# On the Applicability of Combinatorial Designs to Key Predistribution for Wireless Sensor Networks

Keith M. Martin

Information Security Group
Royal Holloway, University of London, U.K.

keith.martin@rhul.ac.uk

IWCC2009

# This presentation

There have been a number of proposals to use combinatorial designs in different ways to construct key predistribution schemes for wireless sensor networks.

We will provide an overview of these techniques and ask:

**Question**

*Are these uses of combinatorial designs appropriate?*

# The Plan

Wireless sensor networks

Combinatorial designs

Key predistribution for WSNs

Direct application of designs

Designs as building blocks

Special networking environments

Conclusions

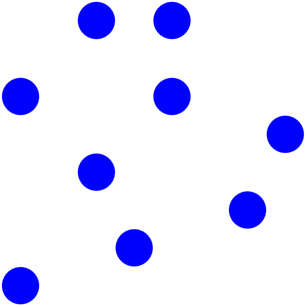
# Wireless sensor networks

A major computing trend is towards:

- distributed,
- dynamic,
- wireless

networks consisting of lightweight devices, such as wireless sensor networks (WSNs).

# Wireless sensor networks

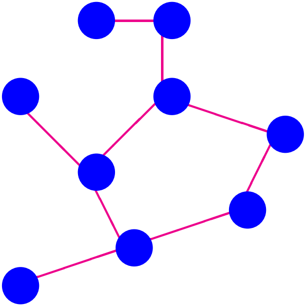


A major computing trend is towards:

- distributed,
- dynamic,
- wireless

networks consisting of lightweight devices, such as wireless sensor networks (WSNs).

# Wireless sensor networks



A major computing trend is towards:

- distributed,
- dynamic,
- wireless

networks consisting of lightweight devices, such as wireless sensor networks (WSNs).

# Wireless sensor network characteristics

WSNs are characterised by:

# Wireless sensor network characteristics

WSNs are characterised by:

- Highly constrained nodes: Very small battery-powered devices, highly constrained with respect to memory, storage and power.

# Wireless sensor network characteristics

WSNs are characterised by:

- Highly constrained nodes: Very small battery-powered devices, highly constrained with respect to memory, storage and power.
- Lack of central control: After deployment, all network functionality must be achieved through co-operation between the nodes.

# Wireless sensor network characteristics

WSNs are characterised by:

- Highly constrained nodes: Very small battery-powered devices, highly constrained with respect to memory, storage and power.
- Lack of central control: After deployment, all network functionality must be achieved through co-operation between the nodes.
- Requirement to connect to a sink: Nodes will communicate data back to a sink.

# Wireless sensor network characteristics

WSNs are characterised by:

- Highly constrained nodes: Very small battery-powered devices, highly constrained with respect to memory, storage and power.
- Lack of central control: After deployment, all network functionality must be achieved through co-operation between the nodes.
- Requirement to connect to a sink: Nodes will communicate data back to a sink.
- Hop-based communication: Nodes communicate by hopping (a node passes data to a node within range).

# Wireless sensor network characteristics

WSNs are characterised by:

- Highly constrained nodes: Very small battery-powered devices, highly constrained with respect to memory, storage and power.
- Lack of central control: After deployment, all network functionality must be achieved through co-operation between the nodes.
- Requirement to connect to a sink: Nodes will communicate data back to a sink.
- Hop-based communication: Nodes communicate by hopping (a node passes data to a node within range).
- Dynamic network structure: Nodes regularly sleep and expire.

# Wireless sensor network characteristics

WSNs are characterised by:

- Highly constrained nodes: Very small battery-powered devices, highly constrained with respect to memory, storage and power.
- Lack of central control: After deployment, all network functionality must be achieved through co-operation between the nodes.
- Requirement to connect to a sink: Nodes will communicate data back to a sink.
- Hop-based communication: Nodes communicate by hopping (a node passes data to a node within range).
- Dynamic network structure: Nodes regularly sleep and expire.
- Nodes vulnerable to compromise. Physical security protection such as tamper-resistance is usually not viable, thus nodes can be fairly easily captured.

# Three assumptions

1. Homogeneous nodes: We will assume that all nodes have the same capabilities and constraints.

# Three assumptions

1. Homogeneous nodes: We will assume that all nodes have the same capabilities and constraints.
2. Communication structure: We will assume that the main aim of any communication in the WSN is to send data from a node to the sink. We will thus not be attempting to set up fully connected subnetworks or establish group keys.

