\mathsf{fail}_1 \quad = \quad & \frac{1}{T}\left[\binom{p}{2}\mathsf{Pr}_{1,1}\mathsf{fail}_{1,1}\right. \\
+ \quad & p(v-p)\mathsf{Pr}_{1,2}\mathsf{fail}_{1,2} \\
+ \quad & \left.\binom{v-p}{2}\mathsf{Pr}_{2,2}\left[\frac{v-p-x}{v-p-1}\mathsf{fail}_{2,2,a} + \frac{x-1}{v-p-1}\mathsf{fail}_{2,2,b}\right]\right] \quad ,
\end{aligned}$$

where $\mathsf{Pr}_{1,1}, \mathsf{Pr}_{1,2}$ *and* $\mathsf{Pr}_{2,2}$ *are as given above, and where*

$$\begin{aligned}
\mathsf{fail}_{1,1} &= \frac{\kappa_1(p-2)+\kappa_2(v-p)}{m(v-2)}, \\
\mathsf{fail}_{1,2} &\approx \frac{1}{v-2}\left(\frac{\kappa_1(p-1)+\kappa_2(v-p-x)}{m}+x-1\right), \\
\mathsf{fail}_{2,2,a} &\approx \frac{1}{v-2}\left(\frac{\kappa_1 p+\kappa_2(v-p-2x)}{m}+2(x-1)\right), \\
\mathsf{fail}_{2,2,b} &\approx \frac{1}{v-2}\left(\frac{\kappa_1 p+\kappa_2(v-p-x)}{m}+x-2\right),
\end{aligned}$$

and

$$T = \binom{p}{2}\mathsf{Pr}_{1,1} + p(v-p)\mathsf{Pr}_{1,2} + \binom{v-p}{2}\mathsf{Pr}_{2,2} \ .$$

Proof. We begin by finding the total number of links in the network, before any compromise, which is

$$T = \binom{p}{2}\mathsf{Pr}_{1,1} + p(v-p)\mathsf{Pr}_{1,2} + \binom{v-p}{2}\mathsf{Pr}_{2,2} \ .$$

Now we consider each type of link and its resilience.

1. **Primary-primary links**

   There are $\binom{p}{2}\mathsf{Pr}_{1,1}$ primary node to primary node links. Fix such a link between some primary nodes N_i and N_j, and consider the advantage to an adversary of compromising a single node $N_k \notin \{N_i, N_j\}$. If N_k is a primary node, the adversary will learn κ_1 keys; if N_k is secondary it will reveal κ_2 keys. Thus the adversary breaks the link with probability

$$\begin{aligned}
\mathsf{fail}_{1,1} &= \frac{1}{m}\left(\kappa_1\frac{p-2}{v-2}+\kappa_2\frac{v-p}{v-2}\right) \\
&= \frac{\kappa_1(p-2)+\kappa_2(v-p)}{m(v-2)} \ .
\end{aligned}$$

2. **Primary-secondary links**

   The number of primary node to secondary node links is $p(v-p)\mathsf{Pr}_{1,2}$. Fix such a link between primary node N_i and secondary node N_j. Suppose that the base station is using less than the maximum broadcast cost. Then the adversary can certainly break the link if it compromises a secondary node which is 'near' to N_j in the LKH tree, such that it stores the same set of temporal keys as N_j. That is, if we assume that on average, each set of temporal keys is broadcast to x secondary nodes, then an adversary who compromised a secondary node N_k will certainly be able to break the $p(x-1)$ links between primary nodes and the $x-1$ secondary nodes with which N_k shares the underlying LKH key used for the broadcast. Therefore, we have that a primary-secondary node link is broken with probability

$$\mathsf{fail}_{1,2} \approx \frac{1}{v-2}\left(\frac{\kappa_1(p-1)+\kappa_2(v-p-x)}{m}+x-1\right),$$

   where the approximation comes from x being an average value.

