M.Tech. (Computer Science) Dissertation

Pairwise Key Establishment in Wireless Sensor Networks

A dissertation submitted in partial fulfillment of the requirements
for the award of M.Tech.(Computer Science) degree

By

Vasala Ravikishore

Roll No: CS0809

under the supervision of

Professor Rana Barua,

Stat-Math Unit

Wireless Sensor Networks

July 15, 2010

Contents

1 Introduction to WSN **3**
- 1.1 Wireless Sensor Network 3
- 1.2 Design Challenges 4
- 1.3 Security Issues in Sensor Networks 5

2 Background **6**
- 2.1 Pairwise Key Establishment 6
  - 2.1.1 Probabilistic Key Pre-Distribution 6
  - 2.1.2 Polynomial-Based Key Pre-Distribution 6
  - 2.1.3 Polynomial Pool-Based Key Pre-Distribution 7
  - 2.1.4 Random Subset Assignment Key Pre-Distribution 8
  - 2.1.5 Grid-Based Key Pre-Distribution 9

3 Grid-Based Key Pre-Distribution Using Multivariate Symmetric Polynomial **13**
- 3.1 Two Dimensional Grid-Based Scheme Using 3-Variate Symmetric Polynomial 13
  - 3.1.1 Polynomial Share Pre-Distribution 14
  - 3.1.2 Key Establishment Mechanism 14
- 3.2 Multi Dimensional(n) Grid-Based (Hyper-Cube) Scheme Using (n+1)-Variate Symmetric Polynomial 15
  - 3.2.1 Polynomial Share Pre-Distribution 15
  - 3.2.2 Key Establishment Mechanism 15

4 Conclusion **20**

Chapter 1

Introduction to WSN

1.1 Wireless Sensor Network

Wireless sensor networks have recently emerged as an important means to study and interact with the physical world. A sensor network typically consists of a large number of tiny sensor nodes and possibly a few powerful control nodes(also called *base stations*). Every sensor node has one or a few sensing components to sense conditions (e.g. temperature, humidity, pressure) from its immediate surroundings and a processing and communication component to carry out simple computation on the raw data and communicate with its neighbor nodes. Sensor nodes are usually densely deployed in a large scale and communicate with each other in short distances via wireless links. The control nodes may further process the data collected from the sensor nodes, disseminate control commands to the sensor nodes, and connect the network to a traditional wired network.

Sensor nodes are usually scattered randomly in the field and will form a sensor network after deployment in an ad hoc manner to fulfill certain tasks. There is usually no infrastructure support for sensor networks. As one example, let us look at the battle field surveillance. In this application, a large number of small sensor nodes are rapidly deployed in a battlefield via airplanes or trucks. After deployment, these sensor nodes are quickly self-organized together to form an ad-hoc network. Each individual sensor node then monitors conditions and activities in its local surroundings and reports its observations to a central server by communicating with its neighbors. Collecting these observations from sensor nodes allows us to conduct accurate detections on the activities (e.g., possible attacks) of the opposing force and make appropriate decisions and responses in the battlefield.

Obviously, the design of sensor networks requires wireless networking techniques, especially wireless ad hoc networking techniques. However, most traditional wireless networking protocols and algorithms are not suitable for sensor networks. One main challenge of designing a sensor network comes from the resource constraints on sensor nodes.

The wide applications of wireless sensor networks and the challenges in designing such networks have attracted many researchers to develop protocols and algorithms for sensor networks. Note that sensor networks may be deployed in hostile environments where enemies may be present. Security becomes a critical

issue to make sure the correct operation of sensor networks in many security sensitive scenarios such as military tasks.

1.2 Design Challenges

Security becomes one of the major concerns when there are potential attacks against sensor networks. many protocols and algorithms (e.g, routing, localization) will not work in hostile environments without security protection. Security services such as authentication and key management are critical to ensure the normal operations of a sensor network in hostile environments. However, some special features of sensor networks make it particularly challenging to provide these security services for sensor networks.

- *Resource constraints* : Sensor nodes are usually resource constrained, especially energy constrained. Every operation reduces the lifetime of a sensor node. This makes it undesirable to perform expensive operations such as public key cryptography (e.g. RSA) on sensor nodes. Though the size of message can be increased, it is generally not practical to accommodate long message, since wireless communication is one of the most expensive operation on sensor node. In addition, public key operation usually involves many expensive computations (e.g. large integer modular exponentiations).

- *Node compromise* : Different from traditional wireless networks, where each individual node may be physically protected, the large scale of wireless sensor networks makes it impractical to protect or monitor each individual sensor node physically. An attacker may capture or compromise one or a number of sensor nodes without being noticed. If sensor nodes are compromised, the attacker learns all the secrets stored on them and may launch a variety of malicious actions against the network through these all compromised nodes. For example, the compromised nodes may discard all important messages in order to hide some critical events from being noticed, or report observations that are significantly different from those observed by non-compromised nodes in order to mislead any decision made based on these data. The result will be even worse if the nodes that provide some critical functions (e.g. data aggregation) are compromised. Though using tamper-resistance hardwares may help to protect security sensitive data on sensor nodes, this solution generally increases the cost of an individual sensor node dramatically. An alternative way is to develop security protocols that are *resilient* to node compromise attacks in the sense that even if one or a number of sensor nodes are compromised, the sensor network can still function correctly.

- *Local Computation and Communication versus Global Threats* : The sensor applications in a typical sensor network are usually based on local computation and communication. For example, they may make decisions based on the message exchanged between neighbor nodes. However, adversaries usually are much more powerful and resourceful than sensor nodes, and they usually have a global view of the network (e.g. topology). Thus, we have to use resource-constrained sensor nodes t deal with very powerful attacks.