# Three assumptions

1. Homogeneous nodes: We will assume that all nodes have the same capabilities and constraints.
2. Communication structure: We will assume that the main aim of any communication in the WSN is to send data from a node to the sink. We will thus not be attempting to set up fully connected subnetworks or establish group keys.
3. No mobility: We will assume that nodes are not mobile after deployment.

# Classification of WSN environments

1. Uncontrolled if the location of sensors cannot be predicted before deployment. This is the default WSN scenario.

# Classification of WSN environments

1. Uncontrolled if the location of sensors cannot be predicted before deployment. This is the default WSN scenario.
2. Partially controlled if some information about the location of sensors is known before deployment.

# Classification of WSN environments

1. Uncontrolled if the location of sensors cannot be predicted before deployment. This is the default WSN scenario.
2. Partially controlled if some information about the location of sensors is known before deployment.
3. Fully controlled if the precise location of sensors is known before deployment.

# Cryptography in WSN environments

- We assume that the constraints lend themselves to the use of symmetric cryptography.
- There has been some debate about whether public key cryptography can be used on sensor nodes:

# Cryptography in WSN environments

- We assume that the constraints lend themselves to the use of symmetric cryptography.
- There has been some debate about whether public key cryptography can be used on sensor nodes:
  - Even if it can, symmetric cryptography may be preferred for efficiency reasons.
  - Related technology with smaller "sensors" is likely to be proposed.

# Sets systems

## Definition

*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 $\mathcal{I}$.*

# Sets systems

## Definition

*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 $\mathcal{I}$.*

## Definition

*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 $r$.*

# Sets systems

## Definition

*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* $\mathcal{I}$.

## Definition

*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* $r$.

## Definition

*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* $k$.

# Designs

## Definition

*A regular, uniform set system with* $|\mathcal{I}| = v$, $|\mathcal{B}| = b$ *is known as a* $(v, b, r, k)$*-design. In such designs it must be the case that* $bk = vr$.

## Designs

### Definition

*A regular, uniform set system with* $|\mathcal{I}| = v$, $|\mathcal{B}| = b$ *is known as a* $(v, b, r, k)$*-design. In such designs it must be the case that* $bk = vr$.

### Definition

*A* $(v, b, r, k)$*-design in which every* $t$ *points occurs on precisely* $\lambda$ *blocks is known as a* $t$*-*$(v, b, r, k, \lambda)$*-design (we often just refer to a* $t$*-*$(v, k, \lambda)$*-design since* $b$ *and* $r$ *can then be uniquely derived).*

# Designs

## Definition

*A regular, uniform set system with $|\mathcal{I}| = v$, $|\mathcal{B}| = b$ is known as a $(v, b, r, k)$-design. In such designs it must be the case that $bk = vr$.*

## Definition

*A $(v, b, r, k)$-design in which every $t$ points occurs on precisely $\lambda$ blocks is known as a $t$-$(v, b, r, k, \lambda)$-design (we often just refer to a $t$-$(v, k, \lambda)$-design since $b$ and $r$ can then be uniquely derived).*

## Definition

*In a dual design, the roles of points and blocks are interchanged.*

# Special designs

## Definition

*Symmetric designs are self-dual and thus have* $v = b$, $k = r$ *and every* $t$ *blocks meeting in* $\lambda$ *points. A symmetric* 2-$(s^2 + s + 1, s^2 + s + 1, s + 1, s + 1, 1)$*-design is known as a projective plane.*

# Special designs

## Definition

*Symmetric designs are self-dual and thus have $v = b$, $k = r$ and every $t$ blocks meeting in $\lambda$ points. A symmetric 2-$(s^2 + s + 1, s^2 + s + 1, s + 1, s + 1, 1)$-design is known as a projective plane.*

## Definition

*A transversal design $TD(k, n)$ is set system consisting of $nk$ points such that there exists a partition $\mathcal{H}$ of $\mathcal{I}$ into $k$ groups of size $n$ such that:*

1. *Every $H \in \mathcal{H}$ intersects a block $B \in \mathcal{B}$ in precisely one point;*
2. *Every pair of points from different groups occur together in precisely one block.*

# Key establishment

- One of the most important key management processes is key establishment, which governs the placement of cryptographic keys in a network.

# Key establishment

- One of the most important key management processes is key establishment, which governs the placement of cryptographic keys in a network.
- This is especially relevant in applications of symmetric cryptography, where it is necessary to ensure that all parties who are authorised to access (or verify) a cryptographically protected piece of information have the appropriate key.

# Key establishment

- One of the most important key management processes is key establishment, which governs the placement of cryptographic keys in a network.
- This is especially relevant in applications of symmetric cryptography, where it is necessary to ensure that all parties who are authorised to access (or verify) a cryptographically protected piece of information have the appropriate key.
- Most key establishment mechanisms involve a key management authority (KMA) at some stage in the process.