3. **Secondary-secondary links**

   There are $\binom{v-p}{2}\mathsf{Pr}_{2,2}$ secondary node to secondary node links. Fix such a link between secondary nodes N_i and N_j. As with primary-secondary links, the adversary can break the link with certainty if the broadcast cost is less than the maximum and the adversary compromises a secondary node N_k which has received the same temporal key set as one (or both) of N_i and N_j. Suppose that N_i and N_j have different temporal key sets $\mathcal{K}_{N_i}$ and $\mathcal{K}_{N_j}$. Then the probability of the link being broken after the

x	p	m	κ_1	κ_2	$\mathsf{Pr}_{1,1}$	$\mathsf{Pr}_{1,2}$	$\mathsf{Pr}_{2,2}$	fail_1	b
2^2	50	500	50	10	0.9962	0.6548	0.1870	0.0290	4875
2^3	50	500	50	10	0.9962	0.6548	0.1905	0.0357	3687.5
2^4	50	500	50	10	0.9962	0.6548	0.1973	0.0492	3093.75
2^3	100	500	50	10	0.9962	0.6548	0.1908	0.0383	6125
2^3	250	500	50	10	0.9962	0.6548	0.1921	0.0476	13437.5
2^3	50	1000	50	10	0.9280	0.4027	0.1027	0.0236	3687.5
2^3	50	500	80	10	0.9999	0.8281	0.1905	0.0384	5187.5
2^3	50	500	50	20	0.9962	0.8835	0.5683	0.0554	4875
2^3	50	500	50	30	0.9962	0.9617	0.8536	0.0744	6062.5

Table 6.2: Examples of connectivity and resilience metrics (to four decimal places) and broadcast cost for fixed network size $v = 1000$ and varying: the average number of secondary nodes to which a single temporal key set is sent, x; the number of primary nodes p; the number of keys in the key pool m; and the number of keys given to primary and secondary nodes, κ_1 and κ_2 respectively.

compromise of a single node is

$$\mathsf{fail}_{2,2,a} \approx \frac{1}{v-2}\left(\frac{\kappa_1 p + \kappa_2(v-p-2x)}{m} + 2(x-1)\right) \quad ,$$

and finally, if $\mathcal{K}_{N_i} = \mathcal{K}_{N_j}$, then the probability of breaking the link is

$$\mathsf{fail}_{2,2,b} \approx \frac{1}{v-2}\left(\frac{\kappa_1 p + \kappa_2(v-p-x)}{m} + x - 2\right) \quad .$$

Combining these results gives the stated formula. □

We illustrate some example values of fail_1 in Table 6.2.

We observe that:

- Increasing x reduces the broadcast cost and creates a marginal increase in $\mathsf{Pr}_{2,2}$, leaving the other connectivities unchanged. However, it noticeably increases fail_1, that is, it substantially reduces the resilience.

- For most applications the number of primary nodes need not be large; $\mathsf{Pr}_{1,1}$ and $\mathsf{Pr}_{1,2}$ can be set to be high independently of p, whilst increasing p reduces the resilience and significantly increases the broadcast cost.

- As we would expect, increasing m lowers the connectivity probabilities and fail_1, increasing the resilience. The broadcast cost is unaffected.

- Increasing κ_2 substantially increases the connectivity $\mathsf{Pr}_{2,2}$, whilst increasing the broadcast cost and reducing the resilience to a lesser extent. It may seem, therefore, that a comparatively high value of κ_2 will be desirable for most network applications. However, secondary node to secondary node communication may be unnecessary as long as $\mathsf{Pr}_{1,2}$ is high enough to ensure that most secondary nodes are connected to at least one primary node. It may therefore be desirable to keep κ_2 very low in order to increase resilience, reduce broadcast cost and conserve battery power in anticipation of secondary nodes becoming primary nodes in the future.

6.7 Conclusion

We have introduced the term *broadcast-enhanced key predistribution schemes* (BEKPS) in order to describe schemes which combine key predistribution with a trusted base station and broadcast channel, and discussed some of the many motivations for using BEKPSs. We developed a framework for the design and analysis of BEKPSs, and demonstrated its use throughout our paper. In Section 6.4 we provided simpler proofs for some of the results given by Cichoń

et al. in [23] for their scheme, which we classify as a BEKPS. We derived more general formulae to calculate the resilience and explained how intersection thresholds can be used to increase resilience at the cost of decreasing connectivity.

In Sections 6.5 and 6.6 we proposed appropriate BEKPS protocols for specific applications. In Section 6.5, we demonstrated a practical BEKPS where revocation can be performed without any uncompromised nodes losing keys. We showed that for a given key storage parameter σ, suitable trade-offs can be found between the connectivity, resilience and broadcast cost by varying the size of the temporal key pool, and the number and size of LKH trees used to distribute underlying keys. In Section 6.6 we demonstrated a BEKPS which creates a network with two-layer hierarchy. This brings the benefit of more efficient data routing. The ability to dynamically change the connectivity probabilities and the allocation of primary nodes reduces the risks of battery drainage and lowered resilience from which other hierarchical networks suffer.