1.3 Security Issues in Sensor Networks

An important step for protecting sensor networks is the development of fundamental security tools such as broadcast authentication and key management. These fundamental tools provide basic building blocks for us to implement various security mechanisms for sensor networks. On the other hand, sensor applications are usually supported by many components such as routing and localization. These components clearly have to be protected properly in hostile environments.

Pairwise Key Establishment: Pairwise key establishment is another important security service. It enables sensor nodes to communicate securely with each other using cryptographic techniques. The main problem is to establish a secure key shared between two communicating sensor nodes. However due to resource constraints on sensor nodes, it is not feasible for them to use traditional pairwise key establishment techniques such as public key cryptography.

Instead of the public key cryptographic techniques, sensor nodes may establish keys between each other through *key pre-distribution*, where keying materials are pre-distributed to sensor nodes before deployment. As two extreme cases, one may setup a *global key* among the network so that two sensor nodes can establish a key based on this key, or one may assign each sensor node a unique random key with each of the other nodes. However, the former is vulnerable to the compromise of a single node, and the latter introduces huge storage overhead at sensor nodes.

Chapter 2

Background

2.1 Pairwise Key Establishment

1. **Key Pre-Distribution Techniques in Sensor Networks**
   (a) *Probabilistic Key Pre-Distribution*
   (b) *Polynomial-Based Key Pre-Distribution*
   (c) *Polynomial Pool-Based Key Pre-Distribution*
   (d) *Random Subset Assignment Key Pre-Distribution*
   (e) *Grid-Based Key Pre-Distribution*

2.1.1 Probabilistic Key Pre-Distribution

The main idea is to have each sensor node randomly pick a set of keys from a key pool before deployment so that any two sensor nodes can share a common key with certain probability.

Specifically, a setup server, which is assumed to be trusted, generates a large pool of random keys, where each key has a unique ID. Each sensor node then gets assigned a random subset of keys as well as their IDs from this pool before the deployment of this sensor node.

In order to establish a common key directly between two sensor nodes after deployment, the nodes only need to identify a common key ID they share. This can be achieved by exchanging the list of key IDs they have.

2.1.2 Polynomial-Based Key Pre-Distribution

To predistribute pairwise keys, the (key) setup server randomly generates a bivariate t-degree polynomial

$$f(x, y) = \Sigma_{i,j=0}^{t} a_{i,j} x^i y^j$$

over a finite field F_q, where q is a prime number that is large enough to accommodate a cryptographic key, such that it has the property of

$$f(x, y) = f(y, x).$$

It is assumed that each sensor has a unique ID. For each sensor i, the setup server computes a *polynomial share* of $f(x, y)$, that is, $f(i, y)$. This polynomial share is pre-distributed to node i. Thus for any two sensor nodes i and j, node i can compute the common key $f(i, j)$ by evaluating $f(i, y)$ at point j, and node j can compute the same key

$$f(j, i) = f(i, j)$$

by evaluating $f(j, y)$ at point i. As a result, nodes i and j can establish a common key $f(i, j)$.
In this approach, each sensor node i needs to store a t-degree polynomial $f(i, y)$, which occupies $(t+1)logq$ storage space. To establish a pairwise key, both sensor nodes need to evaluate the polynomial at the ID of the other sensor node.There is no communication overhead during the pairwise key establishment process. The security proof ensures that this scheme is unconditionally secure and t-collusion resistant. That is, the coalition of no more than t compromised sensor nodes knows nothing about the pairwise key between any two non-compromised nodes.

2.1.3 Polynomial Pool-Based Key Pre-Distribution

The polynomial-based key predistribution scheme has some limitations. In particular, it can only tolerate no more than t compromised nodes, where the value of t is limited by the memory available in sensor nodes. Indeed, the larger a sensor network is, the more likely an adversary compromises more than t sensor nodes and then the entire network. Pairwise key establishment is performed in three phases: *setup, direct key establishment, and path key establishment*. The setup phase is performed to initialize the sensors by distributing polynomial shares to them. After being deployed, if two sensors need to establish a pairwise key, they first attempt to do so through direct key establishment. If they can successfully establish a common key, there is no need to start path key establishment. Otherwise, these sensors start path key establishment, trying to establish a pairwise key with the help of other sensors.

1. **Phase 1: Setup** The setup server randomly generates a set $\mathcal{F}$ of bivariate t-degree polynomials over the finite field F_q. To identify the different polynomials, the setup server may assign each polynomial a unique ID. For each sensor node i, the setup server picks a subset of polynomials $\mathcal{F}_i \subseteq \mathcal{F}$ and assigns the polynomial shares of these polynomials to node i. The main issue in this phase is the *subset assignment* problem, which specifies how to pick a subset of polynomials from $\mathcal{F}$ for each sensor node.

2. **Phase 2: Direct Key Establishment** A sensor node starts *phase 2* if it needs to establish a pairwise key with another node. If both sensors have polynomial shares on the same bivariate polynomial, they can establish the pairwise key directly using the polynomial-based key predistribution scheme. The main issue in this phase is the *polynomial share discovery* problem, which specifies how to find a common bivariate polynomial of which both nodes have polynomial shares.