# Key predistribution

## Definition

*A key predistribution scheme is a key establishment scheme in which the KMA can only be involved in initialisation processes that take place prior to deployment of the network.*

A KMA thus needs to load keys onto nodes prior to deployment using a KPS to determine which keys are allocated to which nodes.

# Key predistribution



### Definition

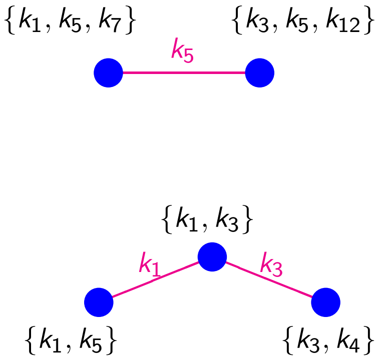
*A key predistribution scheme is a key establishment scheme in which the KMA can only be involved in initialisation processes that take place prior to deployment of the network.*

A KMA thus needs to load keys onto nodes prior to deployment using a KPS to determine which keys are allocated to which nodes.

After deployment, two nodes will be able to use a cryptographic service on a network link if they:

1. are in radio communication range of one another; and
2. share at least one key.

# Example



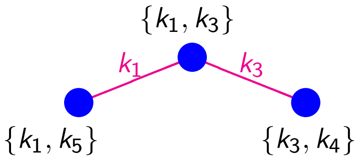

Key predistribution scheme

- nodes are assigned keys before deployment

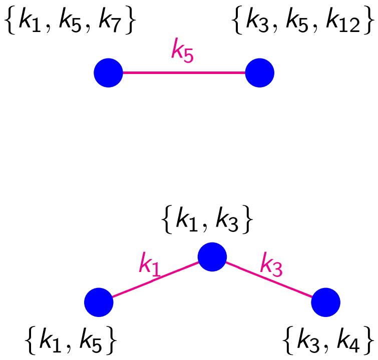
# Example


**Key predistribution scheme**

- nodes are assigned keys before deployment
- nodes that share keys can communicate securely



# Example




Key predistribution scheme

- nodes are assigned keys before deployment
- nodes that share keys can communicate securely
- two-hop path: nodes communicate via intermediate node

# Phase 1: Key predistribution

- The KMA chooses a KPS defined on the $n$ nodes $\mathcal{U} = \{U_1, \ldots, U_n\}$ in the network.

# Phase 1: Key predistribution

- The KMA chooses a KPS defined on the $n$ nodes $\mathcal{U} = \{U_1, \ldots, U_n\}$ in the network.
- This KPS can be modelled by a set system $(\mathcal{I}, \mathcal{B})$, where $\mathcal{I} = \{x_i : 1 \leq i \leq v\}$ is a set of $v$ key identifiers and $\mathcal{B} = \{B_j : 1 \leq j \leq n\}$ is a set of $n$ node allocations.

# Phase 1: Key predistribution

- The KMA chooses a KPS defined on the $n$ nodes $\mathcal{U} = \{U_1, \ldots, U_n\}$ in the network.
- This KPS can be modelled by a set system $(\mathcal{I}, \mathcal{B})$, where $\mathcal{I} = \{x_i : 1 \leq i \leq v\}$ is a set of $v$ key identifiers and $\mathcal{B} = \{B_j : 1 \leq j \leq n\}$ is a set of $n$ node allocations.
- For each key identifier $x_i$, KMA randomly selects a key $K_i$.

# Phase 1: Key predistribution

- The KMA chooses a KPS defined on the $n$ nodes $\mathcal{U} = \{U_1, \ldots, U_n\}$ in the network.
- This KPS can be modelled by a set system $(\mathcal{I}, \mathcal{B})$, where $\mathcal{I} = \{x_i : 1 \leq i \leq v\}$ is a set of $v$ key identifiers and $\mathcal{B} = \{B_j : 1 \leq j \leq n\}$ is a set of $n$ node allocations.
- For each key identifier $x_i$, KMA randomly selects a key $K_i$.
- The KMA associates each node $U_j$ in the network with a node allocation $B_j$ and issues $U_j$ with the keys $L_j = \{K_i : x_i \in B_j\}$.