Chapter 7

Concluding remarks

We have used a variety of combinatorial approaches and techniques to analyse and construct KPSs and BEKPSs for resource-constrained networks.

In Chapter 3 we saw how an unjustified assumption of independence led to an incorrect formula for calculating the resilience of q-composite random KPSs. We stated and proved, using combinatorial probability, the connectivity and resilience parameters for the seminal Eschenauer Gligor scheme [32], and provided a simpler formula for the connectivity of q-composite schemes [20]. We also noted that the graph of a random KPS is not equivalent to an Erdös-Rényi graph [31], as commonly asserted. The papers [27, 5, 74] study the connectivity threshold of the key graph for the Eschenauer Gligor scheme. It remains an open problem to study the key graph of other random KPSs such as the q-composite scheme.

Finally, we presented and proved a generalised resilience formula which applies to these schemes and to a wider class of random KPSs. We showed that the original formula slightly overestimates fail_s, that is, it underestimates the resilience. Although our contribution is of mathematical importance it has

limited impact on applications, as the original formula from [20] provides a close approximation to the true value.

In Chapter 4 we corrected some erroneous claims about the use of expander graphs in KPSs. We showed that expansion is a useful metric, amongst others, for analysing the key graph and intersection graph of a KPS. Exisiting KPSs which are directly based on expander graph constructions have good expansion and perfect resilience. However, we noted that whilst this may be appropriate for some applications, it is often desirable to have slightly lower resilience, with the benefit of much greater connectivity and expansion. We showed that random KPSs and various deterministic schemes based on combinatorial designs have good expansion; this is a previously unstated advantage of using these approaches to construct KPSs.

In Chapter 5 we argued that hypergraph representations of KPSs demonstrate the key storage and resilience, as well as the connectivity, which is the only metric demonstrated by an ordinary (unlabelled) graph representation. Building upon work in Chapter 4, we suggested using expanding hypergraph constructions to produce KPSs with good expansion and without perfect resilience. We gave an explicit example of a Cayley hypergraph, before noting its equivalence to a strongly regular graph. We described in general the relationship between hypergraphs and designs, and discussed the possibility that expanding hypergraph constructions may be related, or even equivalent, to existing constructions for combinatorial designs. In future work we will explore the potential relationship between random uniform hypergraph constructions, and constructions for random strongly regular graphs.

It seems likely that the expansion of other KPS constructions can be analysed, alongside various other graph-theoretic properties, in order to better understand their suitability (or otherwise) for practical applications. In particular, we would like to study the expansion of transversal designs, since these have an additional benefit of providing efficient shared key discovery [48].

Finally, in Chapter 6 we categorised a class of schemes as broadcast-enhanced key predistribution schemes (BEKPSs). We noted that the scheme of Cichoń et al. [23] is a BEKPS, and provided simplified proofs and further analysis of its connectivity and resilience metrics. We then proposed two families of BEKPS for particular purposes: one which allows efficient revocation, the other which allows nodes to act as a dynamically changing, hierarchical network. Our analysis showed that these schemes are effective in achieving their aims and provide flexible trade-offs between the conflicting parameters of key storage, connectivity, resilience and broadcast load. For future work, there are many variations of BEKPSs which can be studied. We note the following open questions:

- Can other revocation schemes provide advantages over LKH in the underlying key predistribution of some BEKPSs?
- Are there advantages to assigning temporal keys deterministically (other than aiding shared key discovery)?
- How can a BEKPS design be adapted to be more efficient if the locations of nodes are known to the base station?

Bibliography

[1] Noga Alon and Vitali D. Milman. Eigenvalues, expanders and supercontractors. In *IEEE 25th Annual Symposium on Foundations of Computer Science*, pages 320–322. IEEE, 1984.

[2] Noga Alon and Joel H. Spencer. *The Probabilistic Method*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2000.

[3] Shimshon Berkovits. How to broadcast a secret. In *Advances in Cryptology - Eurocrypt '91*, volume 547 of *Lecture Notes in Computer Science*, pages 535–541. Springer, 1991.

[4] Simon Blackburn, Keith M. Martin, Maura B. Paterson, and Douglas R. Stinson. Key refreshing in wireless sensor networks. In *ICITS 2008 Proceedings*, volume 5155 of *Lecture Notes in Computer Science*, pages 156–170. Springer, 2008.

