The Combinatorics of Cryptographic Key Establishment

Keith M. Martin

Abstract

One of the most important processes involved in securing a cryptographic system is establishing the keys on which the system will rely. In this article we review the significant contribution of combinatorial mathematics to the development of the theory of cryptographic key establishment. We will describe relevant applications, review current research and, where appropriate, identify areas where further research is required.

1 Introduction

Cryptography provides the core information security services that are necessary to safeguard electronic communications. The sound management of cryptographic keys is the fundamental supporting activity that underpins the secure implementation of cryptography. The purpose of this paper is to demonstrate the significant contribution of combinatorial mathematics to the development of the theory of cryptographic key establishment.

- **Scope**: This paper surveys areas of key establishment where combinatorial models or construction techniques have proven of value. Our aim is not to provide a comprehensive survey of the literature, but rather to provide sufficient coverage that most relevant work will be (to use the terminology of Section 7.3.3) at most a "two-hop path" from this review. This paper is not an attempt to survey the vast research on key establishment in general.

- **Detail**: The primary aim is to bring these applications of combinatorics to the attention of the mathematical community within a sensible unifying framework. We thus focus on introducing concepts and providing pointers for further study. This paper contains no proofs. Combinatorial modelling typically involves the establishment of bounds and constructions. For illustrative purposes we will tend to focus on constructions in this review.

- **Novelty**: This article largely describes existing research and will contain few surprises for those already familiar with the field. That said, as far as we are aware, the full range of applications covered in this review have not previously all been presented within a common framework and so it is hoped that this may be of interest in its own right.

- **Applicability**: While the schemes in this paper are all of potential interest to a designer of a real cryptographic system, most are proposed under more rigorous mathematical security requirements than are demanded by the"real world", where security is often (validly) traded off against efficiency and practicality. Most of the key establishment schemes discussed in this paper are unlikely to be currently employed in commercial applications. This does not,

however, preclude them from influencing real designs or prevent them from being used in the future. They are all of interest in their own right as theoretical models of what is possible.

The remainder of the paper is structured as follows. In Section 2 we provide some background to cryptography and key management. We present a framework for key establishment in Section 3, which sets the context for comparison of schemes presented elsewhere in the paper. Section 4 contains some brief mathematical preliminaries. Our main review is spread over the subsequent three sections, where in Section 5 we look at key predistribution, in Section 6 we look at key distribution, and in Section 7 we look at key agreement. In Section 8 we provide some concluding remarks.

2 Cryptographic key management

We live in a society where electronic communication has become indispensable and ubiquitous. Electronic networks pervade all aspects of our professional and private lives. Many people, however, fail to appreciate that well-established and understood security safeguards that apply to traditional communication media are often absent in their electronic counterparts. In fact many of the features of electronic communication that we most value potentially expose information to previously unimaginable vulnerabilities.

The simple act of writing a letter suffices to illustrate this well. A traditional hand-written letter is normally posted in a sealed envelope and delivered to the specified address by a postal service. Interception of the contents requires physical access to the letter during the delivery service and breaking of the protective seal. The recipient can inspect the envelope for damage and may well gain assurance of the integrity of the contents through physical means, such as inspection of the postmark and recognition of handwriting. In contrast, an email is normally unprotected. In order to reach the specified address it is sent over a series of computer networks, passing through numerous computer servers and network routers on its journey. At any point its contents could be inspected, copied, forwarded, changed, and even the name of its sender could be forged. The recipient gains only cosmetic levels of assurance that the content is genuine and unaltered. Security in this electronic environment relies more on luck and lack of motivation for attack. If someone really wants to learn the content of an email then with very little technological expertise they probably can. With just a few clicks of their mouse button they can also share it with a significant percentage of the world's population.

There are of course solutions to most of these electronic security problems, as it is inconceivable that some of the earliest adopters of commercial electronic networks, such as the banking industry, could have developed electronic business without suitable security mechanisms in place. The science of *cryptography* underpins the bulk of these solutions. Cryptography is essentially a toolkit of mathematical techniques, algorithms and protocols that provide the core security services that are required in electronic communications. These services include *confidentiality* (restricting access to the contents of communicated data), *data integrity* (protecting data from manipulation), *data origin authentication* (correctly attributing the originator of

some data) and *non-repudiation* (providing evidence of the occurrence of a data exchange that cannot later be denied). We have all used cryptography, even if we are not always aware that we are doing so, as cryptographic mechanisms are used to protect banking transactions (for example ATM transactions, Internet banking, SWIFT transfers), mobile telephone communications, secure web transactions (by means of the SSL protocol), password storage on computer operating systems, etc. Most modern computers have the facility to encrypt email (even if we tend not to use it) and almost everyone carries around at least one plastic card with a chip on it, whose purpose is primarily to allow cryptographic computations to be performed when that card is placed in contact with a reader.

Regardless of their purpose or application, most cryptographic mechanisms critically rely on the use of *keys*, which are essentially numbers selected at random from a large space. As the majority of cryptographic mechanisms are published processes that can be analysed by anyone, the entire security of a cryptographic mechanism typically relies on the protection of the relevant keys. The nature of these keys provides a natural broad classification of cryptographic mechanisms into *symmetric* mechanisms, where the secret keys employed by the sender and the receiver of data must be identical, and *public-key* mechanisms, where only one of the keys needs to be secret, and the other key can be made public. While for many applications both symmetric and public-key mechanisms are used in tandem, the fact that symmetric mechanisms tend to be faster and require shorter keys means that for a range of applications, symmetric key mechanisms are favoured. We will encounter several such applications during this paper.

Assuming that strong cryptographic mechanisms are employed and implemented correctly, it is fair to say that the security of cryptographic mechanisms relies almost entirely on the secure management of the relevant keys. The phrase *key management* tends to be associated with the entire lifecycle of a cryptographic key, including its creation (*key generation*), the methods by which it is sent to the relevant users of the system (*key establishment*), the techniques that are used to change or refresh it (*key update*) and ultimately the means by which it is deleted at the end of its usage period (*key destruction*).

The purpose of this paper is to review a number of interesting areas where combinatorics has found application in aspects of key management, and in particular key establishment. We will generally not need to concern ourselves with the purpose, or indeed even the algorithms, for which these keys are needed. The key establishment problems that we will look at in detail are mostly intended to support applications of symmetric cryptography. The reason for this is quite simple. The fundamental key management challenge in symmetric cryptography is one of key establishment. We need to arrange for every group of users who wish to engage in a secure communication exchange to have a common key. It should already be self-evident that this lends itself to a combinatorial setting. This fundamental problem does not always exist for public-key cryptography since one of the keys is public. Key management of public-key cryptography involves quite different challenges, which are mainly beyond our scope.

There are many introductory texts that provide a basic primer in cryptography. For a short mathematics-free background read, we recommend [63]. For a more comprehensive coverage of techniques and methodology we highly recommend [73].

A good survey of some of the topics considered in this paper is [72]. More generally, [10] provides an excellent survey on key establishment that goes beyond areas of combinatorial interest and [18] is probably the definitive work on cryptographic protocols relating to key establishment. Finally we note that both [7] and [26] include good reviews of other combinatorial applications to problems arising in information security.

3 Key establishment framework

In this section we will propose a framework within which the various schemes that we study can be meaningfully compared. In the remainder of the paper we will review schemes that have been proposed for a range of applications within this framework.

We use the term *key establishment* to indicate that this framework primarily covers the key management processes directly related to ensuring that the right keys are established in the right places within the network. We normally assume the existence of a *trusted authority* (or *TA*), which is an entity that is regarded as trustworthy and secure by all users in the network and that is relied on for various security critical operations, in particular during initialisation. We will not be particularly concerned with operations such as key generation, which in most case we leave to the TA, and key destruction, which in most cases we need to leave to individual users.

We represent the set of *users* of our network by $\mathcal{U} = \{U_1, \ldots, U_n\}$ and the TA by $\mathcal{T}$. It is probably most intuitive if we assume that we are establishing keys in this network for confidentiality purposes (in other words our keys are encryption keys), however this need not be the case.

Let $\mathcal{C}$ be a collection of subsets of $\mathcal{U}$, which we refer to as a *communication structure*, that consists of the collection of subsets of users for whom we wish to establish common keys. Note that many treatments of key management assume that cryptographic keys only need to be established between pairs of users. We make no such restriction here and will often refer to *group keys* in order to emphasise that we are establishing keys for general subsets. A group key k_A for a set $A \in \mathcal{C}$ is a value that all members of A can compute and use to secure joint communication within the group.

Definition 3.1 Informally, a *key establishment scheme* for communication structure $\mathcal{C}$ is a set of protocols that allow any set $A \in \mathcal{C}$ to establish a group key k_A. It consists of the following operational phases:

1. **Initialisation**. In this phase $\mathcal{T}$ generates all the data required to initialise the scheme. More precisely, this comprises:

   - Secret data specific to each user. We denote the secret data specific to user U_i by u_i. This value is only known to $\mathcal{T}$ and U_i and we assume that there exists some secure channel by which u_i can be transported from $\mathcal{T}$ to U_i (this channel is regarded as something outside of the key establishment scheme and could include, for example, physical delivery).

On receiving u_i, user U_i is responsible for ensuring that u_i is suitably protected.

- Public system-wide data, which we denote by Pub. This is made available by $\mathcal{T}$ to all users in $\mathcal{U}$ by means of an authenticated channel, the details of which do not concern us here.

2. **Key establishment**. In this phase a group of users $A \in \mathcal{C}$ establish their common key k_A. Whether this process involves the TA, communication between users, or no communication between scheme entities, is a major distinguisher between schemes in this paper. We return to this issue shortly (Section 3.1).

3. **Update**. In this optional phase, the secret and public data are modified. This may be because the communication structure has changed (for example users have left the scheme or new users have joined) or because the original keys have *expired* (all cryptographic keys have a finite lifetime and eventually need to be renewed). The simplest update operation is key *refreshment*, where existing group keys are simply replaced by new keys.

In the following subsections we specify our framework by identifying issues that can be used to define specific types of key establishment scheme.

3.1 Broad classification of key establishment schemes

A major distinguisher between different key establishment schemes is the extent to which communication between entities occurs during the key establishment phase. Note that the costly secure channels between the TA and users that were employed during the initialisation phase are not normally regarded as being readily available throughout the scheme lifetime (if they were available then one easy way to establish a common key would simply be for the TA to generate one at the time of request and distribute it over these same secure channels). We identify three potential operational environments during the key establishment phase:

1. Users have no communication channels available to support key establishment and thus must be able to do so on their own. We refer to such schemes as *group key predistribution schemes*.

2. The TA has some ability to communicate with users during the key establishment phase. We refer to such schemes as *group key distribution schemes*.

3. Users have some ability to communicate with one another during the key establishment phase. We refer to such schemes as *group key agreement schemes*.

Note that these environments apply strictly to the key establishment phase. Most group key predistribution schemes, for example, require involvement of an online TA during any update phase.

3.2 Secondary distinguishers

The next set of issues are *secondary distinguishers* in the sense that they subdivide schemes within the broad categories of Section 3.1.

3.2.1 Security The security model within which a key establishment scheme operates is a secondary distinguisher. The main threat to the security of a key establishment scheme that we consider is the ability of users (or outside parties) to obtain a key that they are not entitled to. There are two different aspects to this security issue that need to be identified for any given solution:

1. **Type of security**: The most common two types of security that we will encounter are:
   - *Unconditional security*: where the security of the scheme is independent of the resources available to an attacker.
   - *Computational security*; where the scheme can only be broken by an attacker with sufficient computational resources.
2. **Resilience**: This specifies the degree of resilience of the scheme to collusion between users. We will refer to the collection $\mathcal{X}$ of subsets of $\mathcal{U}$ who, even if they collude and share all their secret data, are unable to obtain any group keys to which they are not entitled, as the *exclusion structure*. This is always a monotone decreasing set (if $B_1 \in \mathcal{X}$ and $B_2 \subseteq B_1$ then $B_2 \in \mathcal{X}$). While general exclusion structures will be considered, the two most common degrees of resilience we will encounter are:
   - *Full collusion security*: $\mathcal{X}$ consists of all subsets of $\mathcal{U}$, meaning that no collusion of users should be able to determine a key that they are not entitled to.
   - *w-security*: $\mathcal{X}$ consists of all subsets of $\mathcal{U}$ of at most size w, meaning that no collusion of up to w users should be able to determine a key that they are not entitled to.

3.2.2 Deterministic v probabilistic An important secondary distinguisher between key establishment schemes is whether they are:

- *Deterministic*: we can guarantee that a group $A \in \mathcal{C}$ is able to establish a common key.
- *Probabilistic*: we can only guarantee that a group $A \in \mathcal{C}$ is able to establish a common key with a certain probability.

3.2.3 Communication channels Schemes also differ in the types of communication channel that exists between entities involved in the scheme. Two particular types of channel that we will regularly encounter are:

- *Secure*: we assume that any information exchanged on such a channel is totally protected, both in terms of being kept confidential and authentic (unchanged and from an identified originator).
- *Broadcast*: we assume that any information exchanged on such a channel is authentic, but not confidential.

Broadcast channels are much less costly and easier to maintain than secure channels. For example, publishing some data on an authenticated public noticeboard would realise a broadcast channel.

3.2.4 Properties of keys A number of subtle secondary distinguishers concern the nature and structure of the the group keys. The following definitions will be useful in this regard. A group key k_A established by a group of users $A \in \mathcal{C}$ is:

- *Predistributed*: if k_A is a function only of the values $\{u_i \,|\, U_i \in A\}$ and Pub. In other words, k_A is computed only from data made available to the group members during the initialisation phase (this is necessarily the case for group key predistribution schemes).
- *Independent*: if knowledge of other group keys provides no information about the value of k_A.
- *Combinatorial*: if k_A can be represented as a subset of the collective secret user data of users belonging to A.

3.2.5 Extended capabilities Further secondary distinguishers arise from additional properties that may be required by specific applications. Examples include:

- **Flexibility**: the extent to which a key establishment scheme is able to efficiently accommodate an update phase.
- **Computational capability**: the extent to which entities (particularly users) have the ability to perform computations.
- **Decentralisation**: whether roles normally conducted by the TA are required to be distributed amongst a number of separate entities. This can be for reasons of scalability, security or reliability.
- **Collaboration**: the degree of collaboration that is required (or permitted) to take place between users in order to establish a group key.
- **Robustness**: a stronger security model might be required for applications where either the TA or users are not trusted to perform their operations honestly.
- **Temporal restrictions**: whether key establishment for certain groups is restricted to specific time intervals or limited to a finite number of key establishment events.
- **Traceability**: whether it is possible to identify fraudulent users who abuse the key establishment scheme.

3.3 Evaluation criteria

The previous criteria that we have discussed are largely distinguishers based on scheme functionality. The following are the most common evaluation criteria that allow comparisons to be made between functionally similar key establishment schemes.

- **Secret storage**: the amount of information that a user needs to keep secure. As secure storage is expensive, this is an important quantity to minimise.

- **Public storage**: the amount of public information that needs to be maintained in order to operate the scheme. While it not so important to reduce this as it is to reduce secret storage, maintaining authenticated public data induces a cost and keeping this as small as possible is desirable.

- **Communication costs**: the quantity of data that needs to be exchanged (whether by expensive secure channels or less expensive broadcast channels) between entities in the key establishment scheme is something we would like to minimise.

- **Computational costs**: we would like to minimise the computational requirements for users in the scheme. Efficient computation is particularly important for applications where users are represented by low-memory devices with limited computational capabilities.

4 Preliminaries

In this section we briefly review some definitions and notation that we will employ later. We refer the reader to the combinatorial literature for further details.

4.1 Designs

A *set system* $(\mathcal{I}, \mathcal{B})$ consists of a set $\mathcal{I}$ of v elements (*points*) and a collection $\mathcal{B}$ of subsets (*blocks*) of $\mathcal{I}$. The *degree* of $x \in \mathcal{I}$ is the number of blocks of $\mathcal{B}$ containing x and $(\mathcal{I}, \mathcal{B})$ is *regular* if all points have the same degree r. The *rank* k of $(\mathcal{I}, \mathcal{B})$ is the size of the largest block in $\mathcal{B}$ and we say that $(\mathcal{I}, \mathcal{B})$ is *uniform* if all blocks have size k.

A regular, uniform set system with $|\mathcal{I}| = v$, $|\mathcal{B}| = b$, and with every t points occurring on precisely λ blocks is known as a t-(v, b, r, k, λ)-*design* (we often just refer to a t-(v, k, λ)-*design* since b and r can then be uniquely derived). The following special cases are of particular interest:

- A 2-$(s^2 + s + 1, s^2 + s + 1, s + 1, s + 1, 1)$-design is known as a *projective plane*.

- A 1-(v, b, r, k, λ)-design (which by definition has $\lambda = r$) with the further property that any pair of points occur in at most one block is called a (v, b, r, k)-*configuration*.

- A t-(v, k, λ)-design whose blocks can be partitioned into parallel classes is said to be *resolvable*.

A set system is a *group-divisible design* $\mathrm{GD}(n^u, k)$ if $v = nu$ and there exists a partition $\mathcal{H}$ of $\mathcal{I}$ into u *groups* of size n such that:

1. Every $H \in \mathcal{H}$ intersects a block $B \in \mathcal{B}$ in at most one point;

2. Every pair of points from different groups occur together in precisely one block.

A *transversal design* $\mathrm{TD}(k, n)$ is a $\mathrm{GD}(n^k, k)$.