# Phase 1: Key predistribution

- The KMA chooses a KPS defined on the $n$ nodes $\mathcal{U} = \{U_1, \ldots, U_n\}$ in the network.
- This KPS can be modelled by a set system $(\mathcal{I}, \mathcal{B})$, where $\mathcal{I} = \{x_i : 1 \leq i \leq v\}$ is a set of $v$ key identifiers and $\mathcal{B} = \{B_j : 1 \leq j \leq n\}$ is a set of $n$ node allocations.
- For each key identifier $x_i$, KMA randomly selects a key $K_i$.
- The KMA associates each node $U_j$ in the network with a node allocation $B_j$ and issues $U_j$ with the keys $L_j = \{K_i : x_i \in B_j\}$.
- The association of $U_j$ with $B_j$ need not be a secret, however the instantiation of $B_j$ by $L_j$ must be.

# Phase 2: Shared key discovery

If two nodes within communication range of one another wish to deploy a cryptographic service:

- They first need to determine if they have any keys in common.

# Phase 2: Shared key discovery

If two nodes within communication range of one another wish to deploy a cryptographic service:

- They first need to determine if they have any keys in common.
- The default method is to broadcast their node allocations to one another, but more efficient techniques can sometimes be found.

# Phase 2: Shared key discovery

If two nodes within communication range of one another wish to deploy a cryptographic service:

- They first need to determine if they have any keys in common.
- The default method is to broadcast their node allocations to one another, but more efficient techniques can sometimes be found.
- If they have key identifiers in common then a session key can be generated from the common keys associated with these identifiers by means of a suitable key derivation function.

# Phase 3: Path-key establishment



If two nodes fail to identify common keys during shared key discovery then they need to find a secure path between one another that employs intermediate nodes which can.

Obviously, the shorter this secure path the better.

# Competing requirements

- Storage: The number of keys stored on each node should be kept as low as possible.

# Competing requirements

- Storage: The number of keys stored on each node should be kept as low as possible.
- Connectivity: Each node should store sufficient keys that secure paths through the network can be established when needed.

# Competing requirements

- Storage: The number of keys stored on each node should be kept as low as possible.
- Connectivity: Each node should store sufficient keys that secure paths through the network can be established when needed.
- Resilience: Keys should be distributed in such a way that the damage caused by exposure of the keys stored on a node is controlled.

# Competing requirements

- Storage: The number of keys stored on each node should be kept as low as possible.
- Connectivity: Each node should store sufficient keys that secure paths through the network can be established when needed.
- Resilience: Keys should be distributed in such a way that the damage caused by exposure of the keys stored on a node is controlled.
- Efficiency: Processes such as computation, shared key discovery, and path-key establishment should be as efficient as possible.

# Competing requirements

- Storage: The number of keys stored on each node should be kept as low as possible.
- Connectivity: Each node should store sufficient keys that secure paths through the network can be established when needed.
- Resilience: Keys should be distributed in such a way that the damage caused by exposure of the keys stored on a node is controlled.
- Efficiency: Processes such as computation, shared key discovery, and path-key establishment should be as efficient as possible.
- Network size: It is important that a KPS can support as large number of nodes as possible.

# Two trivial KPSs

Single Key KPS: This KPS consists of a single key that is stored by each node in the network. It provides:

- optimal connectivity
- optimal storage
- poor resilience.

# Two trivial KPSs

Single Key KPS: This KPS consists of a single key that is stored by each node in the network. It provides:

- optimal connectivity
- optimal storage
- poor resilience.

Complete Pairwise Key KPS: In this KPS, a unique key is assigned to each pair of nodes. It provides:

- optimal connectivity
- optimal resilience
- poor storage

# Blom's KPS

- Based on a polynomial $P(x, y) \in \mathrm{GF}(q)[x, y]$ with the property that $P(i, j) = P(j, i)$ for all $i, j \in \mathrm{GF}(q)$.
- Node $U_i$ stores the univariate polynomial $f_i(y) = P(U_i, y)$.
- In order to establish a common key with $U_j$, node $U_i$ computes $K_{ij} = f_i(U_j) = f_j(U_i)$.

# Blom's KPS

- Based on a polynomial $P(x, y) \in \mathrm{GF}(q)[x, y]$ with the property that $P(i, j) = P(j, i)$ for all $i, j \in \mathrm{GF}(q)$.
- Node $U_i$ stores the univariate polynomial $f_i(y) = P(U_i, y)$.
- In order to establish a common key with $U_j$, node $U_i$ computes $K_{ij} = f_i(U_j) = f_j(U_i)$.

- optimal connectivity
- if $P$ has degree $w$ then:
  - each node stores $w + 1$ co-efficients
  - full resilience up to capture of $w$ nodes
- very efficient shared key discovery
- does not strictly conform to our model

# Random KPS

- Probabilistic KPS
- Each node draws keys uniformly without replacement from some finite keypool $\mathcal{K}$.
- Properties depend on the number of keys drawn and the size of $\mathcal{K}$.
- Can be further parameterised (for example to require a threshold number of common keys).

