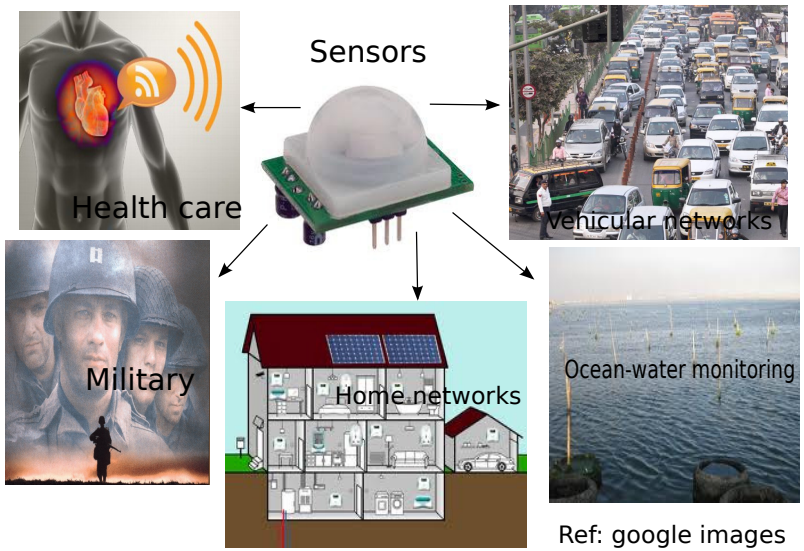
# Key Predistribution in Wireless Sensor Networks

Sushmita Ruj

R C Bose Center for Cryptology and Security
Indian Statistical Institute, Kolkata
E-mail : sush@isical.ac.in
Homepage: http://www.isical.ac.in/∼sush

# Wireless Sensor Networks




Ref: google images

- Tiny low-cost battery powered devices
- Resource constrained
- Often deployed in uninhabited regions and work unattended
- Sense temperature, humidity, smoke etc
- Data collected is processed by base station

# Operating under constraints

- Battery life
- Transmission range
- Bandwidth
- Memory contraints
- Computing constraints
- Prior deployment knowledge might not be known

# Security in wireless sensor networks

Security is important in several applications

- Military
- Health-care
- Commercial purposes

# Symmetric Vs Public key techniques

- Lightweight
- Low level of security
- Easy to implement
- Eg: Predistribution

- Computation intensive
- High degree of security
- Involved implementation
- ECC, RSA

# Why predistribute

- Security of the WSN hinges on the efficient key distribution techniques
- Even with the present day technology PKC and ECC are too computation intensive for WSNs
- Typically a WSN establishes a secure network by the use of predistributed keys