4.2 Arrays

An *orthogonal array* $\mathrm{OA}_\lambda(t, k, v)$ is a $\lambda v^t \times k$ array with entries from a set of size v such that for any tuple $(x_1, \ldots, x_t)$ and any columns $C_1, \ldots, C_t$ there are precisely λ rows of the array in which the entry x_i occurs in column C_i (for all $1 \leq i \leq t$).

4.3 Graphs

A *graph* $\mathcal{G} = (\mathcal{I}, \mathcal{E})$ consists of a set of of vertices (or *nodes*) $\mathcal{I}$ joined by *edges* in $\mathcal{E}$, where $\mathcal{E} \subseteq \mathcal{I} \times \mathcal{I}$. We say that a pair of vertices U and V are *adjacent* if $\{U, V\} \in \mathcal{E}$ (we will also say that V is a *neighbour* of U). The *degree* of a vertex U is the number of vertices adjacent to U. A graph is *regular of degree* r if all vertices have degree r. If the order of adjacent vertices $\{U, V\}$ matters then we write (U, V) (if an edge connects U to V) and we say that $\mathcal{G}$ is a *directed graph*.

A *path* of *length* L from U_0 to U_L is a sequence of edges and vertices of the form $U_0, e_1, U_1, e_2, \ldots, U_{L-1}, e_L, U_L$, where the vertices U_i and the edges e_j are all distinct and U_{i-1} and U_i are adjacent and connected by e_i. A *cycle* is a path from a vertex to itself of length more than one (a cycle of length one is called a *loop*). A graph is *connected* is every pair of vertices are joined by at least one path.

A *complete t-partite* graph is a graph whose vertices can be partitioned into t disjoint subsets such that two vertices are adjacent if and only if they belong to distinct subsets.

An (n, r, λ, μ)-*strongly regular graph* is a regular graph on n vertices with degree r and any two distinct vertices have λ common neighbours if they are adjacent and μ common neighbours if they are not adjacent.

A *tree* is a connected graph with no cycles, loops or multiple edges. There thus exists a unique path between any two vertices. Any vertex of a tree can be chosen to be the *root* of the tree, with all edges and vertices descending from this root. We call this a *rooted* tree and can interpret it is a directed graph with a natural ordering induced from the root. Every vertex U in a rooted tree (except the root) has a unique *parent* and any other vertex adjacent to U is said to be a *child* of U. Any vertex of degree one with no children is called a *leaf*. A *binary* tree is a tree where every vertex has at most two child nodes (in general an *a-ary* tree is one where every vertex has at most a child nodes). A *chain* is a tree consisting of a single path, where each intermediate vertex has precisely one parent and one child. A *starlike subgraph* is a tree in which every path has length at most two.

4.4 Posets

A *partially ordered set* (*poset*) is a pair $(\mathcal{L}, \leqslant)$, where $\leqslant$ is a reflexive, anti-symmetric, transitive binary relation on $\mathcal{L}$. We say that x *covers* y , denoted $y \lessdot x$, if $y < x$ and there does not exist $z \in \mathcal{L}$ such that $y < z < x$ (in this case we also refer to y as a *child* of x and x as a *parent* of y). The *Hasse diagram* $(\mathcal{L}, \lessdot)$ of a poset is the directed graph $(\mathcal{L}, \mathcal{E})$ where $(x, y) \in \mathcal{E}$ if and only if $x \lessdot y$. Note that every rooted tree is a Hasse diagram for the poset defined by $U \lessdot V$ if and only if U is a parent of V in the rooted tree.

4.5 Cryptographic primitives

We will use a number of cryptographic primitives as building blocks in some of the schemes in this paper. We briefly mention three that will see repeated use.

A *symmetric encryption algorithm* E is a function that converts binary strings of *plaintext* into binary strings of *ciphertext*. More precisely, the ciphertext is a function of the plaintext and a symmetric key K, which is shared between sender and receiver. The receiver of the ciphertext is able to use a related decryption algorithm to recover the plaintext from the ciphertext using the same key K. Symmetric encryption algorithms, applied directly as described, provide data confidentiality. They can be applied in other ways to establish other security services.

A *hash function* is a function that converts an arbitrary long input into a fixed length compressed output. A hash function should have the properties that it is *one-way* (it is hard to recover an input from a given output) and *collision-free* (it is hard to find two inputs with the same output, even though there will be many such pairs). Hash functions are extremely versatile cryptographic primitives and are employed widely in cryptographic protocols.

A *secret sharing scheme*, which is a method of sharing a secret value amongst a group of participants by distributing related information (*shares*) in such a way that only certain specified subsets of the participants (defined by the *access structure* Γ) can reconstruct the secret from their shares. If subsets of participants not in the access structure learn nothing about the secret from their shares then the scheme is referred to as being *perfect*. We make the natural restriction that Γ is *monotone* (in other words, if $X \in \Gamma$ and $X \subseteq Y$ then $Y \in \Gamma$). If Γ consist of all subsets of at least t out of n participants then we refer to a secret sharing scheme for Γ as being a (t, n)-*threshold scheme*.

Secret sharing schemes were first proposed in [8, 69] and are of significant combinatorial interest in their own right (see [71] for a review). It can be shown that in perfect secret sharing schemes each participant's share must be at least as large as the secret it is protecting. Secret sharing schemes in which each share has this minimal size are called *ideal*. Ideal secret sharing schemes are closely related to matroids [19] and ideal threshold schemes correspond to orthogonal arrays [39].

5 Key predistribution schemes

The first class of key establishment schemes that we will look at are group key predistribution schemes. Applications suitable for group key predistribution are those where during key establishment the TA cannot be accessed in any capacity (it may have ceased to exist, or be impractical or too costly to communicate with it) and users cannot employ secure communication channels amongst themselves (they may not be able to afford the computational costs of establishing such channels).

Definition 5.1 A $(\mathcal{C}, \mathcal{X})$-*key predistribution scheme* (KPS) is a key establishment scheme with communication structure $\mathcal{C}$ and exclusion structure $\mathcal{X}$ such that:

1. Given $A \in \mathcal{C}$, any $U_i \in A$ can compute the group key k_A from knowledge of u_i and Pub.

2. Given disjoint sets $B \in \mathcal{X}$ and $A \in \mathcal{C}$, it is not possible to compute the group key k_A from knowledge of u_B and Pub (where $u_B = \{u_i \,|\, U_i \in B\}$).

Note that the precise meaning of property (2) in Definition 5.1 depends on the security model within which we are operating. If a KPS is unconditionally secure then these conditions can be stated information theoretically (see, for example [10]).

The literature contains a wide variety of key predistribution schemes. We will begin this section by identifying a number of (generic) fundamental KPSs, most of which have manifested themselves on numerous occasions as published schemes. We then discuss several different types of KPS that are of combinatorial interest.

5.1 Fundamental schemes

In this section we identify seven fundamental key predistribution schemes, divided into two different classes. These fundamental schemes are a combination of generic schemes that help to illustrate some of our definitions as well as extremal schemes that provide useful performance benchmarks for comparison.

5.1.1 Fundamental edge-based KPSs We identify four fundamental schemes in this class. All four schemes are deterministic, have independent keys, offer full collusion security and can be established for arbitrary communication structures.

Scheme 5.2 *A* trivial key predistribution scheme *(TKPS) has the following properties:*

- $u_i = \{k_A \,|\, U_i \in A, A \in \mathcal{C}\}$*;*
- $Pub = \emptyset$*;*
- $k_A \in u_i$ *if and only if* $U_i \in A$*.*

A TKPS offers unconditional security. The most obvious problem with a TKPS is that the secret information u_i that each user has to store is potentially very large. A further problem with this type of scheme is that if group keys have to be refreshed during a key update phase then this requires the initialisation phase to be rerun.

This motivates our next fundamental scheme, where E is a secure symmetric encryption algorithm with key size l and $E_k(m)$ denotes the encryption of plaintext m using key k.

Scheme 5.3 *A* trivial key encrypting key predistribution scheme *(TKEKPS) has the following properties:*

- $u_i = \{K_A \,|\, U_i \in A, A \in \mathcal{C}\}$*, where each* K_A *is randomly chosen from* $\{0,1\}^l$*;*
- $Pub = \{E_{K_A}(k_A) \,|\, A \in \mathcal{C}\}$*;*
- $K_A \in u_i$ *if and only if* $U_i \in A$*, with* k_A *obtained by decrypting* $E_{K_A}(k_A)$*.*

A TKEKPS offers computational security, since any attacker with the computational resources to break the encryption algorithm can obtain group keys. Users have to store as much secret information as in a TKPS, but refreshing group keys can now easily be done by the TA updating Pub (more precisely, by replacing $E_{K_A}(k_A)$ by $E_{K_A}(k'_A)$, where k'_A is the refreshed version of k_A).

Our next fundamental scheme offers the minimum possible secret storage and is, in some sense, the opposite "extreme" to a TKPS.

Scheme 5.4 *A* direct key encrypting key predistribution scheme *(DKEKPS) has the following properties:*

- $u_i = k_i$, *where each* k_i *is randomly chosen from* $\{0,1\}^l$;
- $Pub = \{E_{k_i}(k_A) \,|\, U_i \in A, A \in \mathcal{C}\}$;
- $E_{k_i}(k_A) \in Pub$ *if and only if* $U_i \in A$, *with* k_A *obtained by decrypting* $E_{k_i}(k_A)$.

A TKEKPS also offers computational security. The secret storage is as small as possible, but this comes at the expense of potentially large public storage requirements (as large as the secret storage in a TKPS and TKEKPS).

Our fourth fundamental scheme is essentially a refinement of a TKEKPS that reduces the public storage at the expense of an iterated key derivation process. Let $(\mathcal{C}, \leqslant)$ be the poset induced by set containment, where for $A, B \in \mathcal{C}$, $A \leqslant B$ if and only if $B \subseteq A$. For any U_i let $roots_i = \{C \in \mathcal{C} \,|\, U_i \in C$ and $U_i \notin B$ for any $B \gtrdot C\}$.

Scheme 5.5 *An* iterative key encrypting key predistribution scheme *(IKEKPS) has the following properties:*

- $u_i = k_i$, *where each* k_i *is randomly chosen from* $\{0,1\}^l$;
- $Pub = Pub_1 \cup Pub_2$, *where* $Pub_1 = \{E_{k_i}(k_C) \,|\, C \in roots_i\}$ *and* $Pub_2 = \{E_{k_B}(k_C) \,|\, B, C \in \mathcal{C}, B \gtrdot C\}$;
- $U_i \in A$ *if and only if there exists a path (in the Hasse diagram of* $(\mathcal{C}, \leqslant)$*)* $(Z_0, Z_1), \ldots, (Z_{m-1}, Z_m)$, *where* $Z_0 \in roots_i$ *and* $Z_m = A$. *In this case* U_i *obtains* k_{Z_0} *from* Pub_1 *by decrypting* $E_{k_i}(k_{Z_0})$ *and then iteratively obtains* k_{Z_i} *from* Pub_2 *by decrypting* $E_{k_{Z_{i-1}}}(k_{Z_i})$.

Thus an IKEKPS offers computational security, has minimal secret storage and reduced public storage compared to a TKEKPS. This reduction comes at the expense of greater computational effort to iteratively derive a group key.

We refer to these four schemes as *edge-based* key predistribution schemes because they all make use, either in the public or secret data, of the set of edges in the Hasse diagram of the poset $(\mathcal{C}, \leqslant)$.

5.1.2 Fundamental node-based KPSs Our next fundamental schemes encode the structure of the poset $(\mathcal{C}, \leqslant)$ into the public information by assigning an item of public data Pub_i to each user. For this reason we refer to them as *node-based*. Unlike for the edge-based schemes, which were all distinct, the three fundamental node-based schemes are "nested", with the first being the most generic and each subsequent scheme being a special case of the previous scheme.

Scheme 5.6 *A* node-based key predistribution scheme *(NBKPS) has the following properties:*

- $Pub = \cup_{1 \leq i \leq n} Pub_i$*, where* Pub_i *is associated with user* U_i*;*
- $u_i = f(Pub_i)$ *for some secret function* f *known only to the TA, which is chosen in such a way that there exists a public function* g *such that for any* $A \in \mathcal{C}$ *and any pair* $U_i, U_j \in A$ *we have that* $g(u_i, Pub_A) = g(u_j, Pub_A) = k_A$ *(where* $Pub_A = \cup_{U_i \in A} Pub_i$*).*
- *By choice of* f *and* g *it follows that any* $U_i \in A$ *can compute* k_A*.*

Scheme 5.6 is clearly only the blueprint of a concept and precise properties of actual NBKPSs will depend on specific instances. It should be clear however that NBKPSs demand careful choice of functions and internal structure and are clearly ripe for combinatorial application.

Our next fundamental scheme represents one particular type of NBKPS. Let $\mathcal{I} = \{x_i \mid 1 \leq i \leq v\}$ be a set of v identifiers, each of which is associated by means of a secret function f with a randomly chosen key $k_i = f(x_i)$ from a set $\mathcal{K}$. Let $\mathcal{B}$ be a collection of subsets of $\mathcal{I}$. We will let $\mathcal{R} = (\mathcal{I}, \mathcal{B})$ collectively be referred to as a *key ring.*

Scheme 5.7 *A* key ring predistribution scheme *(KRPS) based on key ring* $\mathcal{R} = (\mathcal{I}, \mathcal{B})$ *is a node-based key predistribution scheme with the following properties:*

1. $Pub_i = B_i$ *is randomly chosen from* $\mathcal{B}$ *(such that* $u_i \neq u_j$ *if* $i \neq j$*);*
2. $u_i = \{k_j \mid x_j \in B_i\}$*;*
3. $\mathcal{C} \subseteq \{A \subseteq \mathcal{U} \mid \cap_{U_i \in A} u_i \neq \emptyset\}$*;*
4. *For* $A \in \mathcal{C}$*, group key* $k_A = g(\cap_{U_i \in A} u_i)$ *for some public combining function* g*. In other words, a group* $A = \{U_1, \ldots, U_t\} \in \mathcal{C}$ *of users check their public identifier sets* $Pub_1, \ldots, Pub_t$ *to see which common identifiers they share. They then establish a group key* k_A *by applying* g *to the keys* k_i *that correspond the identifiers in* $\cap_{j=1}^{t} Pub_j$*.*

KRPSs are examples of group key establishment schemes with combinatorial keys (see Section 3.2.4). Whether they offer unconditional or computational security depends on the combining function g. For example, an unconditionally secure scheme can be obtained if $k_A = \oplus_{k_i \in X} k_i$, where $X = \cap_{U_i \in A} u_i$.

Our final fundamental node-based scheme is a particular type of KRPS.

Scheme 5.8 *A* random key predistribution scheme *(RKPS) is a key ring predistribution scheme based on key ring* $\mathcal{R} = (\mathcal{I}, \mathcal{B})$*, where* $\mathcal{B} = 2^{\mathcal{I}}$ *(the collection of all subsets of* $\mathcal{I}$*).*

In other words, an RKPS involves issuing each user with a set of random keys from $\mathcal{K}$. An RKPS is thus an example of a probabilistic key establishment scheme. This may seem like a very strange way of constructing a KPS for a specific communication structure $\mathcal{C}$, since it involves "getting lucky" with regard to the intersection

properties of the resulting blocks. However the idea behind a KRPS can be useful in situations where certain properties of a KPS (such as user storage) have higher priority than establishing a desired communication structure precisely (we will see examples of such applications in Section 7.3).

5.2 The BDVHKY scheme

In this section we present an important benchmark NBKPS. If $\mathcal{C} = \{A \subseteq \mathcal{U} \,|\, |A| = t\}$ and $\mathcal{X} = \{A \subseteq \mathcal{U} \,|\, |A| \leq w\}$ then we will also refer to a $(\mathcal{C}, \mathcal{X})$-KPS as a (t, w)-KPS. We will also refer to communication structures $\mathcal{C}$ of this type as *threshold* communication structures. The following (t, w)-KPS was proposed by Blundo, De Santis, Vaccaro *et al* in [17] and is a generalisation of a much earlier $(2, w)$-KPS proposed in [9].

Scheme 5.9 *The* BDVHKY key predistribution scheme *(BDVHKY-KPS) is defined as follows, where $q \geq n$:*

- *$Pub_i = s_i$, where $s_i \in GF(q)$ and $Pub_i \neq Pub_j$ if $i \neq j$;*
- *The TA (randomly) constructs a secret t-variate polynomial f with coefficients from $GF(q)$,*

$$f(x_1, \ldots, x_t) = \sum_{i_1=0}^{w} \cdots \sum_{i_t=0}^{w} a_{i_1 \ldots i_t} x_1^{i_1} \ldots x_t^{i_t},$$

  where $a_{i_1 \ldots i_t} = a_{j_1 \ldots j_t}$ for any permutation $(j_1 \ldots j_t)$ of the indices $\{i_1, \ldots, i_t\}$.
- *$u_i = f(Pub_i, x_2, \ldots, x_t) = f(s_i, x_2, \ldots, x_t)$, a $(t-1)$-variate polynomial with coefficients from $GF(q)$;*
- *For any $A = \{U_{z_1}, \ldots, U_{z_t}\} \in \mathcal{C}$, the user U_{z_i} computes*

$$k_A = u_{z_i}(s_{z_1}, \ldots, s_{z_{i-1}}, s_{z_{i+1}}, \ldots, s_{z_t}) = f(s_{z_1}, \ldots, s_{z_t}).$$

The BDVHKY-KPS is an example of a deterministic NBKPS that is not a KRPS. It offers unconditional w-security. We note that in the BDVHKY-KPS, each user needs to store a secret $t-1$ variate polynomial of degree w of a special form. It can be shown that this involves the equivalent of storing $\binom{t+w-1}{t-1}$ elements of $GF(q)$. The BDVHKY-KPS is of particular interest because it is shown in [17] that this is the optimally small user storage for any unconditionally secure (t, w)-KPS.