3. **Phase 3: Path Key Establishment** If direct key establishment fails, two sensor nodes will have to start *phase 3* to establish a pairwise key

with the help of other sensors. We call a sequence of nodes as a path, or key path, since the purpose of such a path is to establish a pairwise key. To establish a pairwise key with node j, a sensor node i needs to find a path between itself and node j such that any two adjacent nodes in the path can establish a pairwise key directly. Then either node i or j initiates a request to establish a pairwise key with the other node through the intermediate nodes along the path. A pairwise key established in this phase is called an *indirect key*. The main issue in this phase is the *Path discovery* problem, which specifies how to find a path between two sensor nodes.

2.1.4 Random Subset Assignment Key Pre-Distribution

For each sensor, the setup server selects a random subset of polynomials in $\mathcal{F}$ and assigns their polynomial shares to the sensor.

This scheme can be considered as an extension to the basic probabilistic scheme. Instead of randomly selecting keys from a large key pool and assigning them to sensors, This method randomly chooses polynomials from a polynomial pool and assigns their polynomial shares to sensors. Now let us describe this scheme by instantiating the three components in the general framework.

(a) *Subset Assignment*: The setup server randomly generates a set $\mathcal{F}$ of s bivariate t-degree polynomials over the finite field F_q. For each sensor node, the setup server randomly picks a subset of s polynomials from $\mathcal{F}$ and assigns shares as well as the IDs of these s polynomials to the sensor node.

(b) *Polynomial share discovery*: Since the setup server does not predistribute enough information to the sensor nodes for polynomial share discovery, sensor nodes that need to establish a pairwise key have to find out a common polynomial with real-time discovery techniques. To discover a common bivariate polynomial, the source node discloses a list of polynomial IDs to the destination node. If the destination node finds that they have shares on the same polynomial, it informs the source node the ID of this polynomial; otherwise, it replies with a message that contains a list of its polynomial IDs, which also indicates that the direct key establishment fails.

(c) *Path Discovery*: If two sensor nodes fail to establish a direct key, they need to start path key establishment phase. During this phase, the source node tries to find an- other node that can help it setup a pairwise key with the destination node. Basically, the source node broadcasts two list of polynomial IDs. One includes the polynomial IDs at the source node, and the other includes the polynomial IDs at the destination node. These two lists are available at both the source and the destination nodes after the polynomial share discovery. If one of the nodes that receives this request is able to establish direct keys with both the source and the destination nodes, it replies with a message that contains two encrypted copies of a randomly generated

key: one encrypted by the direct key with the source node, and the other encrypted by the direct key with the destination node. Both the source and the destination nodes can then get the new pairwise key from this message.

2.1.5 Grid-Based Key Pre-Distribution

This scheme has a number of attractive properties. First, it guarantees that any two sensors can establish a pairwise key when there is no compromised sensors, provided that the sensors can communicate with each other. Second, this scheme is resilient to node compromise. Even if some sensors are compromised, there is still a high probability of establishing a pairwise key between sensors. Third, a sensor can directly determine whether it can establish a pairwise key with another node, and if it can, which polynomial should be used. As a result, there is no communication overhead during polynomial share discovery.

Suppose a sensor network has at most N sensor nodes. The grid-based key predistribution scheme then constructs a $m \times m$ grid with a set of $2m$ polynomials

$$\{f_i^c(x,y), f_i^r(x,y)\}_{i=0,\ldots,m-1},$$

where $m = \sqrt{N}$. As shown in Figure (a), each row i in the grid is associated with a polynomial $f_i^r(x,y)$ and each column i is associated with a polynomial $f_i^c(x,y)$. The setup server assigns each sensor in the network to a unique intersection in this grid. For the sensor at the coordinate $\langle i,j\rangle$, the setup server distributes the polynomial shares of $f_i^c(x,y)$ and $f_j^r(x,y)$ to the sensor. As a result, sensor nodes can perform share discovery and path discovery based on this information.
(a) The grid (b) An example order of node assignment
Figure : Grid-based key predistribution

1. **Subset Assignment:** The setup server randomly generates $2m$ t-degree bivariate polynomials

   $$\mathcal{F} = \{f_i^c(x,y), f_i^r(x,y)\}_{i=0,\ldots,m-1}$$

   over a finite field F_q, where $m = \sqrt{N}$. For each sensor, the setup server picks an unoccupied intersection (i,j) in the grid and assigns it to the node. Thus, the ID of this sensor is $ID = \langle i,j\rangle$. The setup server then distributes polynomials share $\{ID, f_i^c(j,y), f_j^r(i,y)\}$ to this sensor node. To facilitate path discovery, we require that the intersections allocated to sensors are densely selected within a rectangle area in the grid. Figure (b) shows a possible order to allocate intersections to the sensors. It is easy to see that if there exist nodes at $\langle i,j\rangle$ and $\langle i',j'\rangle$, then there must be a node at either $\langle i,j'\rangle$ or $\langle i',j\rangle$ or both.

2. **Polynomial Share Discovery:** To establish a pairwise key with node j, node i checks whether $c_i = c_j$ or $r_i = r_j$. If $c_i = c_j$, both nodes i and j have polynomial shares of $f_{c_i}^c(x,y)$, and they can use the polynomial-based key predistribution scheme to establish a pairwise key directly. Similarly, if $r_i = r_j$, they both have polynomial shares of $f_{r_i}^r(x,y)$, and can establish a pairwise key accordingly. If neither of these conditions is true, nodes i and j go through path discovery to establish a pairwise key.