# Simple ways of predistribution

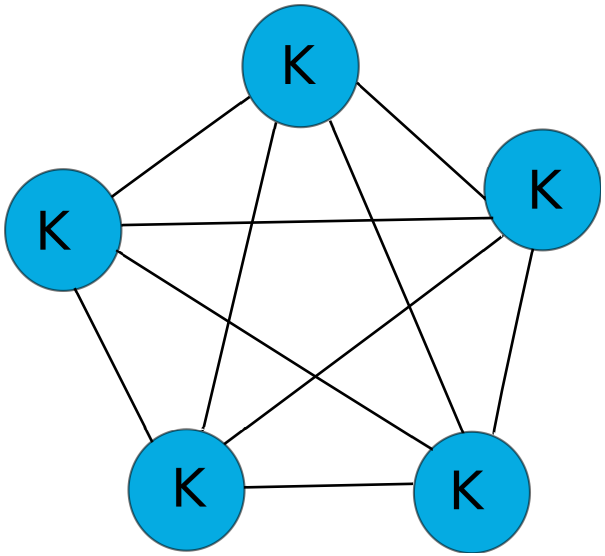



Figure : Master key distribution

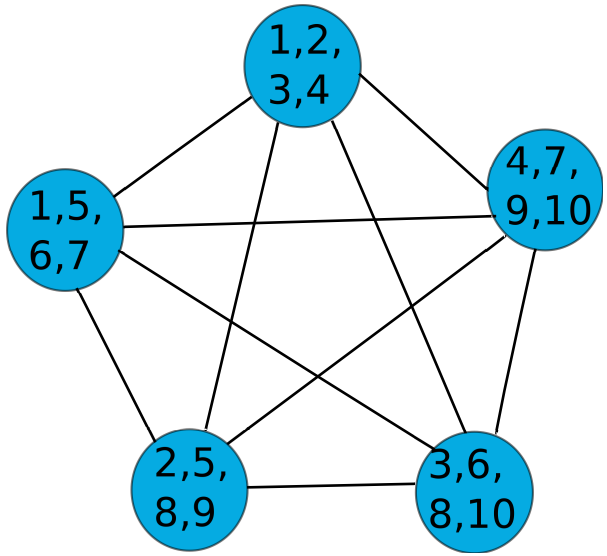



Figure : Pairwise key distribution

- Fully connected
- Minimal storage

- Fully connected
- High storage

# Simple ways of predistribution

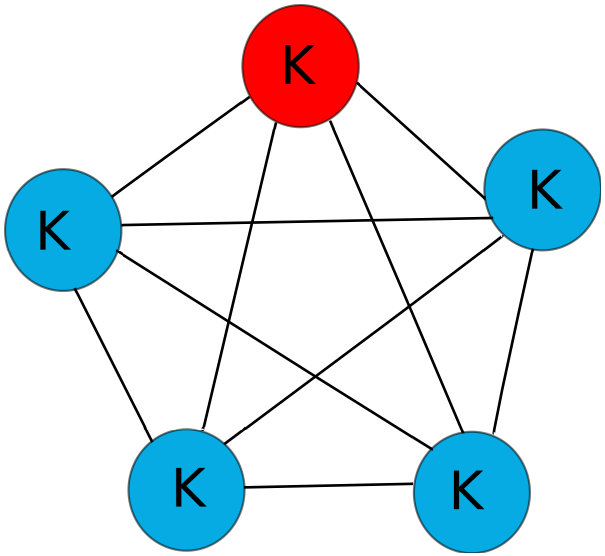



Figure : Master key distribution

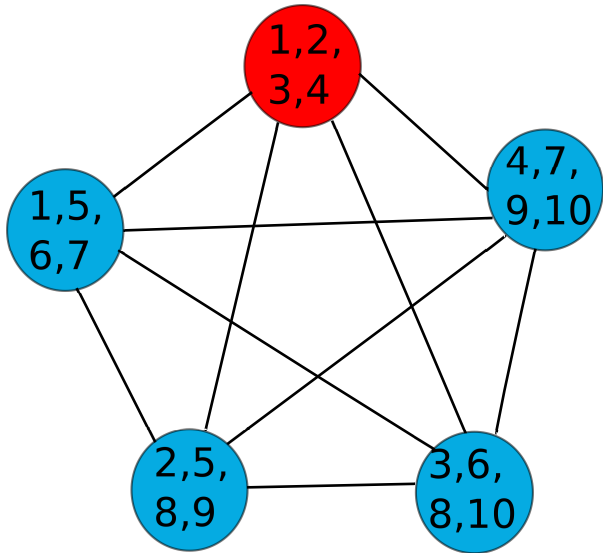



Figure : Pairwise key distribution

- Fully connected
- Minimal storage

- Fully connected
- High storage

# Simple ways of predistribution

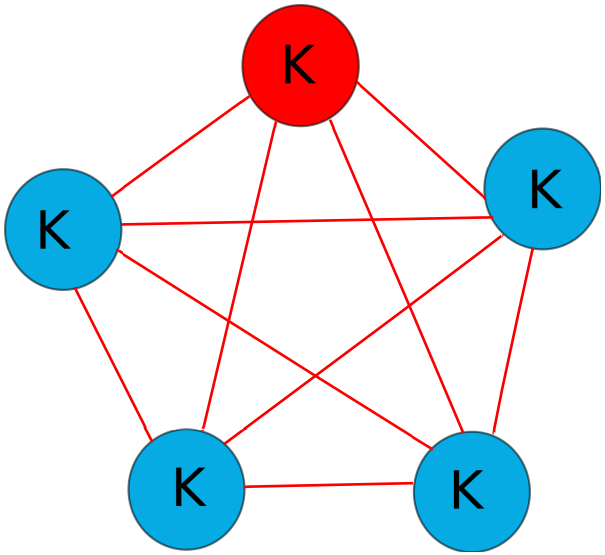



Figure : Master key distribution

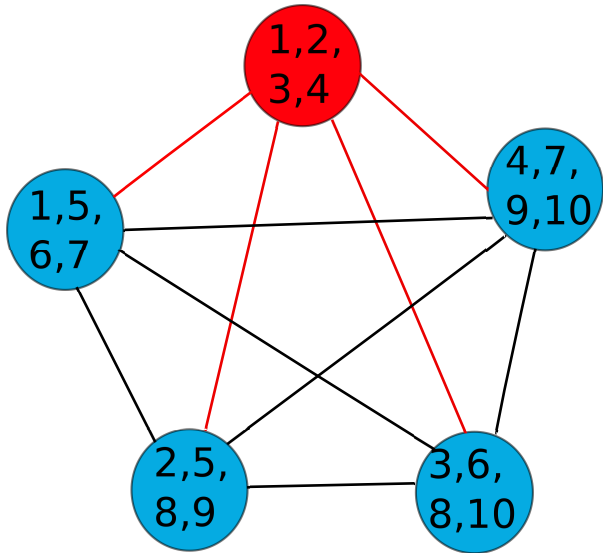



Figure : Pairwise key distribution

- Fully connected
- Minimal storage
- Low resilience

- Fully connected
- High storage
- High resilience

# Metrics used to evaluate key predistribution schemes

- Scalability: The distribution must allow post-deployment increase in the size of the network
- Efficiency
- Storage: Amount of memory required to store the keys
- Computation: Number of cycles needed for key establishment
- Communication : Number of messages exchanged during the key generation/ agreement phase
- Key Connectivity (probability of key share): The probability that two nodes share one/more keys should be high
- Average path length/diameter: Should be low, so that messages are communicated using few hops.
- Resiliency
- Revocation

# What is predistribution?

Distributing keys in nodes prior to deployment.

key-pool (key identifiers)

$$\{0, 1, 2, 3, 4, 5, 6\}.$$

# What is predistribution?

Distributing keys in nodes prior to deployment.