The following variant of Scheme 5.9 is a generalisation of a scheme proposed in [52], which uses the random key predistribution scheme (Scheme 5.8) to obtain some interesting tradeoffs.

Scheme 5.10 *The* randomised BDVHKY-KPS *is similar to Scheme 5.9 except that:*

- *The TA (randomly) constructs r secret t-variate polynomials $f_1, \ldots, f_r$ with coefficients from $GF(q)$, each with the property required for Scheme 5.9;*
- *For each U_i, the TA generates a random subset $U[i] = \{i_1, \ldots, i_{r'}\}$ of the set $\{1, \ldots, r\}$, which is made public;*

- $u_i = \{f_{i_1}(s_i, x_2, \ldots, x_t), \ldots, f_{i_{r'}}(s_i, x_2, \ldots, x_t)\}$;
- *For any* $A = \{U_{z_1}, \ldots, U_{z_t}\} \in \mathcal{C}$, *if* $\cap_{j=1}^{t} U[z_j] \neq \emptyset$ *then for some* $l \in \cap_{j=1}^{t} U[z_j]$ *user* U_{z_i} *computes* $k_A = f_l(s_{z_1}, \ldots, s_{z_t})$. *(Note that since the sets* $U[i]$ *are public, the choice of* l *can be publicly predetermined.)*

Scheme 5.10 is an example of a probabilistic NBKPS, since there is no guarantee that $\cap_{j=1}^{t} U[z_j] \neq \emptyset$. Compared to the BDVHKY-KPS, the scheme also involves an increased user storage by a magnitude of r'. However the significant gain is in resilience. The BDVHKY-KPS is only w-secure, whereas in [52] it is shown that careful selection of the parameters r and r' in Scheme 5.10 can result in very good resilience.

We will discuss a further variant of the BDVHKY-KPS in Section 7.3.4. We note that in [62] it was shown that a number of key predistribution schemes, including Scheme 5.9 (under certain constraints on the combining function used to determine the final key), are examples of a wider family of *linear key predistribution schemes*, which can be described in linear algebraic terms and permit an inherent duality.

5.3 Key distribution patterns

In this section we look at an interesting family of key ring predistribution schemes that have arisen in the literature in a number of different guises.

Definition 5.11 Let $(\mathcal{C}, \mathcal{X})$ be a communication and exclusion structure defined on n users. A $(\mathcal{C}, \mathcal{X})$-*key distribution pattern* (KDP) is a set system $(\mathcal{I}, \mathcal{B})$ with $|\mathcal{B}| = n$, where each user U_i is associated with a block B_i, such that for any disjoint pair $A \in \mathcal{C}$ and $B \in \mathcal{X}$ we have:

$$\bigcap_{U_i \in A} B_i \not\subseteq \bigcup_{U_j \in B} B_j.$$

When $\mathcal{C}$ consists of all t-subsets of users and $\mathcal{X}$ consists of all subsets of at most w users, we will refer to a (t, w)-KDP. In [76], (t, w)-KDPs were noted to correspond to the following more granularly defined family of set systems:

Definition 5.12 A (t, w, d)-*cover-free family* (CFF) is a set system $(\mathcal{I}, \mathcal{B})$ such that for any disjoint sets of t blocks A and w blocks B we have:

$$|\bigcap_{B_i \in A} B_i \setminus \bigcup_{B_j \in B} B_j| \geq d.$$

The motivation for Definition 5.11 is that a KDP can be used as a key ring to form a KRPS.

Scheme 5.13 *A* $(\mathcal{C}, \mathcal{X})$-key distribution pattern predistribution scheme *(KDPPS) is a* $(\mathcal{C}, \mathcal{X})$*-KRPS that arises by applying Scheme 5.7 with a* $(\mathcal{C}, \mathcal{X})$*-KDP as the key ring.*

KDPs were first introduced in [56, 57], where (t, w)-KDPs were proposed and analysed. These structures have subsequently been investigated by a number of authors who have investigated bounds and constructions for efficient KDPs, particularly of uniform (t, w)-KDPs of rank k. We now briefly mention some of the work that has been undertaken on KDP constructions and KDP efficiency.

5.3.1 KDP constructions We first define two fundamental KDPs.

Scheme 5.14 *Given a communication structure $\mathcal{C}$ defined on a user set $\mathcal{U}$, a* $(\mathcal{C}, 2^{\mathcal{U}})$-trivial inclusion KDP *(TIKDP) is defined as follows.*

- *For each $A \in \mathcal{C}$, associate a point $x_A \in \mathcal{I}$;*
- *For each user $U_i \in \mathcal{U}$, define a block $B_i = \{x_A \,|\, U_i \in A\}$.*

Given any $A \in \mathcal{C}$, the blocks B_j such that $U_j \in A$ have the unique point x_A in common. As no B disjoint from A contains x_A, we see that $(\mathcal{I}, \mathcal{B})$ is a $(\mathcal{C}, 2^{\mathcal{U}})$-KDP.

Note that the KDPPS arising from applying a TIKDP in Scheme 5.13 is essentially Scheme 5.2, the trivial KPS.

Scheme 5.15 *Given an exclusion structure $\mathcal{X}$ defined on a user set $\mathcal{U}$, a* $(2^{\mathcal{U}}, \mathcal{X})$-trivial exclusion KDP *(TEKDP) is defined as follows.*

- *For each $B \in \mathcal{X}$, associate a point $x_B \in \mathcal{I}$;*
- *For each user $U_i \in \mathcal{U}$, define a block $B_i = \{x_B \,|\, U_i \notin B\}$.*

Given any subset $B \in \mathcal{X}$, none of the blocks B_j such that $U_j \in B$ contain point $x_{\mathcal{U}\setminus\mathcal{B}}$, and thus $(\mathcal{I}, \mathcal{B})$ is a $(2^{\mathcal{U}}, \mathcal{X})$-KDP.

The TEKDP for the case where $\mathcal{X}$ consists of all subsets of users of size at most w was first defined in [35].

Both Scheme 5.14 and Scheme 5.15 result in users potentially having to store a large amount of secret data. A number of combinatorial objects have been used to construct (t, w)-KDPs that perform much better than these fundamental KDPs.

- In [72] it is shown that a $(t+1)$-(n, k, λ) design with $w < (n-t)/(k-t)$ is the dual of a (t, w)-KDP.
- In [60], [61] and [67] special finite geometrical structures have been used to construct KDPs.
- In [64] KDPs were constructed from conics arising from finite projective planes and affine planes.
- In [74], KDPs are defined from orthogonal and perpendicular arrays.

We note that in [32] a non-constructive existence result for very efficient (t, w)-KDPs was proven which, when applied to Scheme 5.13, generates a KDPPS that is essentially a manifestation of the RKPS.

5.3.2 Efficiency of KDPs Given a fixed number of users n we are particularly interested in trying to find KDPs of low rank (small block size), since the resulting KDPPS produced using Scheme 5.13 will have relatively low user storage. A different (but related) optimisation problem is to minimise v, which corresponds to the number of different keys in the system. In [65] several lower bounds on the information storage of KDPs were determined. Subsequently several bounds on (t, w, d)-cover free families have been proven in [76]. These all indicate that, in general, KDPPSs are not particularly efficient. However there are several generic techniques in which KDPPSs can be made more efficient. One such technique was proposed in [72], based on the following concept:

Definition 5.16 An (n, m, t, q)-*resilient function* is a function $f : [GF(q)]^n \rightarrow GF(q)$ such that if t input bits are fixed and the remaining $n - t$ chosen independently at random, then every possible element of $GF(q)$ occurs as output with equal probability.

Let $(\mathcal{I}, \mathcal{B})$ be a $(\mathcal{C}, \mathcal{X})$-KDP. For any $A \in \mathcal{C}$ let $I_A = \cap_{U_i \in A} B_i$. Denote $c_A = |I_A|$ and $d_A = \max\{|I_A \cap B| \,|\, B \in \mathcal{X} \text{ and } A \cap B = \emptyset\}$. In other words, each set A in the communication structure is associated with at least $c_A - d_A$ identifiers (keys) that are unknown to any disjoint set in the exclusion structure. The following refinement to Scheme 5.13 was observed in [72].

Scheme 5.17 *Let $(\mathcal{I}, \mathcal{B})$ be a $(\mathcal{C}, \mathcal{X})$-KDP and $m = \min\{c_A - d_A \,|\, A \in \mathcal{C}\}$.*

1. *For each $A \in \mathcal{C}$ choose a public (c_A, m, d_A, q)-resilient function f_A. (Such a function always exists for suitable large q [72].)*

2. *Now construct a $(\mathcal{C}, \mathcal{X})$-KDPPS by applying Scheme 5.13 with the $(\mathcal{C}, \mathcal{X})$-KDP as the key ring and f_A as the public combining function for group key k_A. (In other words, using the notation of Scheme 5.7, $k_A = f_A(\cap_{U_i \in A} u_i)$.)*

A KPS arising from a KDP is likely to benefit from the refinement proposed in Scheme 5.17 if the KDP has a relatively high value of m (or, in the case of (t, w)-KDPs, if the KDP is a (t, w, d)-CFF for a high value of d). In [74] some (t, w)-KDPs were constructed from orthogonal and perpendicular arrays that lend themselves to this improvement and result in KPSs with good user storage. In [65] an alternative technique for improving the efficiency of a KDP was proposed, based on the idea of using an *information map* to reduce the information content of the keys k_i held by each user.

5.4 Hash-tree key predistribution schemes

We now describe a family of key predistribution schemes whose security is based on repeated iterations of a cryptographic hash function (see Section 4.5). In a similar manner to the construction of KDPPSs from KDPs, we first define a combinatorial object (in this case an array) from which KPSs can be generated. Recall from Section 4.4 that a rooted tree $\mathcal{T}$ has a natural partial ordering $\leqslant$ defined by parenthood.

Definition 5.18 Let $(\mathcal{C}, \mathcal{X})$ be a communication and exclusion structure defined on n users. Let $\mathcal{T}$ be a rooted tree with vertices labelled by $\{0, 1, \ldots, d-1\}$ and let b be a positive integer. A $(\mathcal{C}, \mathcal{X}, \mathcal{T})$-*hash-tree key predistribution pattern* (HTKDP) is a $b \times n$ matrix $M = (\alpha_{ij})$, where each column is associated with a unique user, with entries from $\{0, 1, \ldots, d-1\}$, such that for any disjoint $A \in \mathcal{C}$ and $B \in \mathcal{X}$ there exists indices (i^{AB}, j^{AB}) such that $j^{AB} \in A$ and:

1. $\alpha_{(i^{AB})j} \leqslant \alpha_{(i^{AB})(j^{AB})}$ for all $j \in A$;
2. $\alpha_{(i^{AB})j} \not\leqslant \alpha_{(i^{AB})(j^{AB})}$ for all $j \in B$.

In [49] it was shown how a KPS can be constructed from a $(\mathcal{C}, \mathcal{X}, \mathcal{T})$-HTKDP.

Scheme 5.19 *Let the $b \times n$ matrix $M = (\alpha_{ij})$ with entries from $\{0, 1, \ldots, d-1\}$ be a $(\mathcal{C}, \mathcal{X}, \mathcal{T})$-HTKDP. A* $(\mathcal{C}, \mathcal{X}, \mathcal{T})$-hash-tree key predistribution scheme *(HTKPS) can be constructed from M and a suitable hash function h as follows:*

1. *The TA publishes M, $\mathcal{T}$ and h as public system parameters.*
2. *For each $1 \leq i \leq b$ the TA chooses a secret random seed value s_i^0. For each $1 \leq j \leq d-1$ a hash value can then be iteratively computed such that if j is a child of l in $\mathcal{T}$ then $s_i^j = h(s_i^l, j)$.*
3. *The TA securely delivers $u_j = \{s_1^{\alpha_{1j}}, \ldots, s_b^{\alpha_{bj}}\}$ to user U_j.*
4. *For $A \in \mathcal{C}$, define*

$$I_A = \{1 \leq i \leq b \mid \text{ there exists } m_j \in A \text{ such that } \alpha_{ij} \leqslant \alpha_{im_j} \text{ for all } j \in A\}.$$

Then $k_A = \sum_{i \in I_A} s_i^{\alpha_{im_j}}$.

Thus we see that in an HTKPS, any user U_l belonging to A can compute k_A since for each $i \in I_A$ they can iteratively compute $s_i^{\alpha_{im_j}}$ from their component $s_i^{\alpha_{il}}$ of u_l. On the other hand, for any $B \in \mathcal{X}$, Definition 5.18 guarantees that $\alpha_{(i^{AB})m_j} \not\leqslant \alpha_{(i^{AB})(j^{AB})}$. Thus $s_i^{\alpha_{(i^{AB})m_j}}$, and hence k_A, cannot be computed by any user in B.

Scheme 5.19 is thus a deterministic KPS that offers computational security, since the security of group keys k_A depends on the security of the underlying hash function.

Following the convention of previous sections, we will refer to a $(t, w, \mathcal{T})$-HTKDP and $(t, w, \mathcal{T})$-HTKPS respectively when $\mathcal{C}$ consists of all t-subsets of users and $\mathcal{X}$ consists of all subsets of at most w users.

Example 5.20 Let $\mathcal{T}$ be a starlike tree with 7 leaves (where the centre is labelled 0 and the leaves labelled $1, \ldots, 7$). The following $(2, 2, \mathcal{T})$-HTKDP on 7 users was given in [49]:

$$M = \begin{array}{ccccccc} 0 & 0 & 3 & 0 & 5 & 6 & 7 \\ 1 & 0 & 0 & 4 & 0 & 6 & 7 \\ 0 & 2 & 0 & 4 & 5 & 0 & 7 \\ 1 & 2 & 3 & 0 & 0 & 0 & 7 \\ 0 & 2 & 3 & 4 & 0 & 6 & 0 \\ 1 & 0 & 3 & 4 & 5 & 0 & 0 \\ 1 & 2 & 0 & 0 & 5 & 6 & 0 \end{array}.$$

To construct a $(2, 2, \mathcal{T})$-HTKPS, the TA first generates secret seeds $s_1^0, \ldots, s_7^0$. Seven copies of $\mathcal{T}$ are then labeled with iterations of h as indicated in Figure 1. User U_1

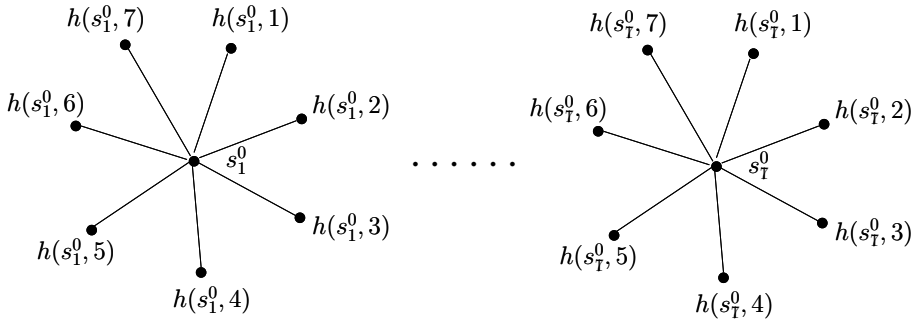

Figure 1: Hash iterations based on the starlike tree with 7 leaves

then receives:

$$\begin{aligned} u_1 &= \{s_1^{\alpha_{11}}, s_2^{\alpha_{21}}, s_3^{\alpha_{31}}, s_4^{\alpha_{41}}, s_5^{\alpha_{51}}, s_6^{\alpha_{61}}, s_7^{\alpha_{71}}\} \\ &= \{s_1^0, s_2^1, s_3^0, s_4^1, s_5^0, s_6^1, s_7^1\} \\ &= \{s_1^0, h(s_2^0, 1), s_3^0, h(s_4^0, 1), s_5^0, h(s_6^0, 1), h(s_7^0, 1)\}. \end{aligned}$$

Similarly, U_2 receives:

$$u_2 = \{s_1^0, s_2^0, h(s_3^0, 2), h(s_4^0, 2), h(s_5^0, 2), s_6^0, h(s_7^0, 2)\}.$$

The group key $k_{\{U_1,U_2\}}$ is constructed by first noting that $I_{\{U_1,U_2\}} = \{1, 2, 3, 5, 6\}$ and thus that $k_{\{U_1,U_2\}} = s_1^0 + h(s_2^0, 1) + h(s_3^0, 2) + h(s_5^0, 2) + h(s_6^0, 1)$.

There are three special cases worth mentioning.

1. If $\mathcal{T}$ degenerately consists of just one vertex then a $(\mathcal{C}, \mathcal{X}, \mathcal{T})$-HTKDP is a $(\mathcal{C}, \mathcal{X})$-KDP, as defined in Section 5.3.

2. Scheme 5.19 was motivated by an earlier scheme in [50] that constructed a $(2, w, \mathcal{T})$-HTKDP, where $\mathcal{T}$ was a chain and the underlying matrix M was generated randomly, resulting only in a probabilistic scheme.

3. In [66] a further variant (called HARPS) was proposed for use in wireless sensor networks (see Section 7.3). This combines the scheme of [50] and the idea behind the RKPS (Scheme 5.8) by only allocating to each user a value on a random subset of the b hash chains (instead of all the chains). This reduces user storage at the expense of a poorer probability that a group will be able to construct a group key.