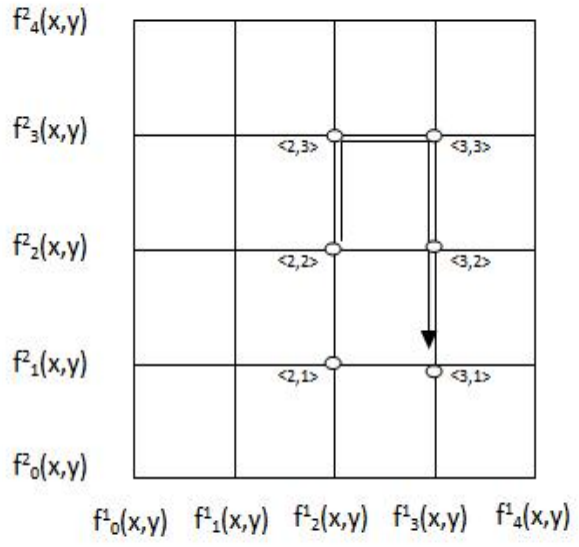

Figure 2.1: (a) An example of hypercube when $n = 2$ and $m = 5$

3. **Path Discovery:** Nodes i and j need to use path discovery if $c_i \neq c_j$ and $r_i \neq r_j$. However, we note that either node $\langle c_i, r_j \rangle$ or $\langle c_j, r_i \rangle$ can establish a pairwise key with both nodes i and j. Indeed, if there is no compromised node, it is guaranteed that there exists at least one node that can be used as an intermediate node between any two sensors due to the node assignment algorithm. For example, in Figure (a), both node$\langle i, j' \rangle$ and $\langle i', j \rangle$ can help node $\langle i, j \rangle$ establish a pairwise key with node $\langle i', j' \rangle$. Note that nodes i and j can predetermine the possible intermediate nodes without communicating with others.
In some situations, both of the above intermediate nodes may have been compromised, or are out of communication range. However, there are still alternative key paths. For example, in Figure (a), besides node $\langle i', j \rangle$ and $\langle i, j' \rangle$, node $\langle i, m-2 \rangle$ and $\langle i', m-2 \rangle$ can work together to help node $\langle i, j \rangle$ setup a common key with node $\langle i', j' \rangle$. Indeed, there are up to $2(m-2)$ pairs of such nodes in the grid.

The Hypercube-Based Key Pre-Distribution

Hypercube-based key pre-distribution is a generalization of grid-based key predistribution. Given a total of N sensor nodes in the network, the hypercube-based scheme constructs an n-dimensional hypercube with m^{n-1} bivariate polynomials arranged for each dimension j,

$$\left\{ f_{\langle i_1, \ldots, i_{n-1} \rangle}(x, y) \right\}_{0 \leq i_1, \ldots, i_{n-1} < m} \text{ where } m = \sqrt[n]{N}.$$

Figure (a) shows a special case of the hypercube-based scheme when $n = 2$ (i.e., the grid-based scheme). In this figure, each column i is associated

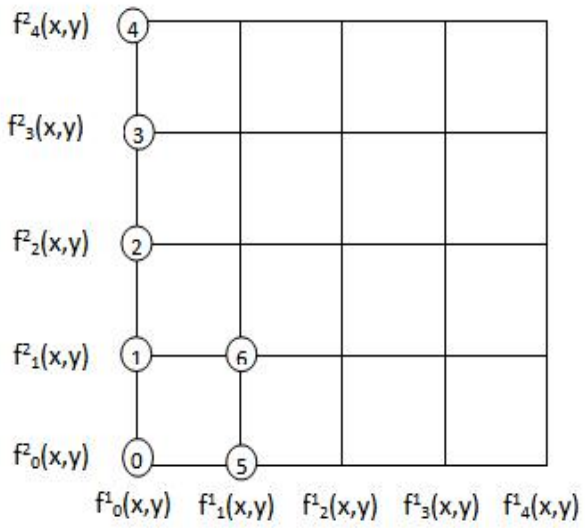

Figure 2.2: (b) An example order of node assignment

with a polynomial $f_i^1(x, y)$, and each row i is associated with a polynomial $f_i^2(x, y)$. The setup server then assigns each node in the network to a unique coordinate in this n-dimensional space. For the sensor node at coordinate $(j_1, \ldots, j_n)$, the setup server pre-distributes the polynomial shares of

$$\left\{ f^1_{\langle j_2,\ldots,j_n\rangle}(x,y), \ldots, f^n_{\langle j_1,\ldots,j_{n-1}\rangle}(x,y) \right\}$$

to this node. As a result, sensor nodes can perform share discovery and path discovery using this pre-distributed information.
We conceptually represent each ID j as $\langle j_1, \ldots, j_n\rangle$, where j_i is called the sub-index of ID j in dimension i.

(a) **Subset Assignment**: The setup server randomly generates $n \times m^{n-1}$ t-degree bivariate polynomials

$$\mathcal{F} = \left\{ f^j_{\langle i_1,\ldots,i_{n-1}\rangle}(x,y) | 1 \le j \le n, 0 \le i_1, \ldots i_{n-1} < m \right\}$$

over a finite field F_q. For each sensor node, the setup server selects an unoccupied coordinate $(j_1, \ldots, j_n)$ in the n-dimensional space and assigns it to this node. This coordinate $\langle j_1, \ldots, j_n\rangle$ is then used as the ID of this node. The setup server then distributes

$$\left\{ ID, f^1_{\langle j_2,\ldots,j_n\rangle}(x,y), \ldots, f^n_{\langle j_1,\ldots,j_{n-1}\rangle}(x,y) \right\}$$

to this sensor node. To facilitate path discovery and guarantee that there is at least one key path exists when there are no compromised nodes and any two nodes can communicate with each other, we always select the coordinate corresponding to the smallest unassigned

ID. Figure (b) shows a possible order to assign coordinates to sensor nodes when $n = 2$. It is easy to see that if there exist nodes at $\langle i, j\rangle$ and $\langle i', j'\rangle$, then there must be a node at either $\langle i, j'\rangle$ or $\langle i', j\rangle$ or both.