key-pool (key identifiers)

$$\{0, 1, 2, 3, 4, 5, 6\}.$$

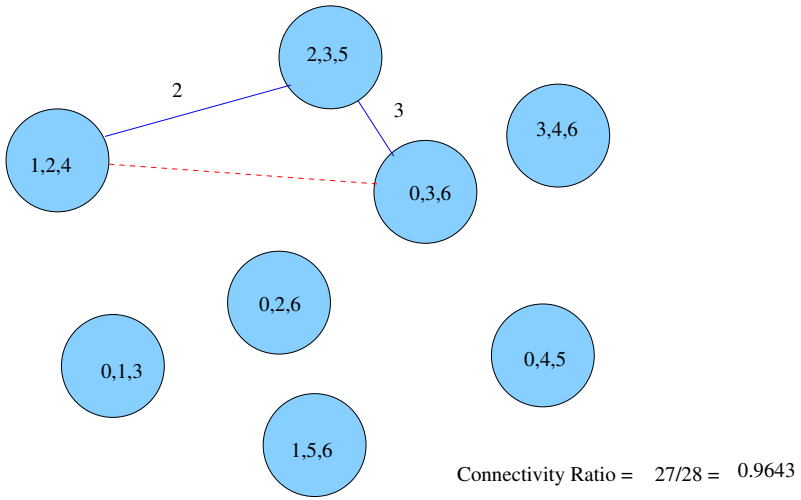



Figure : Nodes and link between them

Proposed by Eschenaur and Gligor, ACMCCS 2002

# Resilience

- Resilience means that even if a number of nodes are compromised, i.e., the keys contained therein are revealed, the complete network should not fail, i.e., only a part of the network should be affected, if at all.

# Resilience

- Resilience means that even if a number of nodes are compromised, i.e., the keys contained therein are revealed, the complete network should not fail, i.e., only a part of the network should be affected, if at all.
- How to define resilience?
- How to find compromised nodes?
- This will not be discussed here.

# Key Graphs

- Construct a Key graph $G(V, E)$, with $|V| = n$ vertices (one each for $n$ nodes)
- There exists an edge between two nodes $u, v \in V$, such that the corresponding vertices share atleast one common key and are within communication range.
- Node attacks and edge attacks
- Define fault tolerant parameters
- Vertex cut, edge cut, etc
- Conductance, expansion, integrity, toughness, tenacity

# Key Graphs

- Construct a Key graph $G(V, E)$, with $|V| = n$ vertices (one each for $n$ nodes)
- There exists an edge between two nodes $u, v \in V$, such that the corresponding vertices share atleast one common key and are within communication range.
- Node attacks and edge attacks
- Define fault tolerant parameters
- Vertex cut, edge cut, etc
- Conductance, expansion, integrity, toughness, tenacity
- Here we consider only vertex attacks

# Fault tolerant measures of key graphs

Resilience in WSN is measured by the following parameters parameters

1. $V(s)$ which is the probability of a node (which is not among the compromised nodes) being isolated when $s$ nodes are compromised.
2. $E(s)$, which is defined as

$$E(s) = \frac{\text{Number of links exposed after } s \text{ nodes are compromised}}{\text{Number of links present before compromise}}$$

# Fault tolerant measures of key graphs

Resilience in WSN is measured by the following parameters parameters

1. $V(s)$ which is the probability of a node (which is not among the compromised nodes) being isolated when $s$ nodes are compromised.
2. $E(s)$, which is defined as
$$E(s) = \frac{\text{Number of links exposed after } s \text{ nodes are compromised}}{\text{Number of links present before compromise}}$$
3. Number of isolated vertices when $s$ nodes are removed
4. Number of components when $s$ nodes are removed
5. Finding vertex cut and edge cut
6. Size of the giant component, if any
7. Giant component is the component which contains a constant fraction of all the nodes in the network

# Connectivity of basic predistribution scheme

- Let there be $K$ keys in the key pool and $k$ be the key chain
- The event that to nodes do not share any key arises when $k$ keys in the second node are disjoint from the set of $k$ keys in the first node
- The probability that this event happens is

$$\frac{\binom{K-k}{k}}{\binom{K}{k}} = \frac{((K-k)!)^2}{(K-2k)!K!}$$

- Probability that two nodes share a common key is given by

$$p = 1 - Pr[\text{two nodes do not share any key}] = 1 - \frac{((K-k)!)^2}{(K-2k)!K!}$$

- Thus, with probability $p$ there exists an edge between two vertices.
- We can think of the key graph as $G(n, p)$, where $n$ is the number of nodes
- For example if $K = 10,000$ and $k = 75$ then $p = 0.5$.

# Redoutable and Unsplittable Properties

- Random key predistribution graphs possess Redoutable and Unsplittable Properties
- A network is redoubtable if the adversary needs to capture a large number of nodes to compromise the confidentiality of the network
- If any adversary that captures sensors at random with the aim of compromising a constant fraction of the links must capture at least a constant fraction of the nodes.

**Theorem**

If $K \geq n \log n$, and $k = O(\sqrt{\log n})$, then the network is not only connected with high probability but is also reboudtable.