[5] Simon R. Blackburn and Stefanie Gerke. Connectivity of the uniform random intersection graph. *Discrete Mathematics*, 309(16):5130–5140, 2009.

[6] Rolf Blom. An optimal class of symmetric key generation systems. In *Advances in Cryptology - Eurocrypt '84*, volume 209 of *Lecture Notes in Computer Science*, pages 335–338. Springer, 1985.

[7] Manuel Blum, Richard M. Karp, Oliver Vornberger, Christos H. Papadimitriu, and Mihalis Yannakakis. The complexity of testing whether a graph is a superconcentrator. *Information Processing Letters*, 13(4-5):164–167, 1981.

[8] Carlo Blundo and Paolo D'Arco. The key establishment problem. In *Foundations of Security Analysis and Design II*, pages 44–90. Springer, 2004.

[9] Carlo Blundo, Alfredo De Santis, Amir Herzberg, Shay Kutten, Ugo Vaccaro, and Moti Yung. Perfectly-secure key distribution for dynamic conferences. In *Advances in Cryptology – CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 471–486. Springer, 1993.

[10] Béla Bollobás. *Combinatorics: Set Systems, Hypergraphs, Families of Vectors, and Combinatorial Probability.* Cambridge University Press, Cambridge, 1986.

[11] Peter J. Cameron. *Combinatorics: Topics, Techniques, Algorithms.* Cambridge University Press, Cambridge, 1994.

[12] Peter J. Cameron. *Permutation Groups.* Cambridge University Press, Cambridge, 1999.

[13] Peter J. Cameron. Random strongly regular graphs? Preprint, 2001. `http://maths.qmul.ac.uk/~pjc/preprints/randsrg.pdf`.

[14] Peter J. Cameron. Strongly regular graphs. Preprint, 2001. `http://designtheory.org/library/preprints/srg.pdf`.

[15] Peter J. Cameron and Jacobus H. van Lint. *Graph Theory, Coding Theory and Block Designs (London Mathematical Society Lecture Note Series)*. Cambridge University Press, Cambridge, 1975.

[16] Seyit A. Çamtepe and Bülent Yener. Combinatorial design of key distribution mechanisms for wireless sensor networks. In *Computer Security–ESORICS 2004*, volume 3193 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2004.

[17] Seyit A. Çamtepe and Bülent Yener. Key distribution mechanisms for wireless sensor networks: a survey. *Rensselaer Polytechnic Institute, Computer Science Department, Technical Report TR-05-07*, 2005.

[18] Seyit A. Çamtepe, Bülent Yener, and Moti Yung. Expander graph based key distribution mechanisms in wireless sensor networks. In *ICC 06, IEEE International Conference on Communications*, pages 2262–2267, 2006.

[19] Haowen Chan and Adrian Perrig. PIKE: Peer intermediaries for key establishment in sensor networks. *IEEE INFOCOM*, 1:524–535, 2005.

[20] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 197–213. IEEE Computer Society, 2003.

[21] Wai-Kai Chen. *Applied Graph Theory: Graphs and Electrical Networks*. North-Holland Publishing Company, Amsterdam, 1976.

[22] Fan R. K. Chung. *Spectral graph theory (CBMS Conference on Recent Advances in Spectral Graph Theory)*. American Mathematical Society, 1997.

[23] Jacek Cichoń, Zbigniew Gołębiewski, and Mirosław Kutyłowski. From key predistribution to key redistribution. In *Algorithms for Sensor Systems*, volume 6451 of *Lecture Notes in Computer Science*, pages 92–104. Springer, 2010.

[24] Charles J. Colbourn and Jeffrey H. Dinitz. *Handbook of Combinatorial Designs*. Chapman & Hall / CRC, 2010.

[25] Giuliana Davidoff, Peter Sarnak, and Alain Valette. *Elementary Number Theory, Group Theory and Ramanujan Graphs*, volume 55 of *London Mathematical Society Student Texts*. Cambridge University Press, 2003.

[26] Yvo Desmedt, Niels Duif, Henk van Tilborg, and Huaxiong Wang. Bounds and constructions for key distribution schemes. *Advances in Mathematics of Communications*, 3(3):273–293, 2009.