(b) **Polynomial Share Discovery**: To establish a pairwise key with node j, node i checks whether they have the same sub-indexes in $n-1$ dimensions. In other words, it checks the Hamming distance d_h between their IDs i and j. If $d_h = 1$, nodes i and j share a common polynomial, and they can establish a direct key using the polynomial-based key pre-distribution scheme; otherwise, they need to go through path discovery to establish an indirect key. For example, if $j_k = i_k$ for all $1 \leq k \leq n-1$ ($d_h = 1$), both nodes i and j have polynomial shares of $f^n_{\langle j_1,\ldots,j_{n-1}\rangle}(x, y)$, and thus can use this polynomial to establish a direct key.

(c) **Path Discovery**: If nodes i and j can not establish a direct key, they need to find a key path between each other in the hypercube. For example, in figure 2.1(a), both of node $\langle 2, 1\rangle$ and $\langle 3, 2\rangle$ can help node $\langle 2, 2\rangle$ establish a pairwise key with node $\langle 3, 1\rangle$. Indeed, if there are no compromised nodes and any two nodes can communicate with each other, it is guaranteed that there are at least one key path which can be used to establish a session key between any two nodes.

Chapter 3

Grid-Based Key Pre-Distribution Using Multivariate Symmetric Polynomial

3.1 Two Dimensional Grid-Based Scheme Using 3-Variate Symmetric Polynomial

Our scheme is based on a t-degree multivariate symmetric polynomial. A t-degree $(k+1)$-variate polynomial is defined as

$$f(x_1, x_2, \ldots, x_k, x_{k+1}) = \sum_{i_1=0}^{t} \sum_{i_2=0}^{t} \cdots \sum_{i_k=0}^{t} \sum_{i_{k+1}=0}^{t} a_{i_1,i_2,\ldots,i_k,i_{k+1}} x_1^{i_1} x_2^{i_2} \ldots x_k^{i_k} x_{k+1}^{i_{k+1}}.$$

All coefficients of the polynomial are chosen from a finite field F_q, where q is a prime that is large enough to accommodate a cryptographic key. All calculations are performed over the finite field F_q. A $(k+1)$-tuple permutation is defined as a bijective mapping

$$\sigma : [1, k+1] \longrightarrow [1, k+1].$$

By choosing all the coefficients according to

$$a_{i_1,i_2,\ldots,i_k,i_{k+1}} = a_{i_{\sigma(1)},i_{\sigma(2)},\ldots,i_{\sigma(k)},i_{\sigma(k+1)}}$$

for any permutation σ, we can obtain a symmetric polynomial in that

$$f(x_1, x_2, \ldots, x_k, x_{k+1}) = f(x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(k)}, x_{\sigma(k+1)}).$$

In two dimensional grid we use 3-variate symmetric polynomial. Let us assume that our network has N nodes. Consider a 2-dimensional grid with u rows and v columns, Where u,v are integers such that $u \times v \geq N$. The setup server assign each sensor node in the network to a unique non-occupied (i, j) co-ordinate in this grid. Where i and j are row and column number in the grid respectively,such that $1 \leq i \leq u$ and $1 \leq j \leq v$. The ID of the sensor node associated with the co-ordinate (i, j) is represented by $\langle i, j \rangle$. Key establishment will be done in two phases

(a) *Polynomial Share Pre-distribution*

(b) *Key Establishment Mechanism*

   i. Direct Key Establishment

   ii. Indirect Key Establishment

3.1.1 Polynomial Share Pre-Distribution

Polynomial share pre-distribution is performed prior to the network deployment by a trusted setup server. The setup server generates a global tri-variate t-degree symmetric polynomial. For each node $\langle i, j \rangle$, the polynomial share assigned to it is $f(i, j, x_3)$. Therefore every node in the network is storing t-degree univariate polynomial having $(t + 1)$ co-efficients over the finite field F_q. Storing the uni-variate polynomial means storing its co-efficients. Before nodes deployment these co-efficients are preloaded in to the sensor nodes and used for computing communication key during key establishment process.

3.1.2 Key Establishment Mechanism

Now after deployment, if two nodes wants to communicate they will use their polynomial shares to establish a communication key. Depends upon the nodes key establishment can be done in any one of the following ways.

(a) *Direct Key Establishment*

(b) *Indirect Key Establishment*

Direct Key Establishment

Suppose node $\langle i_1, j_1 \rangle$ and $\langle i_2, j_2 \rangle$ wants to esblish a communication key and $\{i_1, j_1\} \cap \{i_2, j_2\} \neq \Phi$, say $\{i_1, j_1\} \cap \{i_2, j_2\} = \{i_1\}$ with out loss of generality(W.L.G) say, $j_2 = i_1$, then node $\langle i_1, j_1 \rangle$ computes $f(i_1, j_1, i_2)$ and node $\langle i_2, j_2 \rangle$ computes $f(i_2, i_1, j_1)$. Because of our polynomial is symmetric $f(i_1, j_1, i_2) = f(i_2, i_1, j_1)$. By using this calculated value as established key these nodes will communicate each other. Suppose $\{i_1, j_1\} \cap \{i_2, j_2\} = \Phi$ then these nodes can't establish a direct communication key, so to establish a communication key we will go for *Indirect key establishment*.