# KPS design observations

1. Optimal connectivity is not necessary:
   - Most pairs of nodes never communicate directly.

# KPS design observations

1. Optimal connectivity is not necessary:
   - Most pairs of nodes never communicate directly.
2. Deterministic schemes have some advantages:
   - Analysis is easier
   - Efficient shared key discovery possible

# KPS design observations

1. Optimal connectivity is not necessary:
   - Most pairs of nodes never communicate directly.
2. Deterministic schemes have some advantages:
   - Analysis is easier
   - Efficient shared key discovery possible
3. Flexibility is attractive:
   - Competing requirements require flexibility

# KPS design observations

1. Optimal connectivity is not necessary:
   - Most pairs of nodes never communicate directly.
2. Deterministic schemes have some advantages:
   - Analysis is easier
   - Efficient shared key discovery possible
3. Flexibility is attractive:
   - Competing requirements require flexibility
4. Compromise is desirable:
   - Moderation is more likely to be suitable than extremes

# Direct Application of Combinatorial Designs

Combinatorial designs are very natural objects to consider as candidate key rings for KPSs:

- Deterministic
- Rich and well understood structure
- Have long been associated with the building of KPSs

# Prioritising local connectivity

## Theorem

1. *Any block in a* $(v, b, r, k)$*-design* *meets* *at most* $k(r-1)$ *other blocks.*
2. *Every block meets* $k(r-1)$ *blocks precisely when the design has the property that any two blocks meet in at most one point.*

# Prioritising local connectivity

## Theorem

1. *Any block in a $(v, b, r, k)$-design meets at most $k(r-1)$ other blocks.*
2. *Every block meets $k(r-1)$ blocks precisely when the design has the property that any two blocks meet in at most one point.*

## Definition

*$(v, b, r, k)$-designs where any two blocks meet in at most one point are known as $(v, b, r, k)$-configurations.*

# Prioritising local connectivity

## Theorem

1. *Any block in a $(v, b, r, k)$-design meets at most $k(r-1)$ other blocks.*
2. *Every block meets $k(r-1)$ blocks precisely when the design has the property that any two blocks meet in at most one point.*

## Definition

*$(v, b, r, k)$-designs where any two blocks meet in at most one point are known as $(v, b, r, k)$-configurations.*

- KPSs based on configurations have optimal local connectivity.

# Prioritising local connectivity

## Theorem

1. *Any block in a $(v, b, r, k)$-design meets at most $k(r-1)$ other blocks.*
2. *Every block meets $k(r-1)$ blocks precisely when the design has the property that any two blocks meet in at most one point.*

## Definition

*$(v, b, r, k)$-designs where any two blocks meet in at most one point are known as $(v, b, r, k)$-configurations.*

- KPSs based on configurations have optimal local connectivity.
- However they offer no guarantees about resilience.

# Prioritising resilience

## Definition

*A* *w-key distribution pattern* *(KDP) is a set system* $(\mathcal{I}, \mathcal{B})$ *with* $|\mathcal{B}| = n$ *such that for any:*

1. *pair* $B_i, B_j \in \mathcal{B}$ *with:* $B_i \cap B_j \neq \emptyset$*; and*
2. $\{B_{l_1}, \ldots, B_{l_w}\} \subseteq \mathcal{B} \setminus \{B_i, B_j\}$*:*

$$B_i \cap B_j \not\subseteq (B_{l_1} \cup \cdots \cup B_{l_w}).$$

# Prioritising resilience

## Definition

*A w-key distribution pattern (KDP) is a set system* $(\mathcal{I}, \mathcal{B})$ *with* $|\mathcal{B}| = n$ *such that for any:*

1. *pair* $B_i, B_j \in \mathcal{B}$ *with:* $B_i \cap B_j \neq \emptyset$*; and*
2. $\{B_{l_1}, \ldots, B_{l_w}\} \subseteq \mathcal{B} \setminus \{B_i, B_j\}$*:*

$$B_i \cap B_j \not\subseteq (B_{l_1} \cup \cdots \cup B_{l_w}).$$

- KPSs based on KDPs offer optimal resilience if no more than $w$ nodes are compromised.