5.5 Key assignment schemes

A very interesting class of key predistribution schemes arise from what are known as information flow policies. These have largely been investigated by researchers in computer security since they define a type of access control mechanism, but they

can be considered as a class of group key predistribution schemes. The technique of key predistribution is particularly appropriate for this type of application because it allows an information flow policy to be applied "seamlessly", without users even necessarily being aware that their actions are being controlled using this type of scheme. Of particular theoretical interest is that these schemes provide naturally arising examples of non-threshold communication structures.

Definition 5.21 An *information flow policy* is a tuple $(\mathcal{L}, \mathcal{E}, \mathcal{S}, \mathcal{O}, \lambda)$, where:

- $(\mathcal{L}, \mathcal{E})$ is a directed graph of security labels (see Figure 2);
- $\mathcal{S}$ is a set of subjects (perhaps users of a computer system);
- $\mathcal{O}$ is a set of objects (perhaps computer files);
- $\lambda : \mathcal{S} \cup \mathcal{O} \rightarrow \mathcal{L}$ is a security function that associates subjects and objects with security labels.

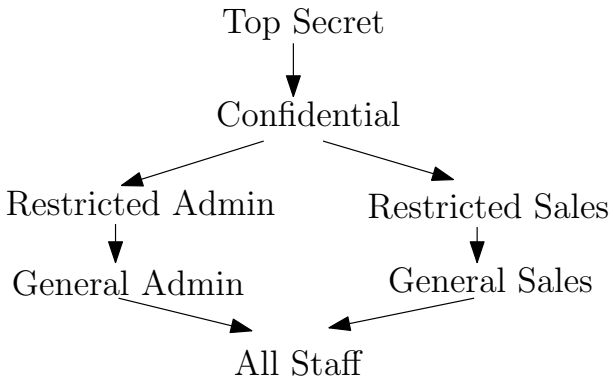

Figure 2: Directed graph of security labels

An information flow policy is used to model the access of subjects in $\mathcal{S}$ to a set of objects in $\mathcal{O}$ in a hierarchical system, where the directed graph indicates when a subject can read an object. More precisely, subject S can read object O if and only if $(\lambda(S), \lambda(O)) \in \mathcal{E}$. One way of implementing this policy is to use what is known as a key assignment scheme.

Scheme 5.22 *A* key assignment scheme *for information flow policy* $(\mathcal{L}, \mathcal{E}, \mathcal{S}, \mathcal{O}, \lambda)$ *is a scheme initialised by a TA as follows:*

- *The TA identifies each label* $x \in \mathcal{L}$ *with a cryptographic key* k_x.
- *For each label* $x \in \mathcal{L}$ *the TA generates secret information* $\sigma(x)$ *and securely distributes it to all subjects with security label* x.
- *The TA generates some system-wide public data* Pub *that is made available to all subjects using an authenticated channel.*
- *There exists a function that takes as input labels* $x, y \in \mathcal{L}$, $\sigma(x)$ *and* Pub *and outputs* k_y *if and only if* $(x, y) \in \mathcal{E}$.

A key assignment scheme implements the information flow policy since k_y can be used to encrypt objects with security label y, and only subjects with label x, where $(x, y) \in \mathcal{E}$ can compute k_y and hence decrypt the encrypted object.

It is worth observing at this stage that the vast majority of information flow policies, and hence key assignment schemes, are defined for hierarchies where the security labels in $\mathcal{L}$ form a poset. Not only are such *poset-based* schemes easier to design, but they are also by far the most natural policies to implement in real applications. In this case we can represent the policy by $(\mathcal{L}, \leqslant, \mathcal{S}, \mathcal{O}, \lambda)$ (more commonly just denoted by $(\mathcal{L}, \leqslant)$ when the context is obvious). In this case subject S can read object O if and only if $\lambda(S) \geqslant \lambda(O)$.

For any $y \in \mathcal{L}$, let $\uparrow y = \{x \in \mathcal{L} \,|\, (x, y) \in \mathcal{E}\}$ and $\downarrow y = \{z \in \mathcal{L} \,|\, (y, z) \in \mathcal{E}\}$. The following result is immediate from the relevant definitions.

Theorem 5.23 *A* key assignment scheme *for information flow policy* $(\mathcal{L}, \mathcal{E}, \mathcal{S}, \mathcal{O}, \lambda)$ *is a* $(\mathcal{C}, \mathcal{X})$*-key predistribution scheme where:*

- $\mathcal{U} = \mathcal{L}$*;*
- $\mathcal{C} = \{\uparrow y \,|\, y \in \mathcal{L}\}$*;*
- $\mathcal{X}$ *is inherited from the degree of collusion security of the underlying key assignment scheme.*
- $u_x = \sigma(x)$*;*
- *Pub is the same as for the key assignment scheme;*
- *For* $A = \uparrow y \in \mathcal{C}$*,* $k_A = k_y$*.*

A key assignment scheme can thus be thought of as a special type of deterministic computationally secure KPS, where there are as many groups in the communication structure as there are users, and where the groups can be derived from the vertices of a directed graph defined on the set of users. Note that it is perhaps more appropriate to consider a key assignment scheme as a KPS defined on the set $\mathcal{S}$ of subjects. In this case each subject S with security label $\lambda(S) = x$ is given the same piece of secret information $\sigma(x)$ in the resulting KPS. A subject S is thus able to compute all the group keys $k_{\uparrow y}$ for each $y \in \downarrow x$.

A review of key assignment schemes can be found in [27]. In the remaining sections we provide examples of some of the techniques used to construct them.

5.5.1 Unconditionally secure key assignment We first observe that unconditionally secure key assignment schemes are not very interesting from either a theoretical or a practical perspective. One obvious example is the *trivial key assignment scheme* (TKAS) based on letting $\sigma(x) = \{k_y \,|\, (x, y) \in \mathcal{E}\}$, which gives rise to Scheme 5.2 when interpreted as a KPS. We have already observed in Section 5.1 that such a scheme has unacceptably high secret storage. The unconditional secure setting was modelled formally in [34] and it was shown that the TKAS is essentially the best possible (more precisely it was shown that it can only be slightly improved by first compressing the representation of the information flow policy and then generating a TKAS for this slightly simpler policy). As a result, the only key assignment schemes of real interest are necessarily computationally secure.

5.5.2 Key assignment for poset policies An extraordinary variety of key assignment schemes for poset policies have been proposed in the literature. In [28] it was shown that they all fall into five broad classes. When interpreted as KPSs, these five classes coincide with five of the fundamental KPSs identified in Section 5.1. As the majority are either IKEKPSs (Scheme 5.5) or NBKPSs (Scheme 5.6) we will present one example from each of these classes here. While these schemes are not strictly combinatorial, the fact that they implement communication structures with interesting combinatorial structure merits their inclusion in this review.

The following key assignment scheme was first proposed in [1] (our version is based on an observation in [28]) and gives rise to a NBKPS.

Scheme 5.24 *The* Akl-Taylor *key assignment scheme (ATKAS) for a poset-based policy* $(\mathcal{L}, \leqslant)$ *is defined as follows:*

- *Let* $n = pq$ *be the product of two large primes and* $m \in Z_n^*$ *(all subsequent calculations are modulo* n*). The value* n *is public, but* p*,* q *and* m *are kept secret by the TA.*
- *For each* $x \in \mathcal{L}$*, let* $Pub_x = p_x$*, where* p_x *is a small prime and* $Pub_x \neq Pub_y$ *if* $x \neq y$ *(it suffices for* $\{p_x \mid x \in \mathcal{L}\}$ *to be chosen to be the first* $|\mathcal{L}|$ *primes). Let* $Pub = \cup_{x \in \mathcal{L}} Pub_x$*.*
- *For each* $x \in \mathcal{L}$*, let* $e(x) = \prod_{y \not\leqslant x} p_x$ *and* $\sigma(x) = k_x = m^{e(x)}$*.*
- *If* $y \leqslant x$ *then given* x*,* y*,* $\sigma(x) = k_x$ *and* Pub*, we can calculate* k_y *as follows:*

$$k_y = k_x^{p(x,y)}, \text{ where } p(x,y) = \prod_{z \in (\mathcal{L} \setminus \downarrow y) \setminus (\mathcal{L} \setminus \downarrow x)} p_z.$$

Thus the ATKAS uses a public labeling of the nodes of the poset $(\mathcal{L}, \leqslant)$ to generate a set of exponents $e(x)$ that have the property that $e(x)|e(y)$ if and only if $y \leqslant x$. This allows keys k_x associated with a higher level in the poset to compute keys k_y at lower levels. If $y \not\leqslant x$ then it is impossible to compute k_y from k_x without knowledge of m. Calculating m from any k_x is believed to be a hard computational problem known as the *RSA problem* (see, for example [73], for more information about the RSA cryptosystem on which this is based). In fact it is possible to show that any collusion of nodes cannot determine a key that they are not entitled to, assuming that the RSA problem is hard, and so the ATKAS (and thus its resulting KPS) is computationally secure with full collusion security.

There have been many variants of the ATKAS proposed (for example [37, 53]) and [27] contains a comprehensive list. Most of these either attempt to optimise the poset labelling in some way or change its performance with respect to an update phase. The principle behind all these schemes remain the same.

The next key assignment scheme was proposed by [4] and gives rise to an IKEKPS.

Scheme 5.25 *The* AFB *key assignment scheme (AFBKAS) for a poset-based policy* $(\mathcal{L}, \leqslant)$ *is defined as follows:*

- *Let h be a one-way hash function such that $h\colon \{0,1\}^* \to \{0,1\}^l$ for some integer l.*
- *For each $x \in \mathcal{L}$, let $\sigma(x) = k_x$ be randomly selected from $\{0,1\}^l$.*
- *For each $x \in \mathcal{L}$, let $Pub = \{k_z - h(k_x, z) \mid z \lessdot x\}$.*
- *If $y \leqslant x$ then given x, y, $\sigma(x) = k_x$ and Pub, we can calculate k_y since there exists a path $(z_0, z_1), \ldots, (z_{m-1}, z_m)$, where $z_0 = x$ and $z_m = y$. The key k_{z_i} can be iteratively obtained from $k_{z_{i-1}}$ and Pub by computing $h(k_{z_{i-1}}, z_i)$, from which $k_{z_i} = (k_{z_i} - h(k_{z_{i-1}}, z_i)) + h(k_{z_{i-1}}, z_i)$.*

To see that the KPS arising from Scheme 5.25 is an IKEKPS (Scheme 5.5), we observe that there exists an isomorphism between the poset $(\mathcal{L}, \leqslant)$ and the poset $(\mathcal{C}, \leqslant^*)$ associated with the resulting KPS, resulting in the following correspondences between Scheme 5.5 and Scheme 5.25:

- $\uparrow x \in \mathcal{C}$ corresponds to $x \in \mathcal{L}$;
- $roots_x$ for security label x correspond to $\{x\}$;
- u_x in Scheme 5.5 corresponds to k_x;
- $k_{\uparrow x}$ in Scheme 5.5 also corresponds to k_x;
- If $(\uparrow z) \lessdot^* (\uparrow x)$ then $E_{k_{\uparrow z}}(k_{\uparrow x})$ in Scheme 5.5 is defined by $k_z - h(k_x, z)$.

Note that Pub_1 in Scheme 5.5 corresponds to $\{k_x - h(k_x, x) \mid x \in \mathcal{L}\}$. This serves no purpose, as it is essentially an encryption of key k_x using key k_x and therefore has been omitted from the description of Scheme 5.25. (In fact Pub_1 is redundant in any KPS arising from a poset-based key assignment scheme.)

There have been many proposals for key assignment schemes that give rise to IKEKPs, for example [51, 80, 81].

5.5.3 Key assignment for directed graphs We have already observed that most key assignment schemes are designed for information flow policies based on posets. It is at least of theoretical interest to investigate schemes for general information flow policies (general directed graphs).

One method of constructing a key assignment scheme for a general information flow policy is to embed the policy into a poset and then use a poset-based key assignment scheme. In [68] such an embedding was exhibited that enables the poset-based scheme of Akl-Taylor [1] to be extended to a general information flow policy. The majority of poset-based key assignment schemes are *simple*, which means that for any $x \in \mathcal{L}$ we have $\sigma(x) = k_x$. The embedding of [68] works by embedding $(\mathcal{L}, \mathcal{E})$ in a poset $(\mathcal{L}^*, \leqslant)$, creating a simple Akl-Taylor poset-based key assignment scheme for $(\mathcal{L}^*, \leqslant)$, and interpreting this as a non-simple scheme for $(\mathcal{L}, \mathcal{E})$. In [27] it is shown that this *De Santis decoupling*, presented as Scheme 5.26, can be applied to any simple poset-based key assignment scheme.

Scheme 5.26 *The* De Santis decoupling *generates a key assignment scheme for the information flow policy $(\mathcal{L}, \mathcal{E})$ as follows:*

- *Define a poset* $(\mathcal{L}^*, \leqslant)$*, where*
  - $\mathcal{L}^* = \{x_l \,|\, x \in \mathcal{L}\} \cup \{x_u \,|\, x \in \mathcal{L}\}$,
  - $x_l \leqslant x_u$,
  - $(y, x) \in \mathcal{E}$ *implies* $y_l \leqslant x_u$.
- *Establish any simple poset-based key assignment scheme for* $(\mathcal{L}^*, \leqslant)$*, with key* k^*_x *for each* $x \in \mathcal{L}^*$.
- *Interpret this as a key assignment scheme for* $(\mathcal{L}, \mathcal{E})$ *where* $k_x = k^*_{x_l}$ *and* $\sigma(x) = k^*_{x_u}$.

An illustration of how the De Santis decoupling works is shown in Figure 3. The upper row of nodes in $(\mathcal{L}^*, \leqslant)$ represents $a_u, \ldots, f_u$, while the lower row of nodes represents $a_l, \ldots, f_l$.

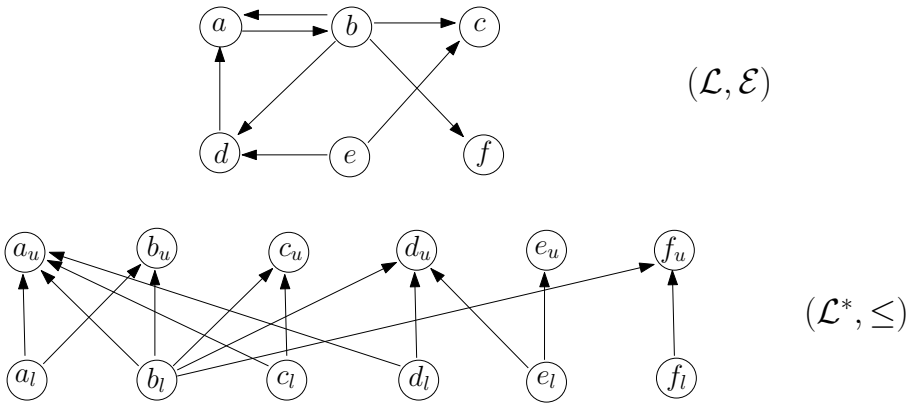

Figure 3: The construction of $(\mathcal{L}^*, \leqslant)$ from $(\mathcal{L}, \mathcal{E})$

6 Group key distribution schemes

Our next class of key establishment schemes, group key distribution schemes, are appropriate for applications where it is possible (and practical) to communicate in some way with a trusted entity throughout the lifetime of the scheme. This scenario is desirable for applications where group keys k_A are necessarily generated at the time of request (not during the initialisation phase as is the case for most key predistribution schemes).

Definition 6.1 A $(\mathcal{C}, \mathcal{X})$*-key distribution scheme* (KDS) is a key establishment scheme with communication structure $\mathcal{C}$ and exclusion structure $\mathcal{X}$ such that:

1. Given $A \in \mathcal{C}$, any $U_i \in A$ can compute the group key k_A from knowledge of u_i and $v_{i,A}$, where $v_{i,A}$ is some information obtained by U_i from the TA during the key establishment phase for key k_A.

2. Given disjoint sets $B \in \mathcal{X}$ and $A \in \mathcal{C}$, it is not possible to compute the group key k_A from knowledge of u_B and v_B (where $u_B = \{u_i \,|\, U_i \in B\}$ and $v_B = \{v_{i,A} \,|\, U_i \in B\}$).

Note that Definition 6.1 includes the case where the secure channels that were used to distribute the initial user secret data u_i are still available and $v_{i,A} = k_A$ if $U_i \in A$, otherwise $v_{i,A} = \emptyset$. This "trivial" solution is in fact one that is often adopted in real applications where such secure channels exist throughout the scheme lifetime, however it is of little mathematical interest and so we do not consider it further here.

6.1 Broadcast encryption

We now look at a well-studied family of group key distribution schemes where, although the TA is online during the key establishment phase, it no longer maintains secure channels to the users and must rely on broadcast channels to establish group keys.

Definition 6.2 A $(\mathcal{C}, \mathcal{X})$-*broadcast encryption scheme* (BES) is a key distribution scheme with communication structure $\mathcal{C}$ and exclusion structure $\mathcal{X}$ such that $v_{i,A} = B_A$ for every user $U_i \in \mathcal{U}$, where B_A is a public message broadcast to all users in $\mathcal{U}$ at the start of the key establishment phase for k_A.