[27] Roberto Di Pietro, Luigi V. Mancini, Alessandro Mei, Alessandro Panconesi, and Jaikumar Radhakrishnan. Redoubtable Sensor Networks. *ACM Transactions on Information and Systems Security (TISSEC)*, 11 (3):1–22, 2008.

[28] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[29] Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly secure message transmission. *Journal of the ACM*, 40(1):17–47, 1993.

[30] Wenliang Du, Jing Deng, Yunghsiang S Han, Pramod K Varshney, Jonathan Katz, and Aram Khalili. A pairwise key predistribution scheme for wireless sensor networks. *ACM Transactions on Information and System Security (TISSEC)*, 8(2):228–258, 2005.

[31] Paul Erdös and Alfréd Rényi. On the evolution of random graphs. In *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, pages 17–61, 1960.

[32] 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. ACM, 2002.

[33] Amos Fiat and Moni Naor. Broadcast encryption. In *Advances in Cryptology - CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer, 1994.

[34] Joel Friedman. Some graphs with small second eigenvalue. *Combinatorica*, 15(1):31–42, 1995.

[35] Joel Friedman and Avi Wigderson. On the second eigenvalue of hypergraphs. *Combinatorica*, 15(1):43–65, 1995.

[36] Subhas Kumar Ghosh. On optimality of key pre-distribution schemes for distributed sensor networks. In *Security and Privacy in Ad-Hoc and Sensor Networks*, volume 4357 of *Lecture Notes in Computer Science*, pages 121–135. Springer, 2006.

[37] Frank Harary. *Graph Theory.* Addison-Wesley, Reading, MA, 1969.

[38] Hugh Harney and Eric Harder. Logical key hierarchy protocol. *Internet Draft, Internet Engineering Task Force*, 1999.

[39] Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin - American Mathematical Society*, 43(4):439–562, 2006.

[40] John Hopcroft and Robert Tarjan. Algorithm 447: efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378, 1973.

[41] Michelle Kendall, Ed Kendall, and Wilfrid S. Kendall. A generalised formula for calculating the resilience of random key predistribution schemes. Preprint, 2012. `http://eprint.iacr.org/2012/426`.

[42] Michelle Kendall and Keith M. Martin. On the role of expander graphs in key predistribution schemes for wireless sensor networks. In *Research in Cryptology*, volume 7242 of *Lecture Notes in Computer Science*, pages 62–82. Springer, 2012.

[43] Michelle Kendall, Keith M. Martin, Siaw-Lynn Ng, Maura B. Paterson, and Douglas R. Stinson. Broadcast-enhanced key predistribution schemes. Preprint, 2012. `http://eprint.iacr.org/2012/295`.

[44] Jon Kleinberg and Ronitt Rubinfeld. Short paths in expander graphs. *IEEE Annual Symposium on Foundations of Computer Science*, 37:86–95, 1996.

[45] Jooyoung Lee and Douglas R. Stinson. A combinatorial approach to key predistribution for distributed sensor networks. In *IEEE Wireless Communications and Networking Conference*, pages 1200–1205. IEEE, 2005.

[46] Jooyoung Lee and Douglas R. Stinson. Deterministic key predistribution schemes for distributed sensor networks. In *Selected Areas in Cryptography*, volume 3357 of *Lecture Notes in Computer Science*, pages 294–307. Springer, 2005.

[47] Jooyoung Lee and Douglas R. Stinson. Common intersection designs. *Journal of Combinatorial Designs*, 14(4):251–269, 2006.

[48] Jooyoung Lee and Douglas R. Stinson. On the construction of practical key predistribution schemes for distributed sensor networks using combinatorial designs. *ACM Transactions on Information and System Security (TISSEC)*, 11(2):1–35, 2008.

[49] Kejie Lu, Yi Qian, and Jiankun Hu. A framework for distributed key management schemes in heterogeneous wireless sensor networks. *IEEE International Performance Computing and Communications Conference*, pages 513–520, 2006.

[50] Keith M. Martin. On the applicability of combinatorial designs to key predistribution for wireless sensor networks. In *Coding and Cryptology: Second International Workshop (IWCC2009)*, volume 5557 of *Lecture Notes in Computer Science*, pages 124–145. Springer, 2009.

[51] Keith M. Martin. *Everyday Cryptography: Fundamental Principles and Applications*. Oxford University Press, Oxford, 2012.

[52] Keith M. Martin and Maura B. Paterson. An application-oriented framework for wireless sensor network key establishment. *Electronic Notes in Theoretical Computer Science*, 192(2):31–41, 2008.