Indirect Key Establishment

Suppose node $\langle i_1, j_1 \rangle$ and $\langle i_2, j_2 \rangle$ wants to esblish a communication key and $\{i_1, j_1\} \cap \{i_2, j_2\} = \Phi$ then these two nodes will communicate with the help of other node(s). Here we can show that these two nodes can communicate with the help of node $\langle i', j' \rangle$,

$$\text{where } \langle i', j' \rangle \in$$
$$\{\langle i_1, i_2 \rangle, \langle i_1, j_2 \rangle, \langle j_1, i_2 \rangle, \langle j_1, j_2 \rangle, \langle i_2, i_1 \rangle, \langle j_2, i_1 \rangle, \langle i_2, j_1 \rangle, \langle j_2, j_1 \rangle\}.$$

Since $\{i_1, j_1\} \cap \{i', j'\} \neq \Phi$ they can communicate directly. Similarly $\{i_2, j_2\} \cap \{i', j'\} \neq \Phi$ they can communicate directly. Therefore even if one node is compromised, *seven* different intermediate nodes are there to help nodes $\langle i_1, j_1 \rangle$ and $\langle i_2, j_2 \rangle$.

3.2 Multi Dimensional(n) Grid-Based (Hyper-Cube) Scheme Using (n+1)-Variate Symmetric Polynomial

Given a total of N sensor nodes in the network, our scheme constructs an n-dimensional hypercube such that $N \leq m^n$. The setup server then assigns each node in the network to a unique unoccupied coordinate in this n-dimensional hypercube and then setup server randomly generates t-degree $(n+1)$-variate symmetric polynomial $f(x_1, x_2, \ldots, x_n, x_{n+1})$. We conceptually represent the ID of the node j located at $(j_1, \ldots, j_n)$ as $\langle j_1, \ldots, j_n \rangle$. Establishment of a communication between any two nodes works in two phases.

(a) *Polynomial Share Pre-Distribution*

(b) *Key Establishment Mechanism*

3.2.1 Polynomial Share Pre-Distribution

Polynomial share pre-distribution is performed prior to the network deployment by a trusted setup server. The setup server generates a global (n+1)-variate t-degree symmetric polynomial. For each node $\langle i_1, \ldots, i_n \rangle$, the polynomial share assigned to it is $f(i_1, \ldots, i_n, x_{n+1})$. Therefore every node in the network is storing t-degree univariate polynomial having $(t+1)$ co-efficients over the finite field F_q. Storing the uni-variate polynomial means storing its co-efficients. Before nodes deployment these co-efficients are preloaded into the sensor nodes and used for computing communication key during key establishment process.

3.2.2 Key Establishment Mechanism

Now after deployment, if two nodes wants to communicate they will use their polynomial shares to establish a communication key. Depends upon the nodes key establishment can be done in any one of the following ways.

1. *Direct Key Establishment*

2. *Indirect Key Establishment*

Direct Key Establishment

Suppose node $\langle i_1, \ldots, i_n \rangle$ and $\langle j_1, \ldots, j_n \rangle$ wants to esblish a communication key. If these two nodes have only one different element and the remaining $n-1$ are the same, Then we can establish key directly without help of any intermediate nodes. Assume that they have only one in different, W.L.G say node i is $\langle i_1, \ldots, i_n \rangle$ and node j is $\langle i_1, \ldots, i_{n-1}, j_n \rangle$ then node i will compute $f(i_1, \ldots, i_n, j_n)$ and node j will compute $f(i_1, \ldots, i_{n-1}, j_n, i_n)$, since our polynomial is symmetric these two values are same. Therefore these two nodes use this value as established common key and these nodes will communicate each other. If node $\langle i_1, \ldots, i_n \rangle$ and $\langle j_1, \ldots, j_n \rangle$ have more than one different elements then we can't establish key directly, we will go for *indirect key establishment*.

Indirect Key Establishment

Node $\langle i_1, \ldots, i_n \rangle$ and node $\langle j_1, \ldots, j_n \rangle$ wants to establish a communication key, and these two nodes have more than one different elements, say suppose two different elements W.L.G let node i is $\langle i_1, \ldots, i_n \rangle$, node j is $\langle i_1, \ldots, i_{n-2}, j_{n-1}, j_n \rangle$. Now we will show that there are huge number of intermediate nodes to help these two nodes, like nodes

$$\langle i_1, \ldots, i_{n-2}, i_{n-1}, j_n \rangle,\ \langle i_1, \ldots, i_{n-2}, i_n, j_n \rangle,\ \text{etc...}$$

and any permutation of these nodes. Since node $\langle i_1, \ldots, i_{n-2}, i_{n-1}, j_n \rangle$ and node $\langle i_1, \ldots, i_n \rangle$ have only one different element, so these two can directly establish key as explained above in *Direct Key Establishment* method, similarly $\langle i_1, \ldots, i_{n-2}, i_{n-1}, j_n \rangle$ and node $\langle i_1, \ldots, i_{n-2}, j_{n-1}, j_n \rangle$ have only one different element, so these two can directly establish key. Therefore $\langle i_1, \ldots, i_{n-2}, i_{n-1}, j_n \rangle$ node acts as an intermediate node, in fact any permutation of this node acts as an intermediate node.
Suppose node $\langle i_1, \ldots, i_n \rangle$ node $\langle j_1, \ldots, j_n \rangle$ have three different elements, W.L.G say node i is $\langle i_1, \ldots, i_n \rangle$, node j is $\langle i_1, \ldots, i_{n-3}, j_{n-2}, j_{n-1}, j_n \rangle$. Now we will show that there are huge number of paths to help these two nodes, nodes like