Broadcast encryption schemes were first proposed with applications such as access to streamed multimedia services in mind. In this type of application some digital content, such as a film, is encrypted using k_A (where A is the group of users permitted to access the service) and then B_A is broadcast as a header that allows an authorised user U_i in A to determine k_A and hence decrypt the service. There are two slightly different applications of broadcast encryption, which we illustrate using the above multimedia service scenario:

1. **General broadcast encryption**: These schemes are usually designed for as large a communication structure as possible, since this maximises the possible number of different groups for whom group keys can be generated. These are suitable for *pay-per-view* services, where the groups of users receiving content are highly variable (for example only a small group from the set of all users may want to pay to watch a particular football match).

2. **Long term group management**: These schemes are characterised by a single large group of users that may change gradually over time. These are suitable for *subscription* services, where we only ever want to broadcast to the entire group of subscribed users, but the make-up of this group is dynamic.

Note that these two applications are far from being mutually exclusive. The main difference is that while schemes designed for the first scenario should be able to efficiently broadcast to user groups of all sizes, schemes designed for the second scenario initially associate a group key k_H with a single group of users H (from the universe $\mathcal{U}$ of possible users) and should be specifically designed to efficiently cope with relatively small changes to H over time. This scenario is often described in terms

of maintaining a *multicast group*, where the term "multicast" arises from internet-related technology for sending a single message to a designated set of recipients [22].

One significant difference between proposed broadcast encryption schemes relates to the computational capabilities of users in the scheme. We say that a broadcast encryption scheme is suitable for:

- *stateless receivers* if the users cannot retain information from previous broadcasts (or have ability to write to memory). This might be the case for example if the user is a set-top decoder. The decoder is preloaded with decryption keys that cannot be changed over time. Each time a broadcast message is sent, the decoder can use these keys to decrypt the broadcast, but it will not retain any memory of the information it receives (if the same group key is used twice, the decoder will have to decrypt it on each occasion, as it cannot store any information supplied to it during a key establishment event).

- *stateful receivers* if the users can retain information from previous broadcasts (or have the ability to write to memory). The critical difference in this case is that if new keys are broadcast to them then users can use these to replace the keys that were distributed to them on initialisation (in other words users have the ability to update their secret data).

The motivation for considering a stateless receiver model is that this greatly simplifies the software or hardware needed by the users. Almost all the schemes that we discuss in this paper are suitable for stateless receivers.(Whether real human users are stateless or stateful will be left as an open problem!)

6.1.1 Benchmark broadcast encryption schemes We now define two benchmark broadcast encryption schemes against which others need to be compared. Both are suitable for stateless receivers. These are analogues of Scheme 5.3 and Scheme 5.4 respectively. Throughout the remainder of this section we assume that E is a secure symmetric encryption algorithm with key size l and $E_k(m)$ denotes the encryption of plaintext m using key k.

Scheme 6.3 *A* trivial broadcast encryption scheme *(TBES) has the following properties:*

- $u_i = \{K_A \,|\, U_i \in A, A \in \mathcal{C}\}$*, where each* K_A *is randomly chosen from* $\{0,1\}^l$*;*
- $B_A = E_{K_A}(k_A)$*;*
- $K_A \in u_i$ *if and only if* $U_i \in A$*, with* k_A *obtained by decrypting* $E_{K_A}(k_A)$*.*

Scheme 6.4 *A* direct broadcast encryption scheme *(DBES) has the following properties:*

- $u_i = k_i$*, where each* k_i *is randomly chosen from* $\{0,1\}^l$*;*
- $B_A = \{E_{k_i}(k_A) \,|\, U_i \in A\}$*;*
- $E_{k_i}(k_A) \in B_A$ *if and only if* $U_i \in A$*, with* k_A *obtained by decrypting* $E_{k_i}(k_A)$*.*

We thus see that Schemes 6.3 and 6.4 offer extreme ends of the tradeoff between the size of user secret data u_i and the size of the broadcast header B_A. Both schemes are deterministic, computationally secure and have independent keys. In general the hunt for good broadcast encryption schemes is about finding schemes with reasonable parameter tradeoffs between these two benchmark schemes.

6.1.2 Broadcast encryption schemes from key predistribution schemes One simple way of establishing a broadcast encryption scheme suitable for stateless receivers is to build it onto an existing key predistribution scheme.

Scheme 6.5 *If we have a $(\mathcal{C}, \mathcal{X})$-KPS then we can realise a $(\mathcal{C}, \mathcal{X})$-BES as follows:*

- *u_i is the same for both the KPS and the BES;*
- *$B_A = E_{k_A^*}(k_A)$, where k_A^* is the group key for $A \in \mathcal{C}$ in the KPS and k_A is a freshly generated group key for A in the BES;*
- *Only a user U_i in A can establish k_A^* from u_i and hence decrypt the new group key k_A.*

While Scheme 6.5 is attractively simplistic, the main problem with it is that for broadcast encryption we generally want $\mathcal{C}$ to be large, and a KPS for a large $\mathcal{C}$ typically has high user storage requirements.

In [15] a broadcast encryption scheme was suggested that employs a KPS that establishes keys for groups of l users in order to construct a BES that establishes keys for groups of $t = \lambda l$ users. This scheme uses a resolvable design defined on t points to partition the t users into blocks of size l, which are then used to define a broadcast message. The advantage of this idea is that the user storage required for the KPS on group size l is smaller than that for group size t. This scheme is not particularly efficient with respect to broadcast size if we are planning to use a BES to distribute a group key k_A (which in this paper we are), however it has some merits if the information to be broadcast to the group is longer.

In [72] another interesting family of broadcast encryption schemes were proposed, which combine a KPS with an ideal secret sharing scheme (see Section 4.5). The idea is the following:

Scheme 6.6 *Let $\mathcal{U}$ be a set of users and $\mathcal{X}$ be an exclusion structure defined on $\mathcal{U}$. Suppose that we can find:*

1. *A set system $(\mathcal{U}, \mathcal{B})$, where $|\mathcal{B}| = b$ and for each block $B_j \in \mathcal{B}$ we can construct a $(2^{B_j}, \mathcal{X}_j)$-KPS on user set B_j, where:*
   (a) *The user secret for each $U_i \in B_j$ is denoted by u_i^j.*
   (b) *The group key for each $A \subseteq B_j$ is denoted by k_A^j and is an element of $\mathcal{K}$.*
2. *An ideal secret sharing scheme (with shares and secrets from $\mathcal{K}$) on participant set $\mathcal{B}$ with access structure Γ such that:*
   (a) *For every $U_i \in \mathcal{U}$, we have $\{B_j \,|\, U_i \in B_j\} \in \Gamma$;*
   (b) *For every $X \in \mathcal{X}$, we have $\{B_j \,|\, X \cap B_j \notin \mathcal{X}_j\} \notin \Gamma$.*

Then a KIO $(2^{\mathcal{U}}, \mathcal{X})$*-broadcast encryption scheme is defined by:*

- *For each* U_i *let* $u_i = \{u_i^j \mid U_i \in B_j\}$.
- $B_A = (E_{k_1}(y_1), \ldots, E_{k_b}(y_b))$*, where:*
  1. $(y_1, \ldots, y_b)$ *are shares of the ideal secret sharing scheme corresponding to secret* k_A*;*
  2. $k_j = k^j_{A \cap B_j}$ *for each* $1 \leq j \leq b$*;*
  3. E *is a symmetric encryption algorithm with keys from set* $\mathcal{K}$*.*

Although at first glance complex, the intuition behind Scheme 6.6 is straightforward. If U_i is a member of set A then they can use their secret information u_i^j to determine the group keys $k^j_{A \cap B_j}$ for each KPS that U_i is a member of. These then allow U_i to decrypt a set of shares y_j that correspond to a set in the access structure of the secret sharing scheme, which means that the shares can be used to reconstruct k_A. On the other hand, a set of users X in the exclusion structure can, in the worst case, determine a set of group keys that decrypt a set of shares not in the access structure, hence they obtain no information about the group key k_A.

Thus we need to find combinations of set systems, KPSs and ideal secret sharing schemes that allow Scheme 6.6 to be enabled. The KIO broadcast encryption scheme construction was first proposed in [72] using the Trivial Exclusion KDPs (Scheme 5.15) from [35] with exclusion parameter w as the KPSs. In particular it has been shown that the following combinations result in KIO broadcast encryption schemes:

1. Let $(\mathcal{U}, \mathcal{B})$ be a $2-(n, b, r, k, \lambda)$ design with $r > \lambda\binom{w}{2}$ and choose an ideal $(\lambda\binom{w}{2} + 1, b)$-threshold scheme [72].
2. An improved scheme is obtained by letting $(\mathcal{U}, \mathcal{B})$ be an (n, b, r, λ)-*broadcast key distribution pattern* (BKDP) [73], which is a set system of n points, b blocks, every point on r blocks, every pair of points in at most λ blocks and $r > \lambda\binom{w}{2}$. These structures were first defined in [74] using the name *threshold designs* and several constructions based on Steiner systems and orthogonal arrays were provided.
3. A further improvement was made in [75], where it was observed that the KIO construction technique can be further generalised to allow the ideal secret sharing scheme to be replaced by a *ramp scheme* (see [38]), which is a type of secret sharing scheme that permits smaller share sizes.

Example 6.7 In order to see the kinds of parameter tradeoff that are possible using KIO, we note that in [74] it was shown that an orthogonal array $\mathrm{OA}_1(t, q, q)$ can be used to construct a $(q^t, q^2, q, t-1)$-BKDP. This gives rise to a BES for q^t users, where each user stores at most $q + (t-1)(q^t - 1)$ values and the broadcast B_A to enable any group key k_A is of length q^2. This compares with user storage of $2^{q^t - 1}$ and broadcast length 1 for the TBES, and user storage of 1 and broadcast length $|A|$ for the DBES. Given that this construction works for any $t < q$, it is clear that the KIO construction provides a balance between the extremes of TBES and DBES,

especially when t is close to q and A is very large (as is likely to be the case in many of the applications envisaged for broadcast encryption).

6.1.3 Logical key hierarchies for stateful receivers Using a tree of keys to manage a long term group key was first suggested in [77] and [79]. The broadcast encryption scheme that they independently proposed is suitable for stateful receivers. The basic idea is as follows.

Scheme 6.8 *Let H be a subset of m users (chosen from a universe $\mathcal{U}$) who wish to establish a group key k_H. For simplicity, assume that $m = 2^h$. To establish a* logical key hierarchy*:*

1. *Define a (complete) binary tree with m leaves, each associated with a user from H. Iteratively label this tree with independent keys as follows: root by $k_{0,0}$; the left child of $k_{i,j}$ by $k_{i+1,2j}$; and right child of $k_{i,j}$ by $k_{i+1,2j+1}$. Associate the users, which we label $U_0, \ldots, U_{m-1}$, with the nodes labeled by keys $k_{h,0}, \ldots, k_{h,m-1}$.*
2. *For each user U_j let $u_j = \{k_{x,y} \mid k_{x,y}$ is on the path from $k_{h,j}$ to $k_{0,0}\}$. Each user thus holds $h+1$ keys.*
3. *The key $k_{0,0}$ is held by every user in H. In order to establish a group key k_H the TA could broadcast $B_H = E_{k_{0,0}}(k_H)$. Note however that since this scheme is intended for stateful receivers (and thus users have the ability to refresh their keys) it is also possible just to let $k_H = k_{0,0}$, in which case this scheme can actually be considered as a type of key predistribution scheme.*

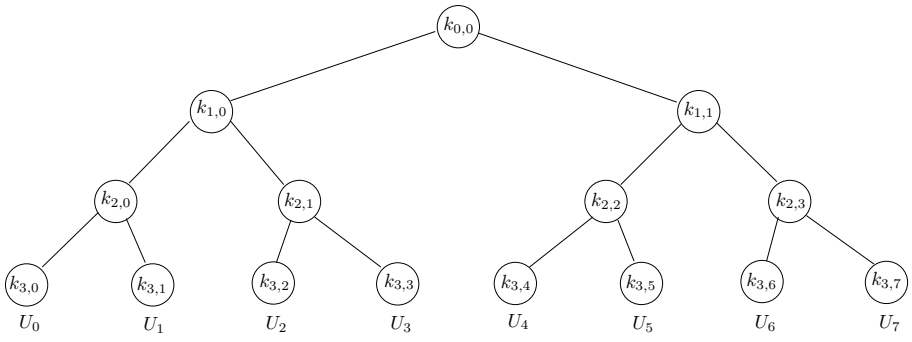

Figure 4: Logical key hierarchy tree for eight users

Example 6.9 Figure 4 shows the binary tree for a logical key hierarchy for eight users H. Each user stores four keys. Thus, for example, user U_3 stores $u_3 = \{k_{3,3}, k_{2,1}, k_{1,0}, k_{0,0}\}$. The group key can be decrypted using (or is) $k_{0,0}$, which is stored by every user, and the remaining keys that a user stores all facilitate group changes. We illustrate this process by an example. Suppose that U_5 leaves the group. It is necessary to replace all the keys held by U_5 that are also held by any other user. The most efficient process for doing this is as follows:

1. The TA generates new keys $k'_{2,2}, k'_{1,1}, k'_{0,0}$;
2. The TA encrypts $k'_{2,2}$ using key $k_{3,4}$;
3. The TA encrypts $k'_{1,1}$ using keys $k_{3,4}$ and $k_{2,3}$;
4. The TA encrypts $k'_{0,0}$ using keys $k_{3,4}$, $k_{2,3}$ and $k_{1,0}$;
5. The TA broadcasts all these encrypted keys;
6. User U_4 decrypts $k'_{2,2}$ and replaces $k_{2,2}$ with this new key (user U_4 can do this because it is a stateful receiver);
7. Similarly, users U_4, U_6 and U_7 replace $k_{1,1}$ by $k'_{1,1}$;
8. Similarly, all users except U_5 replace $k_{0,0}$ by $k'_{0,0}$.

The scheme has now been updated in such a way that the new group key is determined using $k'_{0,0}$, which is a key that the departing user U_5 does not know.

It is straightforward to generalise Scheme 6.8 to use a-ary trees rather than binary trees. Example 6.9 should be sufficient to illustrate how general protocols for leaving or joining groups of users can be derived.

6.1.4 Schemes based on covers for stateless receivers We now look at a family of broadcast encryption schemes designed for stateless receivers.

Definition 6.10 Let $(\mathcal{I}, \mathcal{B})$ be a set system and for each $x \in \mathcal{I}$ let $\beta(x) = \{B \in \mathcal{B} \,|\, x \in B\}$. We say that $(\mathcal{I}, \mathcal{B})$ is a *cover-based revocation system* (CBRS) if for every non-empty $\mathcal{A} \subseteq \mathcal{B}$ there exists $\mathcal{I}_{\mathcal{A}} \subseteq \mathcal{I}$ such that

$$\bigcup_{x \in \mathcal{I}_{\mathcal{A}}} \beta(x) = \mathcal{A}.$$

In other words, a set system is a CBRS if for every non-empty collection $\mathcal{A}$ of blocks there exists a subset $\mathcal{H}$ of points such that the subsets $\{\beta(x) \,|\, x \in \mathcal{H}\}$ form a cover of $\mathcal{A}$.

Scheme 6.11 *Given a cover-based revocation system $(\mathcal{I}, \mathcal{B})$ we can define a broadcast encryption scheme for stateless receivers as follows:*

- *Associate each point $x \in \mathcal{I}$ with a key k_x, and associate each block $B_i \in \mathcal{B}$ with a user U_i;*
- $u_i = \{k_x \,|\, x \in B_i\}$*;*
- *For any subset A of users (corresponding to the set of blocks $\mathcal{A}$), $B_A = \{E_{k_x}(k_A) \,|\, x \in \mathcal{I}_{\mathcal{A}}\}$;*
- *By definition of a CBRS, the only users holding at least one of the keys k_x are those in A.*

Broadcast encryption schemes arising from Scheme 6.11 allow group keys to be established for any subset of users (hence $\mathcal{C} = 2^{\mathcal{U}}$), while not permitting any unauthorised subset access to the group key (full collusion security). It is worth noting however that such schemes are only practical if there also exists an efficient algorithm for determining the appropriate cover of keys, given a particular subset of users.

We now describe a manifestation of Scheme 6.11 from [58], based on the binary tree exhibited in Scheme 6.8.

Scheme 6.12 *For simplicity, assume that* $|\mathcal{U}| = m = 2^h$. *To establish a* complete subtree revocation scheme*:*

1. *Define a (complete) binary tree with* m *leaves as in Scheme 6.8.*
2. *As in Scheme 6.8, let* $u_j = \{k_{x,y} \mid k_{x,y}$ *is on the path from* $k_{h,j}$ *to* $k_{0,0}\}$. *Each user thus holds* $h + 1$ *keys.*
3. *In order to establish a group key* k_A, *where* $A = \mathcal{U} \setminus R$*:*
   - *Form the subtree ST(R) consisting of the paths from the nodes in* R *to the root (this is sometimes called the* Steiner Tree *of nodes* R*).*
   - *Identify the set* $\mathcal{K}(A)$ *of nodes* $k_{x,y}$ *of the main tree such that* $k_{x,y}$ *is not a node of ST(R) but the parent of* $k_{x,y}$ *is a node of ST(R) (such nodes are sometimes referred to as* hanging off *ST(R), and form a cover of* A*).*
   - *Let* $B_A = \{E_{k_{x,y}}(k_A) \mid k_{x,y} \in \mathcal{K}(A)\}$.