# Redoutable and Unsplittable Properties

- A network is unsplittable, if a random adversary cannot partition the network into two large chunks, that is, two linear size sets of vertices, and compromise all links between them, unless it captures a linear fraction of the nodes.

**Theorem**

Let $G$ be a key graph (full-visibility case) such that

$$\frac{k^2}{n} \geq c\frac{\log n}{n},$$

with $c \geq 17$ an $K \geq n \log n$. Then, with probability $1 - o(1)$, the graph is at the same time unsplittable, connected, and redoubtable. The term o(1) goes to zero as n goes to infinity.

- Di Pietro et. al ACM TISSEC, 2008.

# Disadvantages of using Random Key Predistribution: Key establishment

- How to find the common key/keys shared by two nodes A and B

# Disadvantages of using Random Key Predistribution: Key establishment

- How to find the common key/keys shared by two nodes A and B
- Node A sends all its key identifiers
- Node B compares with its set of key identifiers
- If there is a match, use the key for communication
- Computation and Communication overheads? ($k$ keys in each nodes)

# Disadvantages of using Random Key Predistribution: Key establishment

- How to find the common key/keys shared by two nodes A and B
- Node A sends all its key identifiers
- Node B compares with its set of key identifiers
- If there is a match, use the key for communication
- Computation and Communication overheads? ($k$ keys in each nodes)
- Computation cost: $O(k^2)$

# Disadvantages of using Random Key Predistribution: Key establishment

- How to find the common key/keys shared by two nodes A and B
- Node A sends all its key identifiers
- Node B compares with its set of key identifiers
- If there is a match, use the key for communication
- Computation and Communication overheads? ($k$ keys in each nodes)
- Computation cost: $O(k^2)$
- Computation cost: $O(k \log k)$: Sort keys and then match.

# Disadvantages of using Random Key Predistribution: Key establishment

- How to find the common key/keys shared by two nodes A and B
- Node A sends all its key identifiers
- Node B compares with its set of key identifiers
- If there is a match, use the key for communication
- Computation and Communication overheads? ($k$ keys in each nodes)
- Computation cost: $O(k^2)$
- Computation cost: $O(k \log k)$: Sort keys and then match.
- Communication cost: $O(k \log K)$, keys are $K$ bits long

# Anything better?

- Challenge response protocol
- A sends $(\alpha_i, E_{k_i}(\alpha_i))$, for each $k_i$ in the key chain
- B encrypts each $\alpha_i$ with a key it has
- If it matches with $E_{k_i}(\alpha_i)$, then $k_i$ is the common key
- Computation cost: $O(kE)$, time for encryption using $k_i$

# Anything better?

- Challenge response protocol
- A sends $(\alpha_i, E_{k_i}(\alpha_i))$, for each $k_i$ in the key chain
- B encrypts each $\alpha_i$ with a key it has
- If it matches with $E_{k_i}(\alpha_i)$, then $k_i$ is the common key
- Computation cost: $O(kE)$, time for encryption using $k_i$
- Communication cost: $O(k \log K)$, keys are $K$ bits long

# Anything better?

- Challenge response protocol
- A sends $(\alpha_i, E_{k_i}(\alpha_i))$, for each $k_i$ in the key chain
- B encrypts each $\alpha_i$ with a key it has
- If it matches with $E_{k_i}(\alpha_i)$, then $k_i$ is the common key
- Computation cost: $O(kE)$, time for encryption using $k_i$
- Communication cost: $O(k \log K)$, keys are $K$ bits long
- Communication and computation costs are still high.
- This motivates us to use deterministic designs.

# Deterministic schemes

- Deterministic way of selecting keys from key pool or constructing common keys
- Easy to find common keys between two node
- Only node id needs to be known to find common keys
- Limited scalability
- Two broad classes
  - Polynomial based
  - Combinatorial design based

# Combinatorial basis of the problems

- A set system or design is a pair $(X, \mathcal{A})$, where $\mathcal{A}$ is a set of subsets of $X$, called blocks.

# Combinatorial basis of the problems

- A set system or design is a pair $(X, \mathcal{A})$, where $\mathcal{A}$ is a set of subsets of $X$, called blocks.
- $X = \{0, 1, 2, 3, 4, 5\}$
- $\mathcal{A} = \{\{0, 1, 2\}, \{0, 1, 3\}, \{0, 2, 4\}, \{0, 3, 5\}, \{0, 4, 5\}, \{1, 2, 5\}, \{1, 3, 4\}, \{1, 4, 5\}, \{2, 3, 4\}, \{2, 3, 5\}\}$

# Combinatorial basis of the problems

- A set system or design is a pair $(X, \mathcal{A})$, where $\mathcal{A}$ is a set of subsets of $X$, called blocks.
- $X = \{0, 1, 2, 3, 4, 5\}$
- $\mathcal{A} = \{\{0, 1, 2\}, \{0, 1, 3\}, \{0, 2, 4\}, \{0, 3, 5\}, \{0, 4, 5\}, \{1, 2, 5\}, \{1, 3, 4\}, \{1, 4, 5\}, \{2, 3, 4\}, \{2, 3, 5\}\}$
- A $BIBD(v, b, r, k, \lambda)$, is a design which satisfies the following conditions:
  1. $|X| = v$, $|\mathcal{A}| = b$,
  2. Each subset in $\mathcal{A}$ contains exactly $k$ elements,
  3. Each element in $X$ occurs in $r$ many blocks,
  4. Each pair of elements in $X$ is contained in exactly $\lambda$ blocks in $\mathcal{A}$.
- The above design is a $(6, 10, 5, 3, 2)$-design

# Combinatorial basis of the problems

- Take the Dual of the design, interchange between keys are nodes.