$$\langle i_1, \ldots, i_{n-3}, i_{n-2}, i_{n-1}, j_n \rangle, \langle i_1, \ldots, i_{n-3}, i_{n-1}, j_{n-1}, i_n \rangle,$$
$$\langle i_1, \ldots, i_{n-3}, i_n, i_{n-1}, j_n \rangle \text{ etc...}$$

and any permutation of these nodes. Now each of these nodes

$$\text{for example } \langle i_1, \ldots, i_{n-3}, i_{n-2}, i_{n-1}, j_n \rangle$$

have only one dirrecrent element with node $\langle i_1, \ldots, i_n \rangle$ so they can establish key directly as explained above in *Direct Key Establishment* method; and two different elements with node $\langle i_1, \ldots, i_{n-3}, j_{n-2}, j_{n-1}, j_n \rangle$, we already explained how to communicate when they have two different elements. In this way we can find a path for communication.

Generalization Suppose node $\langle i_1, \ldots, i_n \rangle$ node $\langle j_1, \ldots, j_n \rangle$ have k different elements, W.L.G say node i is $\langle i_1, \ldots, i_n \rangle$, node j is $\langle i_1, \ldots, i_{n-k}, j_{n-k+1}, \ldots j_n \rangle$. Now look at these nodes

$$\langle i_1, \ldots, i_{n-k}, i_{n-k+1}, i_{n-k+2}, \ldots i_{n-1}, j_n \rangle,$$
$$\langle i_1, \ldots, i_{n-2}, j_{n-1}, i_n \rangle, \langle i_1, \ldots, i_{n-1}, j_{n-2} \rangle, \text{etc...}$$

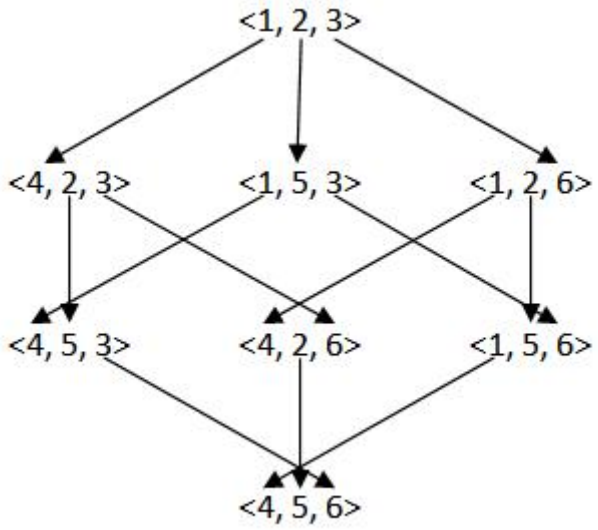

Figure 3.1: Example(a)
The above figure depicts all distinct paths from node $\langle 1, 2, 3\rangle$ to node $\langle 4, 5, 6\rangle$ in *Hypercube-Based Key Pre-Distribution*. The number of distinct paths in this scheme are 3!

and any permutation of these nodes. These nodes and node i has only one different element, so these two can communicate directly; and node j has $k-1$ different elements, so by doing inductively we will find intermediate nodes in such a way that at each step the number of different elements will be decreased by one. By doing this process we will establish huge number of paths between the nodes to communicate each other.

Suppose there are k elements of node i are different from node j. Let us define two sets P and Q in such a way that, elements of P are replaced with elements of Q to get intermediate nodes of path from node i to reach node j. First, we define P and Q as $I - J$ and $J - I$ respectively, where $I = \{i_1, \ldots, i_n\}$ and $J = \{j_1, \ldots, j_n\}$. This should give number of elements in P and Q as k. This is fine if i and j are having n distinct values.

Let us see what happens if some values are same. Consider an example, let node i is $\langle 1, 1, 2, 3\rangle$ and node j is $\langle 1, 3, 3, 4\rangle$, we have to change $1, 2$ of i to $3, 4$ to reach j. But, according to the above definition for P and Q, $P = \{2\}$ and $Q = \{4\}$.
Now the problem modified to find the difference in number of instances of each value $1, 2, \ldots, m$ in i and j. This made us to introduce n_r^i which gives the number of instances of r in node i, where $1 \leq r \leq m$. Now the sets P and Q are constructed as

$$\text{for each } 1 \leq r \leq m, \text{ put } r \text{ in } P \text{ if } n_r^i > n_r^j \text{ and}$$
$$\text{put } r \text{ in } Q \text{ if } n_r^i < n_r^j.$$