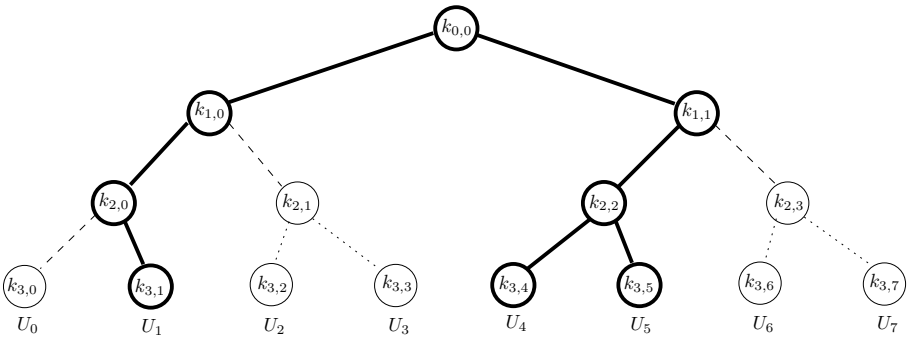

Figure 5: Complete subtree revocation of users U_1, U_4 and U_5

Example 6.13 Figure 5 shows the binary tree for a complete subtree revocation scheme for eight users in which users $R = \{U_1, U_4, U_5\}$ are being revoked (in other words, a group key is being established for $A = \{U_0, U_2, U_3, U_6, U_7\}$). The edges in bold form $\mathrm{ST}(R)$ and the keys connected to $\mathrm{ST}(R)$ by dashed edges form $\mathcal{K}(A)$. In this case $B_A = \{E_{k_{3,0}}(k_A), E_{k_{2,1}}(k_A), E_{k_{2,3}}(k_A)\}$.

Scheme 6.12 was generalised in [2] to a-ary trees and some interesting compression techniques were proposed for further reducing the user storage.

In [58] it was shown that the following alternative CBRS, which we describe informally, can be extracted from a different labeling of a complete binary tree.

Scheme 6.14 *For simplicity, assume that* $|\mathcal{U}| = m = 2^h$. *To establish a* subset difference revocation scheme*:*

1. *Define a (complete) binary tree with* m *leaves whose nodes are labeled from the root downwards by* $v_1, \ldots, v_{2^{h+1}-1}$. *Associate the users, which we label* $U_0, \ldots, U_{m-1}$, *with the leaf nodes (labeled by keys* $v_{2^h}, \ldots, v_{2^{h+1}-1}$*).*
2. *Associate a key* $k_{x,y}$ *with any pair of nodes* v_x *and* v_y *of the tree such that* v_y *is a descendent of* v_x.
3. *For each user* U_j *let* $u_j = \{k_{x,y} \,|\, U_j$ *is a descendent of* v_x *but not of* $v_y\}$.
4. *In order to establish a group key* k_A, *where* $A = \mathcal{U} \setminus R$, *run a simple algorithm (defined in [58]) to find a subset of keys that cover* A.

Without going into further details it should be evident that Scheme 6.14 results in a broadcast encryption scheme with much greater user storage than Scheme 6.12. However in general Scheme 6.14 has a much smaller broadcast message (resulting from a smaller cover) than Scheme 6.12 and so represents an alternative tradeoff.

Several variants of these schemes have been proposed in the literature. For example: in [36] a modification of the subset difference scheme based on defining layers of the underlying tree was proposed; in [3] a compression technique for reducing the user storage of the subset difference scheme was identified; in [55] it was shown that the complete subtree and subset difference schemes can be combined to obtain further examples of attractive tradeoffs.

Lastly we note that in [43] it was observed that a $(1, w, d)$-CFF (see Section 5.3) provides a restricted notion of a CBRS, where there exists a suitable cover for any subset of at least $|\mathcal{B}| - w$ blocks (and hence up to w users can be revoked).

6.1.5 Broadcast encryption with extended capabilities The attractive applications of broadcast encryption have resulted in some interesting extensions to the basic concept being proposed and investigated. We briefly identify two areas where interesting research has been conducted:

- **Traceable broadcast encryption**: The problem of piracy of decoder boxes for commercial information services has led to an interest in incorporating *traceability* into the keys allocated to a user in a broadcast encryption scheme. This means that any group of users who combine their keys to forge a new decoder can have at least one of their identities revealed if that decoder is later captured and analysed. This idea was first proposed in [25] and has subsequently been extensively investigated. Creating suitable distributions of keys presents a number of interesting combinatorial problems, which were comprehensively reviewed in [7].
- **Self-healing broadcast encryption**: If the broadcast channels being used are unreliable then it is possible that some users may not reliably receive the broadcast information that allows them to determine a given group key. The idea behind a *self-healing* broadcast encryption scheme is that additional information is broadcast on each occasion that allows valid group members

to recover any missing group key from a combination of previous and subsequent broadcast messages. This idea is most appropriate for applications where regular group keys are established over a series of discrete time intervals. Self-healing broadcast encryption was first proposed in [70] and subsequently investigated in, for example, [13] and [14].

6.2 Decentralised schemes

One of the concerns of relying on a TA to play a major role during the key establishment phase is that it becomes a potential central point of failure. This even applies to the (trivial) key distribution scheme when the TA maintains secure channels with users throughout the scheme lifetime. One method of mediating against this, which was first studied in [59], is to have a set $TA_1, \ldots, TA_r$ of r different TAs, of which at least m must be involved in the establishment of any group key. This idea was first suggested for decentralising key predistribution schemes in [44]. We capture this concept informally in the following definition.

Definition 6.15 An $(m, r, \mathcal{C}, \mathcal{X})$-*distributed key distribution scheme* (DKDS) is a $(\mathcal{C}, \mathcal{X})$-KDS with the stronger properties that:

1. Given $A \in \mathcal{C}$, any $U_i \in A$ can compute the group key k_A from knowledge of u_i and $\{v_{i,A,j_1}, \ldots, v_{i,A,j_m}\}$, where v_{i,A,j_l} is some information obtained by U_i from TA_l during the key establishment phase for key k_A.

2. Given $A \in \mathcal{C}$, $B \in \mathcal{X}$ and a set of $m-1$ TAs, it is not possible to determine k_A from u_A, the private information held by the $m-1$ TAs and any information sent to any user in a previous key establishment event.

Defintion 6.15 was formalised in [11] in an information theoretically secure model and several bounds on scheme parameters, including TA storage, were established. These essentially show that a scheme suggested in [59] is optimal. We briefly outline this optimal DKDS.

Scheme 6.16 *Given $(\mathcal{C}, \mathcal{X})$, we let $\lambda = \max_{B \in \mathcal{X}} |\{A \in \mathcal{C} \mid A \cap B \neq \emptyset\}$ (in other words, the maximum number of group keys that any set $B \in \mathcal{X}$ can compute) and associate each $A \in \mathcal{C}$ with an element $h_A \in GF(q)$. Initialise the $(m, r, \mathcal{C}, \mathcal{X})$-DKDS as follows:*

- *Each user U_i is issued with a set of keys that allow them to communicate securely with each of the r TAs.*
- *TA_i $(1 \leq i \leq m)$ constructs a random bivariate polynomial $f_i(x, y)$ of degree $m-1$ in x, degree $\lambda - 1$ in y, and with coefficients from $GF(q)$.*
- *TA_i securely sends the univariate polynomial $f_i(j, y)$ to TA_j $(1 \leq j \leq r)$.*
- *TA_j computes and stores the univariate polynomial $t_j(y) = \sum_{i=1}^{m} f_i(j, y)$ as their private information.*

When user $U_i \in A$ wants to establish k_A:

- *U_i sends a request to a set of m TAs, say $TA_{i_1}, \ldots, TA_{i_m}$.*
- *TA_{i_j} sends $v_{i,A,i_j} = t_{i_j}(h_A)$ to U_i.*
- *U_i uses Lagrange interpolation to recover $k_A = \sum_{l=1}^{m} f_l(0, h_A)$ from the values $\{t_{i_1}(h_A), \ldots, t_{i_m}(h_A)\}$.*

Various generalisations and extensions to the basic idea of a DKDS have been studied in the literature. For example in [12] a tradeoff between storage and security was exhibited by employing ramp schemes and in [29] a DKDS was proposed which is more robust against users and TAs who do not follow the specified protocols correctly.

7 Group key agreement schemes

The third class of key establishment schemes that we look at are those where users can communicate with one another during the key establishment phase. We assume that, as for the reasons given at the start of Section 5, there is no trusted authority available to assist with key establishment after the initialisation phase. The majority of group key agreement schemes are particularly suited to environments where the nature of the communication structure is not known in advance. A group key agreement scheme then allows an ad hoc group to create a group key amongst themselves. For this reason (motivated by potential applications to secure teleconferencing) they are sometimes referred to as *conference key* schemes. In [15] they are referred to as *interactive key distribution schemes*.

Our classification of key agreement schemes as any scheme that involves user interaction unassisted by a trusted authority is highly generic and allows us to group together several very different types of group key establishment schemes. The vast majority of group key agreement schemes are based on public key cryptographic techniques and are mostly beyond the scope of this paper as they do not inherently involve combinatorial techniques. Many of these, including one that we will look at in Section 7.2, are based around the classical *Diffie-Hellman* protocol [30], which we briefly describe for the simple two-party case.

Scheme 7.1 *Let G be a finite multiplicative group of some large prime order q and let g be a generator of G (these parameters are published during the initialisation phase). If U_1 and U_2 wish to establish a key k then:*

1. *U_1 randomly chooses $x_1 \in Z_q^*$ and sends g^{x_1} to U_2;*
2. *U_2 randomly chooses $x_2 \in Z_q^*$ and sends g^{x_2} to U_1;*
3. *U_1 computes $k = (g^{x_2})^{x_1}$ and U_2 computes $k = (g^{x_1})^{x_2}$, both of which are equal to $g^{x_1 x_2}$.*

The security of the Diffie-Hellman protocol relies on the difficulty of taking discrete logarithms (see any standard cryptographic text such as [73]). We will not make any attempt in this paper to review the vast range of extensions and alternatives to Diffie-Hellman that have been proposed for group key agreement, and refer to surveys such as [18] and (more recently and exclusive on key agreement) [31].

7.1 Group key agreement from KPSs and KDSs

In a similar way to our discussion in Section 6.1.2, it is conceptually possible to construct a group key agreement scheme for either a group key predistribution scheme or a group key distribution scheme.

Scheme 7.2 *If we have a $(\mathcal{C}, \mathcal{X})$-KPS then we can realise a $(\mathcal{C}, \mathcal{X})$ key agreement scheme as follows:*

- *Each user stores u_i, as issued in the KPS;*
- *Users $U_i \in A$ establish group key k_A by utilising secure channels amongst themselves that are protected by the group key k_A^* associated with the KPS.*

Precisely how Scheme 7.2 can be manifested very much depends on the mutual trust between users in the scheme. One simple option is that one user U_i could generate k_A and then distribute it to the others encrypted by k_A^*. Another option is that each user $U_i \in A$ generates a component k_A^i, which is then distributed encrypted by k_A^*. Each user in A then decrypts the component and forms $k_A = \sum_{U_i \in A} k_A^i$. Regardless of how this is done, the resulting key agreement scheme will suffer from limitations similar to those of Scheme 6.5 that were noted in Section 6.1.2.

Scheme 7.3 *If we have a $(\mathcal{C}, \mathcal{X})$-KDS then we can realise a $(\mathcal{C}, \mathcal{X})$ key agreement scheme as follows:*

- *During the initialisation phase, each user is provided with data that allows them to fulfill the role of TA in a $(\mathcal{C}, \mathcal{X})$-KDS;*
- *Users $U_i \in A$ establish group key k_A by utilising the group keys k_A^{i*} associated with each of the KDSs.*

Again there are many ways in which Scheme 7.3 could actually manifest itself. Note also that there is no need for the individual KDSs strictly to be $(\mathcal{C}, \mathcal{X})$-KDSs, since it is possible that schemes with smaller communication structures could be cleverly combined. This is precisely what was done in [6], which was later generalised in [15]. This scheme used the broadcast encryption scheme based on a resolvable design discussed in Section 6.1.2 to establish a group key agreement scheme, where each user in A acted as a TA and broadcast an encrypted component key k_A^i, which was then combined to form the group key k_A.

7.2 Key agreement schemes for long term group management

Analogously to the situation discussed in Section 6.1, in dynamic application environments where group membership regularly changes it may be desirable to have schemes that allow long term group keys to be established by key agreement techniques. We have already seen in Section 6.1 that trees underpin a number of group key distribution schemes. There have been several proposals for tree-based group key agreement schemes based on Diffie-Hellman. We will describe the basic set up of just one such scheme, from [42].

Scheme 7.4 *Let H be a subset of m users (chosen from a universe $\mathcal{U}$), G be a finite multiplicative group of some large prime order q and let g be a generator of G. During the initialisation phase these parameters are published and users are issued with information that allows them to communicate with one another using authenticated public broadcast channels. For simplicity, assume that $m = 2^h$. The key establishment phase proceeds as follows:*

1. *Define a (complete) binary tree with m leaves, each associated with a user from H. Label this tree iteratively as in Scheme 6.8 as follows: root by $N_{0,0}$; the left child of $N_{i,j}$ by $N_{i+1,2j}$; and right child of $N_{i,j}$ by $N_{i+1,2j+1}$. Hence the users, which we label $U_{h,0}, \ldots, U_{h,m-1}$, correspond to nodes $N_{h,0}, \ldots, N_{h,m-1}$.*

2. *Each user $U_{h,j}$ generates a secret value $k_{h,j}$ and publicly broadcasts $g^{k_{h,j}}$ to the other members of H.*

3. *Each pair of users $U_{h,j}$ and $U_{h,j+1}$ (for all even $0 \leq j \leq m-1$) compute the Diffie-Hellman key $k_{h-1,j/2}$ using Scheme 7.1, which is then associated with node $N_{h-1,j/2}$. Both $U_{h,j}$ and $U_{h,j+1}$ securely store $k_{h-1,j/2}$ and publicly broadcast $g^{k_{h-1,j/2}}$.*

4. *Each quartet of users $U_{h,j}, U_{h,j+1}, U_{h,j+2}, U_{h,j+3}$ (for all $j \equiv 0 \pmod 4$, $0 \leq j \leq m-1$) compute the Diffie-Hellman key $k_{h-2,j/4}$ using Scheme 7.1, which is then associated with node $N_{h-2,j/4}$. All four users securely store $k_{h-2,j/4}$ and publicly broadcast $g^{k_{h-2,j/4}}$.*

5. *This process is iterated until the last Diffie-Hellman calculation results in $k_{0,0}$, which is adopted as the group key k_H.*

At the end of the above protocol, each user $U_{h,j}$ holds each key that is associated with the nodes on the path from $N_{h,j}$ to the root $N_{0,0}$, as well as $g^{k_{i,j}}$ for each node $N_{i,j}$ in the tree.

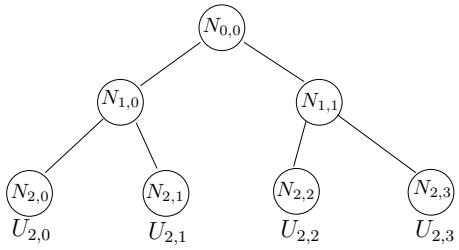

Figure 6: Tree-based Diffie-Hellman node allocation on four nodes

Example 7.5 The underlying tree for $m = 4$ is shown in Figure 6. In this example, $U_{2,j}$ first generates $k_{2,j}$ and broadcasts $g^{k_{2,j}}$ $(1 \leq j \leq 4)$. Users $U_{2,0}$ and $U_{2,1}$ conduct a Diffie-Hellman exchange to compute $k_{1,0} = g^{k_{2,0}k_{2,1}}$, and $U_{2,2}$ and $U_{2,3}$ conduct a Diffie-Hellman exchange to compute $k_{1,1} = g^{k_{2,2}k_{2,3}}$. Both $g^{k_{1,0}}$ and $g^{k_{1,1}}$ are then broadcast. Finally the group key $k_{0,0} = g^{k_{1,0}k_{1,1}}$ can be computed by all

four users. Each user stores all keys on the path from their leaf node to the root. So, for example, $U_{2,0}$ stores $k_{2,0}$, $k_{1,0}$ and $k_{0,0}$.

As mentioned in Section 6.1, the main advantage of setting up the key agreement tree in Scheme 7.4 is that this structure facilitates relatively efficient user join, user leave, group merge and group partition operations in dynamic environments. We leave the details of these protocols to [42].

7.3 Wireless sensor network schemes

A very interesting class of key establishment schemes that are of combinatorial interest are those designed for application in *wireless sensor networks*. These networks consist of tiny, inexpensive, low-powered *sensors* fitted with wireless transmitters, which can be spatially scattered to form an *ad hoc network*. They are particularly suited to applications in environments where it is difficult to manually establish a communication network, such as during disaster relief operations, seismic data collection, wildlife monitoring or military intelligence gathering. Sensors are distributed around the application environment (perhaps by aeroplane drop) and then attempt to set up a network in order to exchange and return data. What makes wireless sensor networks particularly intriguing is that the actual network topology (defined by a *physical graph*, whose edges represent sensors that are able to communicate with one another at a particular instant in time) is not known prior to deployment and is potentially highly dynamic. Thus we might as well model the physical graph as a random graph.

There are three factors that influence the choice of key establishment techniques for wireless sensor networks. The first is the fact that as there is no network controller after initialisation, there is no entity that can play the role of a TA. This lends itself to either key predistribution or key agreement schemes. However sensors also have very limited storage and computational abilities. This presents a dilemma since:

- As we have already seen in Section 5, the cost of relying solely on key predistribution is often substantial secret storage;
- Relying solely on key agreement involves considerable computational costs and is likely to be hampered by the random nature of the physical graph.