# Combinatorial basis of the problems

- Take the Dual of the design, interchange between keys are nodes.
- $X' = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- $\mathcal{A}' = \{\{0, 1, 2, 3, 4\}, \{0, 1, 5, 6, 7\}, \{0, 2, 5, 8, 9\}, \{1, 3, 6, 8, 9\}, \{2, 4, 6, 7, 8\}, \{3, 4, 5, 6, 9\}$

# Combinatorial basis of the problems

- Take the Dual of the design, interchange between keys are nodes.
- $X' = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- $\mathcal{A}' = \{\{0, 1, 2, 3, 4\}, \{0, 1, 5, 6, 7\}, \{0, 2, 5, 8, 9\}, \{1, 3, 6, 8, 9\}, \{2, 4, 6, 7, 8\}, \{3, 4, 5, 6, 9\}$
- A $Dual - BIBD(v, b, r, k, \lambda)$, is a design which satisfies the following conditions:
  1. $|X'| = b$, $|\mathcal{A}'| = v$,
  2. Each subset in $\mathcal{A}'$ contains exactly $r$ elements,
  3. Each element in $X'$ occurs in $k$ many blocks,
  4. Each pair of blocks have exactly $\lambda$ elements in common.
- The above design is a $(10, 6, 3, 5)$-design

# Combinatorial basis of the problems

- Take the Dual of the design, interchange between keys and nodes.

# Combinatorial basis of the problems

- Take the Dual of the design, interchange between keys and nodes.
- $X' = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ Key pool
- $\mathcal{A}' = \{\{0, 1, 2, 3, 4\}, \{0, 1, 5, 6, 7\}, \{0, 2, 5, 8, 9\}, \{1, 3, 6, 8, 9\}, \{2, 4, 6, 7, 8\}, \{3, 4, 5, 6, 9\}$ Sensors

# Combinatorial basis of the problems

- Take the Dual of the design, interchange between keys and nodes.
- $X' = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ Key pool
- $\mathcal{A}' = \{\{0, 1, 2, 3, 4\}, \{0, 1, 5, 6, 7\}, \{0, 2, 5, 8, 9\}, \{1, 3, 6, 8, 9\}, \{2, 4, 6, 7, 8\}, \{3, 4, 5, 6, 9\}$ Sensors
- A $Dual - BIBD(v, b, r, k, \lambda)$, is a design which satisfies the following conditions:
  1. $|X'| = b$, $|\mathcal{A}'| = v$, $b$: size of key pool, $v$: # sensors
  2. Each subset in $\mathcal{A}'$ contains exactly $r$ elements, $r$: size of key chain
  3. Each variety in $X'$ occurs in $k$ many blocks,
  4. Each pair of blocks have exactly $\lambda$ elements in common.
  5. Two sensors share atleast $\lambda$ keys
- The above design is a $(10, 6, 3, 5)$-design

# BIBD and Sensor nodes

Correspondence between a dual of $\lambda - (v, b, r, k)$ design and sensor network.

- $b =$ key-pool size
- $v =$ number of sensor nodes
- $k =$ number of nodes in which a given key occurs
- $r =$ number of keys in a node ( key-chain length)
- $\lambda =$ number of common keys between two nodes

  Consider dual of 2-(6,10,5,3) :

  $$\text{Key-pool} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

  $n_1 : \{0, 1, 2, 3, 4, 5\}$ $\quad$ $n_4 : \{1, 3, 6, 8, 9\}$

  $n_2 : \{0, 1, 5, 6, 7\}$ $\quad$ $n_5 : \{2, 4, 6, 7, 8\}$

  $n_3 : \{0, 2, 5, 8, 9\}$ $\quad$ $n_6 : \{3, 4, 5, 6, 9\}$

# Key predistribution using PBIBD: Ruj-Roy2007

| | | | | |
|---|---|---|---|---|
| * | 1 | 2 | 3 | 4 |
| 1 | * | 5 | 6 | 7 |
| 2 | 5 | * | 8 | 9 |
| 3 | 6 | 8 | * | 10 |
| 4 | 7 | 9 | 10 | * |

A 2 - associate class PBIBD.
Firsts Associates : Elements belonging to the same row or column
Second Associates: Rest of the elements
First associate of 1 : 2, 3, 4, 5, 6, 7. Second associate of 1: 8, 9, 10

Block 1: (2, 3, 4, 5, 6, 7) Block 2: (1, 3, 4, 5, 8, 9)
Block 3: (1, 2, 4, 6, 8, 10) Block 4: (1, 2, 3, 7, 9, 10)
Block 5: (1, 2, 6, 7, 8, 9) Block 6: (1, 3, 5, 7, 8, 10)
Block 7: (1, 4, 5, 6, 9, 10) Block 8: (2, 3, 5, 6, 9, 10)
Block 9: (2, 4, 5, 7, 8, 10) Block 10: (3, 4, 6, 7, 8, 9)