# Prioritising resilience

## Definition

*A* *w-key distribution pattern* *(KDP) is a set system* $(\mathcal{I}, \mathcal{B})$ *with* $|\mathcal{B}| = n$ *such that for any:*

1. *pair* $B_i, B_j \in \mathcal{B}$ *with:* $B_i \cap B_j \neq \emptyset$*; and*
2. $\{B_{l_1}, \ldots, B_{l_w}\} \subseteq \mathcal{B} \setminus \{B_i, B_j\}$*:*

$$B_i \cap B_j \not\subseteq (B_{l_1} \cup \cdots \cup B_{l_w}).$$

- KPSs based on KDPs offer optimal resilience if no more than $w$ nodes are compromised.
- A general KDP does not provide any guarantees of connectivity.

# Fully connected configurations

## Theorem

1. *A* $(v, b, r, k)$ *configuration is fully connected precisely when it is the dual of a* $2-(b, v, k, r, 1)$*-design.*
2. *When this happens,* $b \leq k(k-1)+1$.
3. *This bound is met by the projective planes.*

# Fully connected configurations

## Theorem

1. *A $(v, b, r, k)$ configuration is fully connected precisely when it is the dual of a $2-(b, v, k, r, 1)$-design.*
2. *When this happens, $b \leq k(k-1)+1$.*
3. *This bound is met by the projective planes.*

- Optimal connectivity
- Efficient shared key discovery
- Facilitating a very large number of nodes comes at the unattractive cost of relatively large key storage for each node (storage is approximately the square root of the number of nodes).

# Fully connected designs

Full connectivity places too many constraints:

- Lack of flexibility:
  - little room for tradeoff between important parameters

# Fully connected designs

Full connectivity places too many constraints:

- Lack of flexibility:
  - little room for tradeoff between important parameters
- Restrictions on number of nodes:
  - reasonable storage limitations lead to limited number of possible nodes.

# Fully connected designs

Full connectivity places too many constraints:

- Lack of flexibility:
  - little room for tradeoff between important parameters
- Restrictions on number of nodes:
  - reasonable storage limitations lead to limited number of possible nodes.
- Too much to the extreme:
  - full connectivity is more than we need
  - storage and resilience costs too high

# KPSs from Common Intersection Designs



**Definition**

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

# KPSs from Common Intersection Designs

## Definition

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

- CIDs do not have full connectivity
- Optimal CIDs have been constructed from:
  - generalised quadrangles
  - group-divisible designs
  - strongly-regular graphs

# KPSs from transversal designs

- A TD$(k, n)$ is a $(kn, n^2, n, k, k^2 - k)$-CID.
- For any prime $k \leq n$, a useful class of TD$k, n)$s can be constructed known as linear schemes.
- $k$ and $n$ can be varied to produce a range of compromises between the storage $k$, maximum number of nodes $n^2$, local connectivity $k(n + 1)$ and resilience

# KPSs from transversal designs

- A TD$(k, n)$ is a $(kn, n^2, n, k, k^2 - k)$-CID.
- For any prime $k \leq n$, a useful class of TD$k, n)$s can be constructed known as linear schemes.
- $k$ and $n$ can be varied to produce a range of compromises between the storage $k$, maximum number of nodes $n^2$, local connectivity $k(n+1)$ and resilience
- the local connectivity and resilience can be computed
- very efficient shared-key discovery phase

# Designs without full connectivity

Designs without full connectivity tend to provide:

- more room for flexibility between parameters
- more nodes for a given storage constraint

# Designs without full connectivity

Designs without full connectivity tend to provide:

- more room for flexibility between parameters
- more nodes for a given storage constraint

However...

- Less known about them
- Still a lack of flexibility
- Still a restriction on the number of nodes

# Designs as Building Blocks



- Combinatorial designs are very natural objects to use as components in the construction of a KPS.
- The resulting KPSs can hopefully be:
  - more flexible
  - inherit the advantages of designs

# Splitting a KPS

1. Start with an original KPS
2. Split nodes by associating each node in the original KPS with a set of nodes in a component KPS.
3. Form a new KPS consisting of:
   - several component KPSs
   - "bound" together by the original KPS.

# Splitting a KPS

1. Start with an original KPS
2. Split nodes by associating each node in the original KPS with a set of nodes in a component KPS.
3. Form a new KPS consisting of:
   - several component KPSs
   - "bound" together by the original KPS.