A sensible compromise is thus to predistribute keys "as well as possible", while also permitting a limited amount of communication between sensors to take place during key establishment, effectively making such schemes group key agreement schemes.

7.3.1 A key establishment model for wireless sensor networks

The basic idea behind a *wireless sensor network scheme* (WSN scheme) for communication structure $\mathcal{C}$ is to first establish a $(\mathcal{C}^*, \mathcal{X})$-key predistribution scheme, where both $\mathcal{C}$ and $\mathcal{C}^*$ are defined on the same set of sensors (users). We refer to $\mathcal{C}^*$ as the *network communication structure*. When a set $A \in \mathcal{C}$ of sensors requires a group key k_A:

1. if $A \in \mathcal{C}^*$ then they establish k_A using the KPS;
2. if $A \notin \mathcal{C}^*$ then they use some key agreement protocol to establish a key k_A, potentially using other sensors in the network to assist them.

Note that it is quite reasonable to rely on other sensors to assist in a key agreement process since the random nature of the physical graph necessitates that nodes typically rely on one another for message transmission services. One commonly proposed key agreement technique is to seek a path of secure links in the physical graph that joins the sensors in A, and then get one of the sensors on this path to generate a key and securely relay it.

It is clearly desirable to have $\mathcal{C}^*$ matching $\mathcal{C}$ as closely as possible. As a result, the effectiveness of a WSN scheme is often measured in terms of the *local connectivity*, which is a notion of the probability that a group of sensors in $\mathcal{C}$ either are in $\mathcal{C}^*$ or are "close" to being in $\mathcal{C}^*$ (where "close" is normally measured in terms of the number of other sensors that need to be involved in establishing k_A if $A \notin \mathcal{C}^*$).

There are a wide variety of different approaches to the design of WSN schemes and we refer to [21] for a comprehensive survey. We now review a number of interesting applications of combinatorial structures to the design of WSN schemes. We will mainly restrict our interest here to the case where $\mathcal{C} = \{A \subseteq \mathcal{U} \,|\, |A| = t\}$ (and in particular the case $t = 2$), in which case we will refer to *t-wise* (*pairwise*) WSN schemes.

7.3.2 Using a KPS Any key predistribution schemes that we have already discussed in Section 5 could potentially be adopted as part of a WSN scheme. Thus before considering dedicated designs, it is worth considering existing candidate KPSs. The most appropriate schemes are those with relatively low user storage and fast key computation. This makes schemes such as tree-based key distribution patterns (Section 5.4) attractive candidates. Also of interest are probabilistic schemes such as Scheme 5.8 (the random KPS) and Scheme 5.10, where efficiencies have been gained at the expense of a slightly unpredictable network communication structure. However these KPSs have not all been proposed explicitly for WSNs and, in particular, it is desirable to try to custom-design a KPS that also results in an efficient key agreement phase.

7.3.3 Key ring WSN schemes We now discuss a class of WSN schemes that are based on key ring predistribution schemes (fundamental Scheme 5.7). Let t be a positive integer and let $U_1, \ldots, U_n$ be a collection of sensors. Let $\mathcal{R} = (\mathcal{I}, \mathcal{B})$ be a key ring, as defined in Section 5.1.2.

Definition 7.6 A $(t, n, \mathcal{R})$*-key ring WSN scheme* (KRWSN) is a WSN scheme arising from a key ring predistribution scheme based on $\mathcal{R}$, where the (network) communication structure is

$$\mathcal{C}^* = \{A \subseteq \mathcal{U} \,|\, |A| = t \text{ and } \bigcap_{U_i \in A} u_i \neq \emptyset\}.$$

In other words, a group $U_1, \ldots, U_t$ of t sensors check their public identifier sets $Pub_1, \ldots, Pub_t$ to see if they share any common identifiers. If they do, then they can establish a group key k_A by applying g to the keys k_i that correspond to the identifiers in $\cap_{j=1}^{t} Pub_j$. If not, then they establish the group key by an alternative key agreement mechanism.

The first KRWSN schemes proposed were based on a version of Scheme 5.8, the random key predistribution scheme, where the key ring was defined by $\mathcal{B} = \mathcal{I}^k$ (the collection of all subsets of $\mathcal{I}$ of some fixed size k) [24, 33]. Not surprisingly however, improved schemes can be obtained if the key ring has a more combinatorial structure. In [20], both projective planes and generalized quadrangles were suggested as candidate key rings.

Scheme 7.7 *Let q be a prime power and $\mathcal{R} = (\mathcal{I}, \mathcal{B})$ be a projective plane of order q. For any $n \leq q^2 + q + 1$ we obtain a $(2, n, \mathcal{R})$-KRWSN scheme such that:*

- *Each sensor needs to store $q + 1$ keys;*
- *Every pair of sensors shares precisely one common key.*

Scheme 7.7 is of course also an example of a KDP (see Section 5.3) and thus could have been included in Section 7.3.2. The fact that every pair of sensors share a key means that in this example no key agreement stage is necessary. There is a subtle problem with this scheme however. Many of the applications for wireless sensor networks involve a large number of sensors (potentially tens of thousands). This necessitates a suitable large choice of q in Scheme 7.7, which in turn requires the storage-limited sensor to hold too many keys. In [47] both transversal designs and quadratic curves were considered as candidate key rings and shown to have better properties.

Scheme 7.8 *Let q be a prime and $\mathcal{R} = (\mathcal{I}, \mathcal{B})$ be a transversal design $TD(k, q)$. For any $n \leq q^2$ we obtain a $(2, n, \mathcal{R})$-KRWSN scheme such that:*

- *Each sensor needs to store k keys;*
- *Every pair of sensors share precisely zero or one common key;*
- *The probability that a pair of sensors share a common key is $k/(q + 1)$.*

To see that Scheme 7.8 is an improvement on Scheme 7.7, consider the following example:

Example 7.9 Suppose that we require a WSN with 2400 sensors. If Scheme 7.7 is used then we need to choose $q = 49$ and each node is required to store 50 keys. On the other hand, we could apply Scheme 7.8 with a TD$(30, 49)$ [47], in which case each node only needs to store 30 keys. In this case any pair of nodes share a key with probability 0.6. It is further shown in [47] that if we make certain reasonable assumptions about the valency of the physical graph, the probability that any pair of sensors either share a key, or are connected in the physical graph to a third sensor with whom they both share a common key, is very close to 1.

Example 7.9 motivates the hunt for a more general class of combinatorial structures with similar properties. We say that two sensors U_i and U_j in a WSN can communicate via a *two-hop path* if there exists a third sensor U_k such that both U_i and U_k, and U_j and U_k, share common keys. For a KRWSN scheme this condition equates to requiring that $u_i \cap u_k \neq \emptyset$ and $u_j \cap u_k \neq \emptyset$. It is particularly advantageous if there

are several different choices of intermediary node U_k since this increases the chances that one of them is able to act as a direct relay between U_i and U_j in the physical graph. The following type of structure was first proposed in [45].

Definition 7.10 Let $(\mathcal{I}, \mathcal{B})$ be a (v, b, r, k)-configuration. We say that $(\mathcal{I}, \mathcal{B})$ is a (v, b, r, k, μ)-*common intersection design* (CID) if for any distinct pair of blocks $B_i, B_j \in \mathcal{B}$ we have: $|\{B_k \in \mathcal{B} \,|\, B_i \cap B_k \neq \emptyset \text{ and } B_j \cap B_k \neq \emptyset\}| \geq \mu$.

Clearly common intersection designs make ideal candidates for KRWSN scheme key rings, as well as being of intrinsic combinatorial interest in their own right. We have already seen one example in Scheme 7.8, since a TD(k, q) is an example of a $(qk, q^2, q, k, k(k-1))$-CID. Since we require μ to be as large as possible in a KRWSN scheme, an interesting question is to determine the maximum possible μ when fixing other parameters. Several upper bounds on μ were established in [48] and optimal CIDs were constructed using group-divisible designs, strongly-regular graphs and generalized quadrangles. Further investigation of CIDs is certainly merited.

7.3.4 Graph-based WSN schemes Given that one of our goals in a WSN scheme is to limit the number of hops between sensors who do not share a common key, another sensible design approach is to base the allocation of keys around a virtual *network graph*, whose vertices are sensors and whose edges join sensors who share a common key (in some sense this is the opposite approach to that taken in Section 7.3.3). We restrict our proposals in this section to pairwise WSN schemes, but the approach could be generalised using hypergraphs. This idea was again first proposed in the literature using random graphs (Scheme 5.8) but we will again see that combinatorial structures provided a more intuitive basis for construction. We will use the following generic scheme.

Scheme 7.11 *A* graph-based WSN scheme *(GWSN) for a graph $\mathcal{G} = (\mathcal{U}, \mathcal{E})$ is a pairwise WSN scheme based on an underlying node-based KPS where:*

- *Each edge $e \in \mathcal{E}$ is associated with a random key k_e;*
- *$u_i = \{k_e \,|\, U_i$ is adjacent to $e\}$;*
- *$\mathcal{C}^* = \{\{U_i, U_j\} \,|\, U_i$ and U_j are joined by an edge $e \in \mathcal{E}\}$.*

One difference between graph-based schemes and KRWSN schemes in general is that all graph-based schemes have full collusion security. In order to exploit this we need to define appropriate graphs on which to base a graph-based WSN scheme. In [46] it was pointed out that (n, r, λ, μ)-strongly regular graphs make ideal candidates, since by definition any pair of sensors in the graph that do not share a key are connected by μ two-hop paths. It was further demonstrated in [46] that careful choices of strongly regular graph have better local connectivity than schemes based on a random graph.

One problem with graph-based WSN schemes is that good connectivity often comes at the expense of requiring a graph where vertices typically have a high degree (and hence sensors have high storage). A clever efficiency improvement that can be applied to certain network graphs was observed in [46]:

Scheme 7.12 *let $\mathcal{G} = (\mathcal{U}, \mathcal{E})$ be a network graph that can be decomposed into star-like subgraphs. We can then establish a pairwise WSN scheme based on an underlying node-based KPS where:*

- *Each U_i is associated with an identifier ID_i and a random key k_i;*
- *$u_i = \{k_i \cup \{h(k_j, ID_i) \,|\, U_i$ is connected to a starlike subgraph centred at $U_j\}\}$, where h is a hash function;*
- *$\mathcal{C}^* = \{\{U_i, U_j\} \,|\, U_i$ and U_j are joined by an edge $e \in \mathcal{E}\}$.*

The above "trick" allows the sensor storage to be reduced compared to Scheme 7.11, since U_i only needs to store one key k_i for all the edges of the star-like subgraphs centred at U_i (it still needs to store a key for every other edge adjacent to U_i). This saving comes at the cost of reducing the security from unconditional to computational, since security is now dependent on the strength of the hash function.

The last scheme we will look at here is not strictly a graph-based scheme in the notion of Scheme 7.11, but it is based on a complete t-partite network graph. The scheme we describe is a generalisation to t-wise (from pairwise) of a scheme from [46].

Scheme 7.13 *Let $\mathcal{G}_{N_1,\ldots,N_t} = (\mathcal{I}, \mathcal{E})$ denote a complete t-partite graph on n vertices, based on a partition of $\mathcal{I}$ into subsets $N_1, \ldots, N_t$. The* t-partite *BDVHKY-KPS for $\mathcal{G}_{N_1,\ldots,N_t}$ is defined as follows, where $q \geq n$:*

- *$Pub_i = s_i$, where $s_i \in GF(q)$ and $Pub_i \neq Pub_j$ if $i \neq j$.*
- *The TA (randomly) constructs a secret t-variate polynomial f with coefficients from $GF(q)$,*

$$f(x_1, \ldots, x_t) = \sum_{i_1=0}^{w} \cdots \sum_{i_t=0}^{w} a_{i_1 \ldots i_t} x_1^{i_1} \ldots x_t^{i_t}.$$

- *If $U_i \in N_i$ then $u_i = f(x_1, \ldots, x_{i-1}, s_i, x_{i+1}, \ldots, x_t)$.*
- *$\mathcal{C}^* = \{A \,|\, A$ contains precisely one member of each $N_i\}$.*
- *For any $A = \{U_{z_1}, \ldots, U_{z_t}\} \in \mathcal{C}^*$, the user U_{z_i} computes $k_A = f(s_{z_1}, \ldots, s_{z_t})$.*

Note that the underlying polynomial in Scheme 7.13 differs from that in Scheme 5.9 by not necessarily being symmetric. The t-partite BDVHKY-KPS provides efficient key agreement since any t sensors A that are not in $\mathcal{C}^*$ must not contain any members of some partition subset N_l. Thus there are at least $|N_l|$ common neighbours of the sensors in A who can potentially act as a two-hop relay to all the sensors in A. Scheme 7.13 also has better resilience than Scheme 5.9, since now $w + 1$ sensors from the same partition subset N_i have to be captured before the scheme is broken.

7.3.5 Hybrid WSN schemes The last approach that we will briefly mention involves mixing WSN schemes with different properties. The first two ideas each "randomise" in a different way an underlying combinatorial design in order to create schemes that exhibit interesting tradeoffs in comparison to schemes that we have already seen. The third technique combines schemes in a combinatorial way.

- Recall that key ring WSN schemes based on KDPs, such as Scheme 7.7, suffer from the fact that if sensor storage is kept low, the number of possible sensors in the network is restricted. In [20] it was suggested that Scheme 7.7 be combined with a version of the RKPS (Scheme 5.8), essentially "topping up" the number of blocks in a projective plane with some random blocks. This leads to a degradation in the local connectivity but was shown in [20] to offer interesting tradeoffs between local connectivity and sensor storage.

- Recall that Scheme 7.8 was proposed as an alternative to Scheme 7.7 that permitted more sensors at the expense of a loss in connectivity. In [23] it was suggested that this connectivity loss can be avoided by randomly merging blocks of the underlying transversal design, thus creating a key ring with much longer blocks but greater connectivity. This idea thus improves connectivity at the expense of greater sensor storage.

- Recall that Scheme 5.10 was developed from Scheme 5.9 using a randomised product construction to improve resilience at the expense of a loss of connectivity. In [78] a deterministic product construction that combines multiple copies of a key predistribution scheme using a generic set system was studied. Combinatorial properties for desirable set systems were derived and it was shown that special types of 1-designs known as *difference families* made good candidates.

This concept of combining different types of WSN scheme merits further investigation.

7.4 Multisecret sharing schemes

The previous key agreement schemes that we have looked at involve users having to collaborate to construct a group key for practical reasons (such as making key establishment efficient or through restrictions in the connectivity of the network). We now look at a family of key agreement schemes were users are forced to collaborate to construct a group key for *security* reasons. This is most likely to happen in applications where the group keys protect sensitive assets, with no single user being trusted with the sole authority to access them.

Definition 7.14 Let $\mathcal{C} = \{A_1, \ldots, A_m\}$ be a communication structure defined on a set $\mathcal{U}$. An *access structure* for $\mathcal{C}$ is a collection $\Gamma = \{\Gamma_1, \ldots, \Gamma_m\}$ of subsets of $\mathcal{U}$ with the property that:

1. Γ_i consists of subsets of A_i;

2. Γ_i is *monotone* (in other words, if $X \in \Gamma_i$ and $X \subseteq Y \subseteq A_i$ then $Y \in \Gamma_i$).

We will use an access structure on $\mathcal{C}$ to specify the degree of mandatory collaboration between users that is required before a group key can be established. More precisely, we will require the property that the group key k_{A_i} can be established only if users belonging to a set in Γ_i collaborate. This is more clearly specified in the following definition.

Definition 7.15 Let $\mathcal{C} = \{A_1, \ldots, A_m\}$ be a communication structure and $\mathcal{X}$ be an exclusion structure defined on a set $\mathcal{U}$, and let $\Gamma = \{\Gamma_1, \ldots, \Gamma_m\}$ be an access structure for $\mathcal{C}$. A $(\mathcal{C}, \mathcal{X}, \Gamma)$-*multisecret sharing scheme* is a $(\mathcal{C}, \mathcal{X})$-key establishment scheme such that for any set of users B:

1. If $(B \cap A_i) \in \Gamma_i$ then there exists a public function g such that $g(\{u_i \,|\, U_i \in B\}) = k_{A_i}$. In other words, the users in B can construct k_{A_i} from their collective set of secret values.

2. If $(B \cap A_i) \notin \Gamma_i$ and $B \in \mathcal{X}$ then, even if users in B exchange all their secret values $\{u_i \,|\, U_i \in B\}$, they will not learn any information about k_{A_i}.

Note that we have deliberately avoided formulating the notion of *not learning any information* and refer to [40] for a combinatorial formalisation and [54] for an information-theoretic formalisation of this concept. We have also avoided a detailed discussion of how the users exchange their values u_i and apply the public function g (typically it is either assumed that users share secret channels or that there exists an entity called a *combiner* that performs this task for them).

Multisecret sharing schemes are generalisations of secret sharing schemes (see Section 4.5), which correspond to the case of a scheme with just one group A_1 (associated with Γ_1) in its communication structure. While bounds on the secret storage have been established for general multisecret sharing schemes under a couple of different threat models [16, 54], we will restrict our attention to the special case of multisecret threshold schemes, defined as follows.