$v = b = 10, r = k = 6, \lambda_1 = 3, \lambda_2 = 4$

# PBIBD example

| | | | | |
|---|---|---|---|---|
| * | 1 | 2 | 3 | 4 |
| 1 | * | 5 | 6 | 7 |
| 2 | 5 | * | 8 | 9 |
| 3 | 6 | 8 | * | 10 |
| 4 | 7 | 9 | 10 | * |

Set of keys identifiers $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

Node 1: (2, 3, 4, 5, 6, 7)
Node 2: (1, 3, 4, 5, 8, 9)
Node 3: (1, 2, 4, 6, 8, 10)
Node 4: (1, 2, 3, 7, 9, 10)
Node 5: (1, 2, 6, 7, 8, 9)
Node 6: (1, 3, 5, 7, 8, 10)
Node 7: (1, 4, 5, 6, 9, 10)
Node 8: (2, 3, 5, 6, 9, 10)
Node 9: (2, 4, 5, 7, 8, 10)
Node 10: (3, 4, 6, 7, 8, 9)

# Combinatorial Design based solutions

- Mitchell and Piper (1988) Laid the foundations. Not applied to WSN

# Combinatorial Design based solutions

- Mitchell and Piper (1988) Laid the foundations. Not applied to WSN
- Camtepe and Yener (2004) : Generalized Quadrangles
- Lee and Stinson (2005) : Transversal Designs
- Chakrabarty, Maitra and Roy (2006) : Merging Blocks
- Ruj and Roy (2007) : Partially Balanced Incomplete Block Designs (PBIBDs)
- Ruj and Roy (2008) : Reed-Solomon codes
- Blackburn, Etzion, Martin and Paterson (2008) : Costas Arrays
- Ruj and Roy ( ACM TOSN 2009) : Deterministic grid-group deployment
- Martin, Paterson, Stinson (2010) : Group Deployment
- Ruj, Nayak, Stojmenovic (INFOCOM 2011 and IEEE TC): Triple Key Distribution, Combinatorial Trades
- Ruj, Sakurai (Globecom 2013): Hybrid Key Distribution,

# Modified Camtape-Yener Scheme

- Nodes are indexed by $(a, b, c)$ where $a, b, c \in GF(q)$.
- Nodes are given by the identifiers $(1, b, c)$, $(0, 1, c)$ and $(0, 0, 1)$, where $b, c \in GF(q)$.
- So there are a total of $q^2 + q + 1$ nodes.
- The identifiers of the keys are indexed by $(x, y, z)$ where $x, y, z, \in GF(q)$.
- Identifiers of the keys are given by $(x, y, 1)$, $(x, 1, 0)$ and $(1, 0, 0)$, where $x, y \in GF(q)$.
- So there are a total of $q^2 + q + 1$ keys ( or elements).
- A key $(x, y, z)$ is assigned to node $(a, b, c)$ if $ax + by + cz = 0$.
- Number of keys in each node is $q + 1$.

# Example

$q = 3$ Number of nodes $= 13$ Keys per node $= 4$

| Node | Key identifiers |
|---|---|
| 1(1,0,0) | (0,0,1), (0,1,1), (0,2,1), (0,1,0) |
| 2(1, 0, 1) | (2,0,1), (2,1,1), (2,2,1), (0,1,0) |
| 3(1, 0, 2) | (1,0,1), (1,1,1), (1,2,1), (0,1,0) |
| 4(1, 1, 0) | (0,0,1), (1,2,1), (2,1,1), (2,1,0) |
| 5(1, 1, 1) | (0,2,1), (1,1,1), (2,0,1), (2,1,0) |
| 6(1, 1, 2) | (0,1,1), (1,0,1), (2,2,1), (2,1,0) |
| 7(1, 2, 0) | (0,0,1), (1,1,1), (2,2,1), (1,1,0) |
| 8(1, 2, 1) | (0,1,1), (1,2,1), (2,0,1), (1,1,0) |
| 9(1, 2, 2) | (0,2,1), (1,0,1), (2,1,1), (1,1,0) |
| 10(0, 1, 0) | (0,0,1), (1,0,1), (2,0,1), (1,0,0) |
| 11(0, 1, 1) | (0,2,1), (1,2,1), (2,2,1), (1,0,0) |
| 12(0, 1, 2) | (0,1,1), (1,1,1), (2,1,1), (1,0,0) |
| 13(0, 0, 1) | (0,1,0), (2,1,0), (1,1,0), (1,0,0) |

# Key establishment scheme for deterministic designs

- Only the node identity $i$ is to be broadcasted
- Given a node identifier $i$ node $j$ can calculate the common key between the nodes, if there is one, using some simple algorithm, which is embedded in the node.
- None of the keys or the identifiers are revealed

# Key establishment scheme for deterministic designs

- The pattern in the the deterministic designs is exploited in designing key establishment schemes
- Communication overhead $= O(\log N)$ , $N$ is the number of nodes
- Computation complexity $= O(1)$

# Overheads

- The identifier of the node identified by the tuple $(a, b, c)$ must be broadcasted
- Communication overhead : $O(\log q)$ where $N = q^2 + q + 1$ is the number of nodes
- Computation overhead $= O(1)$