[53] Keith M. Martin, Maura B. Paterson, and Douglas R. Stinson. Key predistribution for homogeneous wireless sensor networks with group deployment of nodes. *ACM Transactions on Sensor Networks*, 7(2):1–19, 2010.

[54] Dalit Naor, Moni Naor, and Jeff Lotspiech. Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology - Crypto 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer, 2002.

[55] Alon Nilli. On the second eigenvalue of a graph. *Discrete Mathematics*, 91(2):207–210, 1991.

[56] Maura B. Paterson and Douglas R. Stinson. A unified approach to combinatorial key predistribution schemes for sensor networks. *Designs, Codes and Cryptography,* advance online publication, doi: 10.1007/s10623-012-9749-4. Springer, 2012.

[57] Eric Purdy. *Locally expanding hypergraphs and the unique games conjecture.* PhD thesis, University of Chicago, 2008.

[58] Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 3–13, Washington, DC, USA, 2000. IEEE.

[59] Ronald L. Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[60] Hosein Shafiei, Arash Mehdizadeh, Ahmad Khonsari, and Mohamed Ould-Khaoua. A combinatorial approach for key-distribution in wireless sensor networks. In *IEEE Global Telecommunications Conference*, pages 1–5. IEEE, 2008.

[61] Simon Singh. *The Code Book: The Secret History of Codes and Code-breaking*. Fourth Estate, 2002.

[62] Stanislava Soro and Wendi B. Heinzelman. Cluster head election techniques for coverage preservation in wireless sensor networks. *Ad Hoc Networks*, 7(5):955–972, 2009.

[63] Douglas R Stinson. *Cryptography: Theory and Practice*. Chapman & Hall/CRC, Boca Raton, FL 33487-2742, 2006. ISBN 1-58488-508-4.

[64] Douglas R. Stinson. Comments on a sensor network key redistribution technique of Cichoń, Gołębiewski and Kutyłowski. Preprint, 2011. `http://eprint.iacr.org/2011/259.pdf`.

[65] Robert M. Tanner. Explicit concentrators from generalized N-gons. *SIAM Journal on Algebraic Discrete Methods*, 5(3):287–293, 1984.

[66] Robert Tarjan. A note on finding the bridges of a graph. *Information Processing Letters*, 2(6):160–161, 1974.

[67] Jacobus H. van Lint and Alexander Schrijver. Construction of strongly regular graphs, two-weight codes and partial geometries by finite fields. *Combinatorica*, 1(1):63–73, 1981.

[68] Walter D. Wallis. *Combinatorial Designs (Pure and Applied Mathematics)*. Marcel Dekker Inc, 1988.

[69] Debby Wallner, Eric Harder, and Ryan Agee. Key Management for Multicast: Issues and Architectures. *Internet Engineering Task Force*, (2627), 1999.

[70] Reizhong Wei and Jiang Wu. Product construction of key distribution schemes for sensor networks. In *Selected Areas in Cryptography: 11th International Workshop, SAC 2004*, volume 3357 of *Lecture Notes in Computer Science*, pages 280–293. Springer, 2004.

[71] Geoffrey Werner-Allen, Jeff Johnson, Mario Ruiz, Jonathan Lees, and Matt Welsh. Monitoring volcanic eruptions with a wireless sensor network. *Proceeedings of the Second European Workshop on Wireless Sensor Networks, 2005*, pages 108–120, 2005.

[72] Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. Secure group communications using key graphs. *Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 68–79, 1998.

[73] Yang Xiao, Venkata K. Rayi, Bo Sun, Xiaojiang Du, Fei Hu, and Michael Galloway. A survey of key management schemes in wireless sensor networks. *Computer Communications*, 30(11-12):2314–2341, 2007.

[74] Osman Yağan and Armand M. Makowski. On the existence of triangles in random key graphs with a note on their small-world property. Preprint, 2013. `http://andrew.cmu.edu/user/oyagan/Journals/IT_Triangle.pdf`.

[75] Dae Hyun Yum and Pil Joong Lee. Exact formulae for resilience in random key predistribution Schemes. *IEEE Transactions on Wireless Communications*, pages 1–5, 2012.

[76] yWorks GmbH. Analyzing Graphs, Chapter 4: Working with the Graph Structure. `http://docs.yworks.com/yfiles/doc/developers-guide/analysis.html`.