- the main gain is an increase in the number of nodes
- the main costs can be either in connectivity or resilience

# Extending a KPS

1. Start with an original KPS
2. Extend it to a new KPS by appending additional node allocations, which could be:
   - random
   - from another KPS

# Extending a KPS

1. Start with an original KPS
2. Extend it to a new KPS by appending additional node allocations, which could be:
   - random
   - from another KPS

As an example:

- KPSs from projective planes were extended by adding random subsets of blocks of the complementary design
- the main gain was more nodes and greater resilience
- the main loss was connectivity (although it was better than the random KPS)

# Packing a KPS

1. Start with an original KPS
2. Pack it into a new KPS by adding key identifiers to each node allocation.

# Packing a KPS

1. Start with an original KPS
2. Pack it into a new KPS by adding key identifiers to each node allocation.

As an example:

- the main gain is connectivity
- the main losses are storage and resilience

# Breaking a KPS

1. Start with an original KPS
2. Break it up to form a new KPS:
   - Contracting: removing key identifiers (throughout) the KPS
   - Block splitting: splitting node allocations.

# Breaking a KPS

1. Start with an original KPS
2. Break it up to form a new KPS:
   - Contracting: removing key identifiers (throughout) the KPS
   - Block splitting: splitting node allocations.

- the main gain is reduced storage and resilience
- the main losses are connectivity

# Joining KPSs

1. Start with an outer KPS based on set system $(\mathcal{I}, \mathcal{B}^{\text{out}})$
    - Node $U_j$ is associated with the block $B_j^{\text{out}}$
    - Key identifier $x_i$ defines a subset of nodes $N_i = \{U_j : x_i \in B_j^{\text{out}}\}$.

# Joining KPSs

1. Start with an outer KPS based on set system $(\mathcal{I}, \mathcal{B}^{\text{out}})$
   - Node $U_j$ is associated with the block $B_j^{\text{out}}$
   - Key identifier $x_i$ defines a subset of nodes $N_i = \{U_j : x_i \in B_j^{\text{out}}\}$.
2. Define an inner KPS on the nodes $N_i$
   - Node $U_j \in N_i$ receives the node allocation $B_j^{\text{in}-i}$

# Joining KPSs

1. Start with an outer KPS based on set system $(\mathcal{I}, \mathcal{B}^{\mathrm{out}})$
   - Node $U_j$ is associated with the block $B_j^{\mathrm{out}}$
   - Key identifier $x_i$ defines a subset of nodes $N_i = \{U_j : x_i \in B_j^{\mathrm{out}}\}$.
2. Define an inner KPS on the nodes $N_i$
   - Node $U_j \in N_i$ receives the node allocation $B_j^{\mathrm{in}-i}$
3. Node $U_j$ has total node allocation:

$$B_j = \cup_{x_i \in B_j^{\mathrm{out}}} B_j^{\mathrm{in}-i}.$$

# Joining KPSs

1. Start with an outer KPS based on set system $(\mathcal{I}, \mathcal{B}^{\text{out}})$
   - Node $U_j$ is associated with the block $B_j^{\text{out}}$
   - Key identifier $x_i$ defines a subset of nodes $N_i = \{U_j : x_i \in B_j^{\text{out}}\}$.
2. Define an inner KPS on the nodes $N_i$
   - Node $U_j \in N_i$ receives the node allocation $B_j^{\text{in}-i}$
3. Node $U_j$ has total node allocation:

$$B_j = \cup_{x_i \in B_j^{\text{out}}} B_j^{\text{in}-i}.$$

- KPSs not based on a set system can be used as inner KPSs
  - such as the Blom KPS

# Combinatorial engineering

## Definition

*Combinatorial engineering is the use of combinatorial objects as components, which are combined with other (not necessarily combinatorial) objects to build new structures.*

# Combinatorial engineering

## Definition

*Combinatorial engineering is the use of combinatorial objects as components, which are combined with other (not necessarily combinatorial) objects to build new structures.*

Cons:

- relatively unstudied from a mathematical perspective

# Combinatorial engineering

## Definition

*Combinatorial engineering is the use of combinatorial objects as components, which are combined with other (not necessarily combinatorial) objects to build new structures.*

Cons:

- relatively unstudied from a mathematical perspective
- desirable properties of components can be lost

# Combinatorial engineering

## Definition

*Combinatorial engineering is the use of combinatorial objects as components, which are combined with other (not necessarily combinatorial) objects to build new structures.*

Cons:

- relatively unstudied from a mathematical perspective
- desirable properties of components can be lost
- resulting constructions hard to analyse
  - often can only be studied through simulations

# Combinatorial engineering

## Definition

*Combinatorial engineering is the use of combinatorial objects as components, which are combined with other (not necessarily combinatorial) objects to build new structures.*

Cons:

- relatively unstudied from a mathematical perspective
- desirable properties of components can be lost
- resulting constructions hard to analyse
  - often can only be studied through simulations

Pros:

- some combinatorics can be better than no combinatorics
  - some structural guarantees can usually be preserved
  - can potentially gain the “best of all worlds”