# Example

| Node | Key identifiers |
|---|---|
| 1(1,0,0) | (0,0,1), (0,1,1), (0,2,1), (0,1,0) |
| 2(1, 0, 1) | (2,0,1), (2,1,1), (2,2,1), (0,1,0) |
| 3(1, 0, 2) | (1,0,1), (1,1,1), (1,2,1), (0,1,0) |
| 4(1, 1, 0) | (0,0,1), (1,2,1), (2,1,1), (2,1,0) |
| 5(1, 1, 1) | (0,2,1), (1,1,1), (2,0,1), (2,1,0) |
| 6(1, 1, 2) | (0,1,1), (1,0,1), (2,2,1), (2,1,0) |
| 7(1, 2, 0) | (0,0,1), (1,1,1), (2,2,1), (1,1,0) |
| 8(1, 2, 1) | (0,1,1), (1,2,1), (2,0,1), (1,1,0) |
| 9(1, 2, 2) | (0,2,1), (1,0,1), (2,1,1), (1,1,0) |
| 10(0, 1, 0) | (0,0,1), (1,0,1), (2,0,1), (1,0,0) |
| 11(0, 1, 1) | (0,2,1), (1,2,1), (2,2,1), (1,0,0) |
| 12(0, 1, 2) | (0,1,1), (1,1,1), (2,1,1), (1,0,0) |
| 13(0, 0, 1) | (0,1,0), (2,1,0), (1,1,0), (1,0,0) |

- Node 13 shares key identifier (1, 0, 0) with node 10, 11, 12.
- Node 13 shares $(-b_j, 1, 0) = (2, 1, 0)$ with node 4.
- Node 10 shares $(b_j c_i - c_j, -c_i, 1) = (1, 0, 1)$ with node 6
- Node 2 shares $(-c_1 + b_1 \frac{c_1 - c_2}{b_1 - b_2}, \frac{c_2 - c_1}{b_1 - b_2}, 1) = (2, 2, 1)$ with node 7

# Example $V(s)$

| Node | Key identifiers |
|---|---|
| 1(1,0,0) | (0,0,1), (0,1,1), (0,2,1), (0,1,0) |
| 2(1, 0, 1) | (2,0,1), (2,1,1), (2,2,1), (0,1,0) |
| 3(1, 0, 2) | (1,0,1), (1,1,1), (1,2,1), (0,1,0) |
| 4(1, 1, 0) | (0,0,1), (1,2,1), (2,1,1), (2,1,0) |
| 5(1, 1, 1) | (0,2,1), (1,1,1), (2,0,1), (2,1,0) |
| 6(1, 1, 2) | (0,1,1), (1,0,1), (2,2,1), (2,1,0) |
| 7(1, 2, 0) | (0,0,1), (1,1,1), (2,2,1), (1,1,0) |
| 8(1, 2, 1) | (0,1,1), (1,2,1), (2,0,1), (1,1,0) |
| 9(1, 2, 2) | (0,2,1), (1,0,1), (2,1,1), (1,1,0) |
| 10(0, 1, 0) | (0,0,1), (1,0,1), (2,0,1), (1,0,0) |
| 11(0, 1, 1) | (0,2,1), (1,2,1), (2,2,1), (1,0,0) |
| 12(0, 1, 2) | (0,1,1), (1,1,1), (2,1,1), (1,0,0) |
| 13(0, 0, 1) | (0,1,0), (2,1,0), (1,1,0), (1,0,0) |

Suppose nodes 1,5, 6 and 13 are compromised. Then the nodes 7 and 10 are disconnected.

$V(s) = 2/13 = 0.1538$

# Example $E(s)$

| Node | Key identifiers |
|---|---|
| 1(1,0,0) | (0,0,1), (0,1,1), (0,2,1), (0,1,0) |
| 2(1, 0, 1) | (2,0,1), (2,1,1), (2,2,1), (0,1,0) |
| 3(1, 0, 2) | (1,0,1), (1,1,1), (1,2,1), (0,1,0) |
| 4(1, 1, 0) | (0,0,1), (1,2,1), (2,1,1), (2,1,0) |
| 5(1, 1, 1) | (0,2,1), (1,1,1), (2,0,1), (2,1,0) |
| 6(1, 1, 2) | (0,1,1), (1,0,1), (2,2,1), (2,1,0) |
| 7(1, 2, 0) | (0,0,1), (1,1,1), (2,2,1), (1,1,0) |
| 8(1, 2, 1) | (0,1,1), (1,2,1), (2,0,1), (1,1,0) |
| 9(1, 2, 2) | (0,2,1), (1,0,1), (2,1,1), (1,1,0) |
| 10(0, 1, 0) | (0,0,1), (1,0,1), (2,0,1), (1,0,0) |
| 11(0, 1, 1) | (0,2,1), (1,2,1), (2,2,1), (1,0,0) |
| 12(0, 1, 2) | (0,1,1), (1,1,1), (2,1,1), (1,0,0) |
| 13(0, 0, 1) | (0,1,0), (2,1,0), (1,1,0), (1,0,0) |