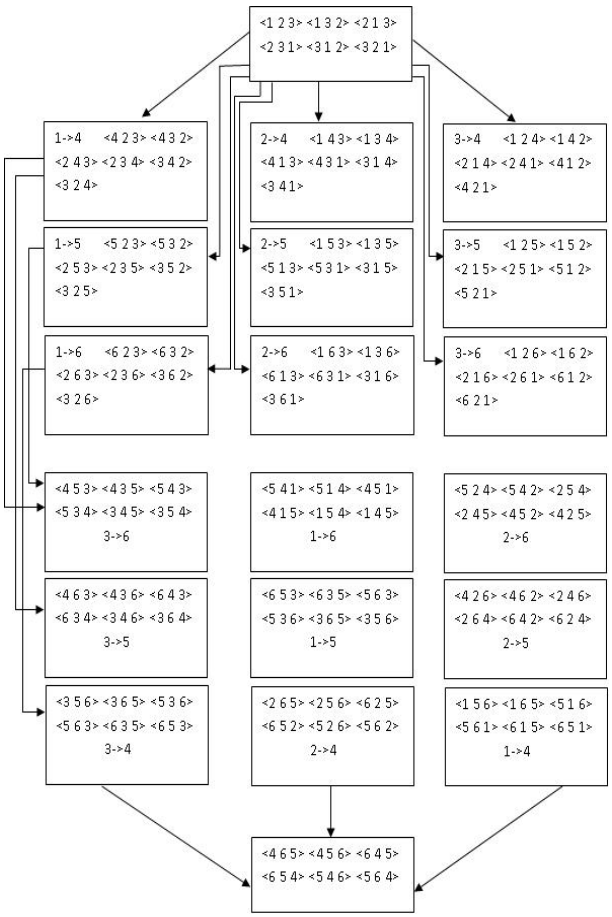

Figure 3.2: Example(b)
The above figure depicts all distinct paths from node $\langle 1, 2, 3\rangle$ to node $\langle 4, 5, 6\rangle$ in *Hypercube-Based(3-dimensional) Key Pre-Distribution using 4-variate symmetric polynomial*. The number of distinct paths in this scheme are $(3 * 3)(2 * 2)(3! * 3!)$.

Now for each $p \in P$ and for each $q \in Q$ we can have a new intermediate node in which p is replaced by q, and all permutations of that intermediate node. The new intermediate node has $n_p^i - 1$ instances of p and $n_q^i + 1$ instances of q. Therefore the number of intermediate nodes(with all permutations)is given by

$$\frac{n!}{(n_p^i-1)!(n_q^i+1)!\prod_{r\neq p,q}(n_r^i)!}.$$

As p varies in P and q varies in Q the total number of possible intermediate nodes at one step is given by

$$\sum_{(p,q)\in P\times Q} \frac{n!}{(n_p^i-1)!(n_q^i+1)!\prod_{r\neq p,q}(n_r^i)!}$$

Chapter 4

Conclusion

In this report we developed a n-dimensional Grid-Based scheme using a single multivariate symmetric polynomial. First, the Grid Based scheme using tri-variate symmetric polynomial can be easily extended to a n-dimensional or Hyper-Cube Based scheme using (n+1)-variate symmetric polynomial. By using only one single multivariate polynomial depending on the dimension of the Hyper-Cube; for communication between any two nodes there are large number of paths in our scheme comparitively Hyper-Cube Based Scheme discussed in chapter 2. In our scheme polynomial share of node i is $f(i_1, \ldots, i_n, x_{n+1})$, we have to store the coefficients of this univariate polynomial, i.e $(t+1)$ coefficients. Where as in Hyper-Cube Based scheme discussed in chapter 2 has to store n univariate polynomials. Therefore in terms of memory our scheme is efficient.

Bibliography

[1] C.BLUNDO, A.DE SANTIS, Amir HERZBERG, S.KUTTEN, U.VACCARO, AND M.YUNG. Perfectly secure key distribution for dynamic conferences. *In Advances in Cryptology - CRYPTO '92, LNCS 740*, 471-486,1993.

[2] CHAN, H., PERRIG, A., AND SONG, D. Random key predistribution schemes for sensor networks. *In IEEE Symposium on Research in Security and Privacy*, 197-213,2003.

[3] ESCHENAUER, L. AND GLIGOR, V. D. 2002. A key-management scheme for distributed sensor networks. *In Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 41-47.

[4] L. Eschenauer and V. Gligor, A key management scheme for distributed sensor networks, in Proc. ACM CCS, Nov. 2002, pp. 41-47.

[5] LIU, D. AND NING, P. 2003b. Establishing pairwise keys in distributed sensor networks, *In Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*, Pages 52-61.

[6] LIU, D. , NING, P. , and R.Li. A key-management scheme for distributed sensor networks, *ACM Transactions on information and System Security(TISSEC)*,2004.

[7] Yun Zhou and Yuguang Fang, Scalable and Deterministic Key Agreement for Large Scale Networks, *IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS*, VOL.6, NO.12, DECEMBER 2007.

[8] A. K. Pathan, T. T. Dai, C. S. Hong, A Key Management Scheme with Encoding and Improved Security for Wireless Sensor Networks,*ICDCIT* 2006, LNCS 4317, pp. 102-115.

[9] Sung Jin Choi and Hee Yong Youn, An Efficient Key Pre-distribution Scheme for Secure Distributed Sensor Networks, EUC Workshops 2005,LNCS 3823,pp. 1088-1097,2005.

[10] Rolf Blom. An optimal class of symmetric key generation systems. In Proceeding of EUROCRYPT, pages 335–338, 1984.

[11] Seyit Ahmet C¸ amtepe and Bulent Yener. Combinatorial design of key distribution mechanisms for wireless sensor networks. IEEE/ACM Trans. Netw.,15(2):346–358, 2007.

[12] Chang Won Park, Sung Jin Choi, and Hee Yong Youn, A Noble Key Predistribution Scheme with LU Matrix for Secure Wireless Sensor Networks CIS 2005, Part 11, LNAI 3802, pp. 494-499,2005.

[13] Sushmita Ruj, Application of Combinatorial Structures to Key Predistribution in Sensor Networks and Traitor Tracing Thesis, submitted to Indian Statistical Institute in partial fulfillment of the requirements for the award of the degree of Doctor of Philosophy by Sushmita Ruj Applied Statistics Unit Indian Statistical Institute Kolkata,India.

[14] M.Tech dissertation report submitted by Ashish Kumar in 2008.