# Combinatorial engineering

## Definition

*Combinatorial engineering is the use of combinatorial objects as components, which are combined with other (not necessarily combinatorial) objects to build new structures.*

Cons:

- relatively unstudied from a mathematical perspective
- desirable properties of components can be lost
- resulting constructions hard to analyse
  - often can only be studied through simulations

Pros:

- some combinatorics can be better than no combinatorics
  - some structural guarantees can usually be preserved
  - can potentially gain the "best of all worlds"
- highly flexible
  - KPSs for WSNs are not classical combinatorial objects

# KPSs for Special Networking Environments

Thus far we have discussed uncontrolled and homogeneous KPSs.

- Partially controlled KPSs
- Fully controlled KPSs
- Heterogenous KPSs

# Partially controlled KPSs

- One such scenario arises when nodes are deployed in groups where: nodes within one group are deployed **closer together on average** than nodes from different groups

# Partially controlled KPSs

- One such scenario arises when nodes are deployed in groups where: nodes within one group are deployed **closer together on average** than nodes from different groups
- Keys can be predistributed more efficiently (than for uncontrolled environments):
  - Can deploy a "relaxed" KPS on each group
  - Build in some means of inter-group communication
  - Balanced local connectivity desirable

# Partially controlled KPSs

- One such scenario arises when nodes are deployed in groups where: nodes within one group are deployed **closer together on average** than nodes from different groups
- Keys can be predistributed more efficiently (than for uncontrolled environments):
  - Can deploy a "relaxed" KPS on each group
  - Build in some means of inter-group communication
  - Balanced local connectivity desirable
- Such a KPS can be constructed from joining KPSs, where:
  - outer KPS is based on a resolvable transversal design:
  - two types of inner KPS, both based on Blom KPSs
  - parameters are flexible
  - connectivity and resilience can be traded off against storage
  - efficient shared key discovery inherited from components

# Fully controlled KPSs

- Precise knowledge of node location allows for very efficient KPSs

# Fully controlled KPSs

- Precise knowledge of node location allows for very efficient KPSs
- One option is to only assign shared keys to "neighbours"
  - For dense networks there are more efficient options

# Fully controlled KPSs

- Precise knowledge of node location allows for very efficient KPSs
- One option is to only assign shared keys to "neighbours"
  - For dense networks there are more efficient options
- If nodes are deployed in a highly structured physical formation then it is natural to look to combinatorial mathematics for building KPSs:
  - square grids
  - hexagonal grids
  - constructions use distinct difference configurations (which are related to designs)

# Heterogeneous KPSs

In heterogeneous networks not all the nodes have the same capabilities.

Two simple cases are:

1. Simple two-level hierarchies
2. Two-level hierarchies with a backbone

# Summary

- Combinatorial designs have been widely proposed for use as KPSs (for WSNs).

# Summary

- Combinatorial designs have been widely proposed for use as KPSs (for WSNs).
- Many direct applications of designs are too inflexible.

# Summary

- Combinatorial designs have been widely proposed for use as KPSs (for WSNs).
- Many direct applications of designs are too inflexible.
- Some specific designs offer a degree of flexibility:
  - More research on not fully connected designs is merited.

# Summary

- Combinatorial designs have been widely proposed for use as KPSs (for WSNs).
- Many direct applications of designs are too inflexible.
- Some specific designs offer a degree of flexibility:
  - More research on not fully connected designs is merited.
- Designs are potentially useful building blocks for KPSs:
  - More investigation of **meaningful** construction techniques required
  - Schemes offering good trade-offs are desirable
  - Ad hoc proposals do not necessarily add to the knowledge base

# Summary

- Combinatorial designs have been widely proposed for use as KPSs (for WSNs).
- Many direct applications of designs are too inflexible.
- Some specific designs offer a degree of flexibility:
  - More research on not fully connected designs is merited.
- Designs are potentially useful building blocks for KPSs:
  - More investigation of **meaningful** construction techniques required
  - Schemes offering good trade-offs are desirable
  - Ad hoc proposals do not necessarily add to the knowledge base
- Some interesting applications of combinatorial designs to special WSN environments.

# Back to the original question

## Question

*Are combinatorial designs appropriate tools to construct KPSs for WSNs?*

# Back to the original question

## Question

*Are combinatorial designs appropriate tools to construct KPSs for WSNs?*

## Answer

- *Yes, in some cases they do provide very interesting constructions.*
- *The most interesting KPSs do not necessarily come directly from designs.*
- *Designs can be used as components for interesting KPSs.*