Definition 7.16 A (t, w, λ)-*multisecret threshold scheme* (MTS) is a special class of $(\mathcal{C}, \mathcal{X}, \Gamma)$-*multisecret sharing scheme* where:

1. $\mathcal{C} = \{A \subseteq \mathcal{U} \,|\, |A| = t\}$;

2. $\mathcal{X} = \{A \subseteq \mathcal{U} \,|\, |A| \leq w\}$;

3. For each $A_i \in \mathcal{C}$, $\Gamma_i = \{X \subseteq A_i \,|\, |X| \geq \lambda\}$.

A $(t, w, 1)$-MTS corresponds to a (t, w)-KPS (see Section 5) since in this case there is no requirement for users to collaborate to construct their group keys. In [40] it was shown that for most meaningful choices of w, each user in a (t, w, λ)-MTS needs to be given a secret value u_i that is at least $\binom{w+t-2\lambda+1}{t-\lambda}$ times larger than the size of any key k_{A_i} in the system. This bound is a generalisation of the bound on user storage for KPSs proved in [17] (see Section 5.2). It is thus of particular interest to find MTSs that meet this bound. The following are all "degenerate" cases of optimal MTSs:

- Scheme 5.9 [17] is an optimal $(t, w, 1)$-MTS;

- Optimal (n, w, λ)-MTSs (where $|\mathcal{U}| = n$) correspond to optimal secret sharing schemes (more precisely, if $w = \lambda - 1$ they correspond to *ideal threshold schemes* such as the classical scheme in [69] and for general w they correspond to optimal *ramp schemes*, see [38]);

- An optimal (t, w, t)-MTS is easily constructed by letting u_i be randomly chosen in $GF(q)$ and letting $k_A = \sum_{U_i \in A} u_i$ for any $A \in \mathcal{C}$ [40].

However, the task of constructing optimal MTSs with $1 < \lambda < t$ appears to be intriguingly difficult and to date only two constructions are known. In [41] a family of optimal $(t, n - k + 1, 2)$-MTSs were constructed and in [5] a family of optimal $(3, w, 2)$-MTSs. Both these constructions were based on complex and rather intricate projective geometrical configurations and used a geometrical interpretation of Scheme 5.9 as a building block.

8 Concluding remarks

In this paper we have reviewed a wide variety of applications of combinatorics objects, including designs and graphs, to different types of key establishment scheme. The theory of group key establishment is by no means complete and there are a number of areas where combinatorics can make further contributions to our understanding. Some specific areas where more research would be beneficial include:

- Several combinatorial objects that have direct application to key establishment merit further investigatory work:

  - While a moderate amount of research has been conducted on key distribution patterns (cover free families), there is a great deal of information about these structures to learn. In particular very little is known about KDPs for non-threshold communication structures.
  - Hash-tree key distribution patterns are relatively newly proposed structures and more theoretical work needs to be done concerning both constructions and performance bounds.
  - Cover-based revocation systems have attracted a great deal of interest in the area of broadcast encryption and greater understanding is needed of how efficiently these can be implemented.
  - Common intersection designs provide an interesting solution to the problem of key establishment in wireless sensor networks. More knowledge of how to generate constructions with useful parameters is required.

- There has been some interesting preliminary research conducted on how to take key establishment schemes with nice mathematical structure and convert them into more practical schemes with less inherent structure, but better performance. This can either be through merging schemes, extending schemes or simply using a mathematical scheme as a starting point on which to build a practical solution (Section 7.3.5 describes some work of this type for group key agreement schemes). This area certainly merits further investigation.

- Most of the schemes that we have presented in this review have been discussed in their most basic form. We have not discussed how they can be extended to incorporate all of the extended capabilities mentioned in Section 3.2.5. There remains plenty work to de done in designing schemes with extended capabilities, the most important of which is probably flexibility, in other words the ability to efficiently process dynamic changes to the communication structure over time.

It is hoped that this review has provided convincing evidence that combinatorial mathematics has already made a substantial contribution to the theory of key establishment, and that we can expect it to continue to do so in future cryptographic systems.

References

[1] S.G. Akl and P.D. Taylor. Cryptographic solution to a problem of access control in a hierarchy. *ACM Transactions on Computer Systems*, 1(3):239–248, 1983.

[2] T. Asano. A revocation scheme with minimal storage at receivers. In *Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security*, volume 3108 of *Lecture Notes in Computer Science*, pages 433–450. Springer-Verlag, 2002.

[3] T. Asano. Secure and insecure modifications of the subset difference broadcast encryption scheme. In *Information Security and Privacy: ACISP '04*, volume 3108 of *Lecture Notes in Computer Science*, pages 12–23. Springer-Verlag, 2004.

[4] M.J. Atallah, K.B. Frikken, and M. Blanton. Dynamic and efficient key management for access hierarchies. In *Proceedings of 12th ACM Conference on Computer and Communications Security*, pages 190–202, 2005.

[5] S.G. Barwick and W.-A. Jackson. An optimal multisecret threshold scheme construction. *Designs Codes and Crytography*, 37:367–389, 2005.

[6] A. Beimel and B. Chor. Communication in key distribution schemes. *IEEE Transactions on Information Theory*, 42:19–28, 1996.

[7] S.R. Blackburn. Combinatorial schemes for protecting digital content. In *Surveys in Combinatorics 2003*, pages 43–78. Cambridge University Press, 2003.

[8] B. Blakley. Safeguarding cryptographic keys. In *Proceedings AFIPS 1979 National Computer Conference*, pages 313–317, June 1979.

[9] R. Blom. An optimal class of symmetric key generation systems. In *Eurocrypt '84*, volume 209 of *Lecture Notes in Computer Science*, pages 335–338. Springer-Verlag, 1985.

[10] C. Blundo and P. D'Arco. The key establishment problem. In *FOSAD 2001/2002*, volume 2946 of *Lecture Notes in Computer Science*, pages 44–90. Springer-Verlag, 2004.

[11] C. Blundo and P. D'Arco. Analysis and design of distributed key distribution centers. *Journal of Cryptology*, 18(4):391–414, 2005.

[12] C. Blundo, P. D'Arco, and C. Padrò. A ramp model for distributed key distribution schemes. *Discrete Applied Mathematics*, 128:47–64, 2003.

[13] C. Blundo, P. D'Arco, and A. De Santis. On self-healing key distribution schemes. To appear in IEEE Transactions on Information Theory.

[14] C. Blundo, P. D'Arco, A. De Santis, and M. Listo. Design of self healing key distribution schemes. *Designs Codes and Cryptography*, 32:15–44, 2004.

[15] C. Blundo, L. Frota Mattos, and D.R. Stinson. Generalized beimel-chor schemes for broadcast encryption and interactive key distribution. *Theoretical Computer Science*, 200:313–334, 1998.

[16] C. Blundo, A. De Santis, G. Di Crescenzo, A. Giorgio Gaggia, and U. Vaccaro. Multi-secret sharing schemes. In *Crypto '94*, volume 839 of *Lecture Notes in Computer Science*, pages 150–163. Springer-Verlag, 1994.

[17] C. Blundo, A. De Santis, U. Vaccaro, A. Herzberg, S. Kutten, and M. Yung. Perfectly secure key distribution for dynamic conferences. In *Crypto '92*, volume 740 of *Lecture Notes in Computer Science*, pages 471–486. Springer-Verlag, 1993.

[18] C. Boyd and A. Mathuria. *Protocols for authentication and key establishment.* Springer-Verlag, 2003.

[19] E.F. Brickell and D.M. Davenport. On the classification of ideal secret sharing schemes. *Journal of Cryptology*, 4:123–134, 1991.

[20] S. A. Camtepe and B. Yener. Combinatorial design of key distribution mechanisms for wireless sensor networks. In *ESORICS 2004*, volume 3193 of *Lecture Notes in Computer Science*, pages 293–308. Springer-Verlag, 2004.

[21] S. A. Camtepe and B. Yener. Key distribution mechanisms for wireless sensor networks: a survey. Rensselaer Polytechnic Institute, Computer Science Department, Technical Report TR-05-07, March 2005.

[22] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: a taxonomy and some efficient constructions. In *Proceedings of INFOCOM '99*, pages 708–716. IEEE Press, 1999.

[23] D. Chakrabarti, S. Maitra, and B. Roy. A hybrid design of key pre-distribution scheme for wireless sensor networks. In *ICISS 2005*, volume 3803 of *Lecture Notes in Computer Science*, pages 228–238. Springer-Verlag, 2005.

[24] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Research in Security and Privacy*, pages 197–213, May 2003.

[25] B. Chor, A. Fiat, M. Naor, and B. Pinkas. Traitor tracing. *IEEE Transactions on Information Theory*, 46:893–910, 2000.

[26] C.J. Colbourn, J.H. Dinitz, and D.R. Stinson. Applications of combinatorial designs to communications, cryptography and networking. In *Surveys in Combinatorics 1999*, pages 37–100. Cambridge University Press, 1999.

[27] J. Crampton, K.M. Martin, and P.R. Wild. An explication of key assignment schemes. In preparation, 2006.

[28] J. Crampton, K.M. Martin, and P.R. Wild. Proceedings of 19th computer security foundations workshop. pages 98–111, 2006.

[29] P. D'Arco and D.R. Stinson. On unconditionally secure robust distributed key distribution centers. In *ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 346–363. Springer-Verlag, 2002.

[30] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[31] R. Dutta and R. Barua. Overview of key agreement protocols. Cryptology ePrint Archive, Report 2005/289, 2005. http://eprint.iacr.org/.

[32] M. Dyer, T. Fenner, and A. Thomason. On key storage in secure networks. *Journal of Cryptography*, 8:189–200, 1995.

[33] L. Eschenauer and V. Gligor. A key management scheme for distributed sensor networks. In *Proceedings of 9th ACM Conference on Computer and Communication Security*, November 2002.

[34] A.L. Ferrara and B. Masucci. An information-theoretic approach to the access control problem. In C. Blundo and C. Laneve, editors, *ICTCS 2003*, volume 2841 of *Lecture Notes in Computer Science*, pages 342–354. Springer-Verlag, 2003.

[35] A. Fiat and M. Naor. Broadcast encryption. In *Advances in Cryptology - CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer-Verlag, 1994.

[36] D. Halevy and A. Shamir. The lsd broadcast encryption scheme. In *Crypto '02*, volume 2442 of *Lecture Notes in Computer Science*, pages 47–60. Springer-Verlag, 2002.

[37] L. Harn and H.Y. Lin. A cryptographic key generation scheme for multilevel data security. *Computers and Security*, 9(6):539–546, 1990.

[38] W.-A. Jackson and K.M. Martin. A combinatorial interpretation of ramp schemes. *Australasian Journal of Combinatorics*, 14:51–60, 1996.

[39] W.-A. Jackson and K.M. Martin. Combinatorial models for perfect secret sharing schemes. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 28:249–265, 1998.

[40] W.-A. Jackson, K.M. Martin, and C.M. O'Keefe. Multisecret threshold schemes. In *Crypto '93*, volume 773 of *Lecture Notes in Computer Science*, pages 126–135. Springer-Verlag, 1994.

[41] W.-A. Jackson, K.M. Martin, and C.M. O'Keefe. A construction for multisecret threshold schemes. *Designs Codes and Crytography*, 9:287–303, 1996.

[42] Y. Kim, A. Perrig, and G. Tsudik. Tree-based group key agreement. *ACM Transactions on Information and System Security*, 7(1):60–96, 2004.

[43] R. Kumar, S. Rajagopalan, and A. Sahai. Coding constructions for blacklisting problems without computational assumptions. In *Advances in Cryptology - CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 609–623. Springer-Verlag, 1999.

[44] K. Kurosawa, K. Okada, and K. Sakano. Security of the center in key distribution schemes. In *Asiacrypt '94*, volume 917 of *Lecture Notes in Computer Science*, pages 333–341. Springer-Verlag, 1995.

[45] J. Lee and D.R. Stinson. A combinatorial approach to key predistribution for distributed sensor networks. In *IEEE Wireless Communications and Networking Conference*, pages 6–11, 2005. CD-ROM, paper PHY53-06, http://www.cacr.math.uwaterloo.ca/dstinson/pubs.html.

[46] J. Lee and D.R. Stinson. Deterministic key predistribution schemes for distributed sensor networks. In *SAC 2004*, volume 3357 of *Lecture Notes in Computer Science*, pages 294–307. Springer-Verlag, 2005.

[47] J. Lee and D.R. Stinson. One the construction of practical key predistribution schemes for distributed sensor networks using combinatorial designs. http://www.cacr.math.uwaterloo.ca/ dstinson/pubs.html, November 2005.

[48] J. Lee and D.R. Stinson. Common intersection designs. *Journal of Combinatorial Designs*, 14:251–269, 2006.

[49] J. Lee and D.R. Stinson. Tree-based key distribution patterns. In *SAC 2005 Proceedings*, volume 3897 of *Lecture Notes in Computer Science*, pages 189–204. Springer-Verlag, 2006.

[50] T. Leighton and S. Micali. Secret-key agreement without public-key cryptography. In *Advances in Cryptology - CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 456–479. Springer-Verlag, 1994.

[51] C.-H. Lin. Hierarchical key assignment without public key cryptography. *Computers & Security*, 20(7):612–619, 2001.

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

[53] S.J. MacKinnon, P.D. Taylor, H. Meijer, and S.G. Akl. An optimal algorithm for assigning cryptographic keys to control access in a hierarchy. *IEEE Transactions on Computers*, C-34(9):797–802, 1985.

[54] B. Masucci. Sharing multiple secrets: models, schemes and analysis. *Designs Codes and Crytography*, 39:89–111, 2006.

[55] M. J. Mihaljevic. Key management schemes for stateless receivers based on time varying heterogeneous logical key hierarchy. In *Asiacrypt 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 127–154. Springer-Verlag, 2003.

[56] C. J. Mitchell and F.C. Piper. The cost of reducing key storage requirements in secure networks. *Computers and Security*, 6:339–341, 1987.

[57] C. J. Mitchell and F.C. Piper. Key storage in secure networks. *Discrete Applied Mathematics*, 21:215–228, 1988.

[58] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology - CRYPTO '01*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer-Verlag, 2001.

[59] M. Naor, B. Pinkas, and O. Reingold. Distributed preudo-random functions and kdcs. In *Advances in Cryptology - Eurocrypt '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 327–346. Springer-Verlag, 1999.

[60] C.M. O'Keefe. A comparison of key distribution patterns constructed from circle geometries. In *Auscrypt '92*, volume 718 of *Lecture Notes in Computer Science*, pages 517–527. Springer-Verlag, 1993.

[61] C.M. O'Keefe. Key distribution patterns using minkowski planes. *Designs Codes and Crytography*, 5:261–267, 1995.

[62] C. Padró, I. Gracia, S.M. Molleví, and P. Morillo. Linear key predistribution schemes. *Designs Codes and Cryptography*, 25:281–298, 2002.

[63] F.C. Piper and S. Murphy. *Cryptography: a very short introduction*. Oxford University Press, 2002.

[64] K.A.S. Quinn. Some constructions for key distribution patterns. *Designs Codes and Cryptography*, 4:177–191, 1994.

[65] K.A.S. Quinn. Bounds for key distribution patterns. *Journal of Cryptology*, 12:227–239, 1999.

[66] M. Ramkumar and N. Memon. An efficient random key pre-distribution scheme for manet security. *IEEE Journal on Selected Areas of Communication*, 2005.

[67] G. Rinaldi. Key distribution patterns using tangent circle structures. *Designs Codes and Crytography*, 31:289–300, 2004.

[68] A. De Santis, A.L. Ferrara, and B. Masucci. Cryptographic key assignment schemes for any access control policy. *Information Processing Letters*, 92:199–2005, 2004.

[69] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[70] J. Staddon, S. Miner, M. Franklin, D. Balfanz, M. Malkin, and D. Dean. Self-healing key distribution with revocation. *IEEE Symposium on Security and Privacy*, May 2002.

[71] D.R. Stinson. An explication of secret sharing schemes. *Designs Codes and Cryptography*, 2:357–390, 1992.

[72] D.R. Stinson. On some methods of unconditionally secure key distribution and broadcast encryption. *Designs Codes and Cryptography*, 12:215–243, 1997.

[73] D.R. Stinson. *Cryptography: theory and practice*. Chapman & Hall/CRC, 3rd edition, 2006.

[74] D.R. Stinson and T. Van Trung. Some new results on key distribution patterns and broadcast encryption. *Designs Codes and Cryptography*, 14:261–279, 1998.

[75] D.R. Stinson and R. Wei. An application of ramp schemes to broadcast encryption. *Information Processing Letters*, 69:131–135, 1999.

[76] D.R. Stinson and R. Wei. Generalized cover free families. *Discrete Mathematics*, 279:463–477, 2004.

[77] D.M. Wallner, E.J. Harder, and R.C. Agee. Key management for multicast: issues and architectures. Internet Request for Comments 2627, June, 1999.

[78] R. Wei and J. Wu. Product construction of key distribution schemes for sensor networks. In *SAC 2004*, volume 3357 of *Lecture Notes in Computer Science*, pages 280–293. Springer-Verlag, 2005.

[79] C.K. Wong, M.G. Gouda, and S.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.

[80] Y. Zheng, T. Hardjono, and J. Seberry. New solutions to the problem of access control in a hierarchy. Technical Report 93-2, Department of Computer Science, University of Wollongong, 1993.

[81] S. Zhong. A practical key management scheme for access control in a user hierarchy. *Computers & Security*, 21(8):750–759, 2002.

Information Security Group
Royal Holloway, University of London
Egham Hill, Egham, Surrey TW20 0EX, UK
keith.martin@rhul.ac.uk