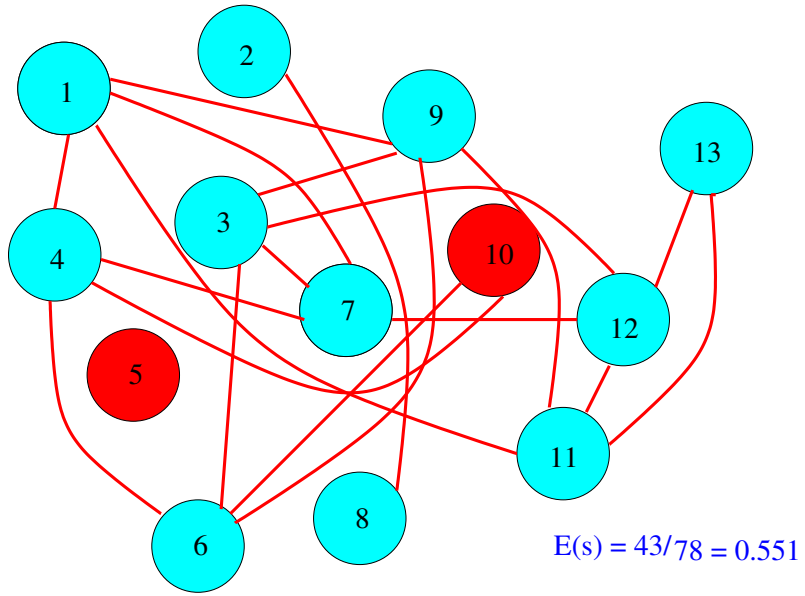
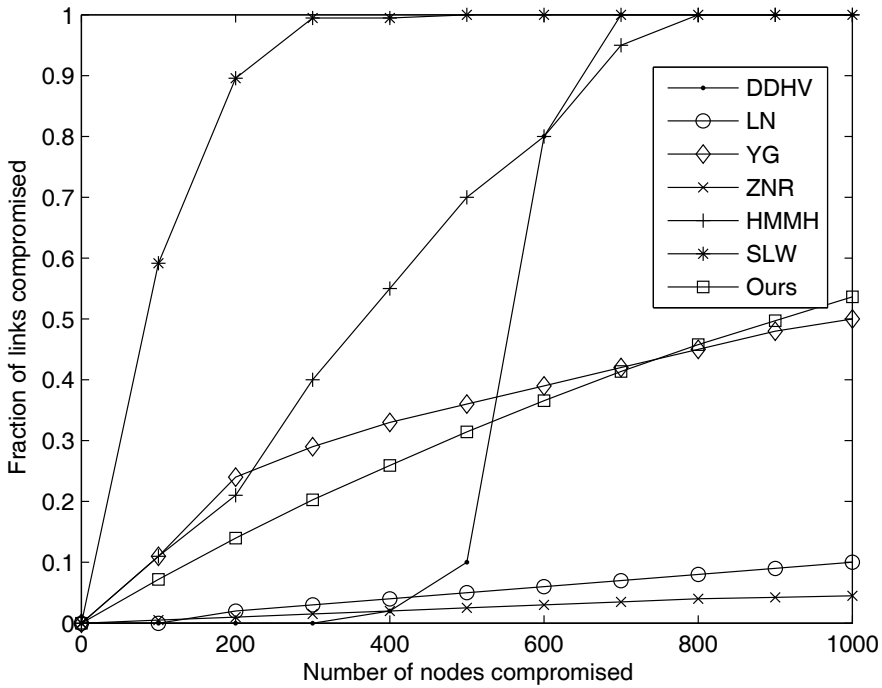



Figure : $E(s)$ when nodes 5 and 10 are compromised

# Comparison of our work (TOSN 2009) with other grid-group deployment




Although LN and SLW have better resilience, two nodes cannot communicate directly. Thus they have higher communication cost than ours. Number of nodes in the network is 10000.

# Comparison of our work (IEEE TC 2013) with previous schemes

| Scheme | Storage | Resilience | Computation | Communication | Mobile Network |
|---|---|---|---|---|---|
| Naive | $N-1$ | Fully resilient | $O(1)$ | $O(\log N)$ | Yes |
| Chan et al. (2003) | $O(k \log N)$ | Fully resilient | $O(1)$ | $O(\log N)$ | No |
| Blom | $(c+1)$ | $c$-secure | $O(c)$ | $O(\log N)$ | Yes |
| Blundo et al. (1998) | $(c+1)$ | $c$-secure | $O(c)$ | $O(\log N)$ | Yes |
| Du et al. (2003) | $\omega(c+1)$ | $c$-secure | $O(c)$ | $O(\log N)$ | Yes |
| Zu et al. (2005) | $k$ | Depends on $s$ | $O(n)$ | $O(\log N)$ | Yes |
| PIKE | (2005)$\sqrt{N}+1$ | Depends on $s$ | $O(1)$ | $O(\log N)$ | No |
| Traynor (2006) | $k$ | Depends on $s$ | $O(k \log k)$ | $k$ | No |
| Huang-Medhi (2007) | $k$ | Depends on $s$ | $O(k \log k)$ | $k$ | No |
| Ruj et al (2011) | $O(\sqrt{N})$ | High resilience | $O((\log N)^2)$ | $O(\log N)$ | Yes |

Table : Comparison of different pairwise key establishment schemes. $s$ is the number of nodes compromised

$s$ is the number of compromised nodes

# End

Thank You