APPLICATION OF TRANSVERSAL DESIGN AND SECURE PATH KEY ESTABLISHMENT FOR KEY PRE-DISTRIBUTION IN WSN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF DEGREE OF

Master of Technology

in

Computer Science and Engineering

(Specialization: Information Security)

By

Subhasish Dhal

Department of Computer Science and Engineering
National Institute of Technology, Rourkela
Orissa-769 008, India

May 2009

APPLICATION OF TRANSVERSAL DESIGN AND SECURE PATH KEY ESTABLISHMENT FOR KEY PRE-DISTRIBUTION IN WSN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF DEGREE OF

Master of Technology

in

Computer Science and Engineering

(Specialization: Information Security)

By

Subhasish Dhal

Roll No-207CS212

Under the Supervision of

Prof. Pabitra Mohan Khilar

Department of Computer Science and Engineering
National Institute of Technology, Rourkela
Orissa-769 008, India
May 2009

To my parents

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Orissa, India.

Certificate

This is to certify that the work in the thesis entitled **"Application of Transversal Design and Secure Path Key Establishement in WSN"** submitted by **Mr. Subhasish Dhal** is a record of an original research work carried out by him under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Master of Technology in the specialization of Information Security in the department of Computer Science and Engineering, National Institute of Technology Rourkela. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere

Prof. Pabitra Mohan Khilar
Professor
Dept. of Computer Science & Engineering
National Institute of Technology
Rourkela-769 008 Orissa (India)

Place: NIT Rourkela
Date: 26 May 2009

ACKNOWLEDGEMENT

A journey is easier when you travel together. Interdependence is certainly more valuable than independence. During preparation of this thesis, I have accompanied and supported by many people. It is my pleasure to have opportunity to express my gratitude for all of them.

The first person I would like to thank is my supervisor Prof. Pabitra Mohan Khilar. From the very beginning, he has inspired me to start a new era. Everybody can be innovative, but to bring it out from one is an art. Prof. khilar did this exactly for preparing this thesis. He is such a magician that when I felt any difficulty regarding my thesis, his innovative ideas forced me to feel that difficulty is a word which is really difficult to find from him. Above all, he is nice human being and always stood by me during my hard times.

I am indebted to Prof. B. Majhi, Prof. S. K.Rath, Prof. A. K. Tururk and Prof. S. K. Jena for their guidance throughout my study at NIT Rourkela. I am also thankful to all the other teaching and non teaching members of the department of Computer Sc. & Engg. for their kind help during my study in this institution.

My research was initiated by Professor Bimal Roy, a great personality in ISI Kolkata. He was my supervisor during my stay at ISI Kolkata, where I went to pursue a summer internship. He has helped me to learn a lot of research areas in information security where people need to apply their innovative ideas for securing this universe. I am lot of indebted to him.

During stay at ISI Kolkata, I have also come in contact with a number of highly skillful researchers in ASU unit of ISI Kolkata. Ms. Sushmita Ruj is one of such researchers who helped me a lot regarding my research topic. I am thankful to her as well as the others in ASU unit.

My friend Mr. Anupam Pattanayak played a very vital role which has boosted my morale during trouble hours. Mr. Deepak Krishnakutty is another friend who has helped me a lot to fix any technological problem I have faced. Likewise, all the friends during stay in NIT Rourkela provided me such an environment which makes me feel as a part of happy family. My language is not strong enough to express my gratefulness to all of them.

My parents and my grand mother were very supportive throughout the duration of my study in NIT Rourkela. My elder brother has guided me a lot throughout my study. I am indebted to all of them.

Subhasish Dhal

CONTENTS

ABSTRACT

Wireless sensor network is composed of a number of sensor devices which can communicate with each other through radio wave. The sensor devices are limited with computation ability, communication ability, and memory capacity and battery power. This makes the implementation of any task in Wireless Sensor Network is very challenging. Amid various requirements, secure communication in Wireless sensor Network is a major requirement. Suppose two or more sensor nodes want to communicate with each other securely, they need such an environment which can fulfill all the security requirements amid the constraints mentioned earlier. Therefore, secure communication in this network is not an easy task. Two or more nodes can communicate using any cryptography scheme which can be applicable to this network. Nodes under communication process have to use one or more key for encryption and decryption. Single key for the entire network can serve for encryption and decryption of shared information. However compromising of that key may reveal the whole communication in the network. Therefore, although a single key for an entire network provides a certain range of security to the communication of the network, the resiliency of the network is very low which is not at all acceptable for secure communication. Keeping shared keys for every other node in the network is another option. However, increment of number of nodes in the network increases the key ring size of each node. Although it provides maximum resiliency, however, it suffers from non scalability due to memory constraints of sensor node. Another scheme is public key cryptography, which requires public key and private key for secure communication. It provides good resiliency to the network. However, it consumes much computation which is a limitation for its application in wireless sensor network.

Key pre-distribution is an optimum scheme which loads a finite number of keys to each node taking from a set of predefined keys before deployment of the network. Pair of node which wants to communicate with each other searches for existence of any common key between them and if find start communication using that common key. If no such common key found, they establish a path for exchange of temporarily generated key and start communication using that key. Several key pre-distribution schemes have been proposed for distributing keys for secure communication.

Pre-key distribution with merging of blocks is one of the major key pre-distribution schemes. We have studied that merging of nodes randomly incurred an

amount of communication cost due to its randomness. We propose a scheme which will merge different blocks in a deterministic way yields a pattern of block ids in a node. Our aim is to decrease the communication task during key establishment. For our case, the communication cost during common key establishment is only *O(1)* which is constant, whereas in case of random merging it is *O(z),* where *z* is the merging factor. Therefore, scheme proposed by us mostly suitable for this type of network.

Again in case of those communications which require temporary key, the communication is not secure due to the fact that if any intermediate nodes in the path between actual communicators become compromise, then the newly generated communication is revealed to the attacker. We have proposed two schemes which provide security to such temporarily generated key. One of them is Identity based public key cryptography for path key establishment which exchange the newly generated temporary key using Identity based public key encryption process using ηT pairing as bilinear tool. Although Public key encryption along with pairing needs only once for a particular session, however, due to public key encryption, it may not be appropriate for Wireless Sensor Networks. Therefore, we have revised our scheme and proposed another scheme Identity based symmetric key cryptography for path key establishment. This scheme consumes less computation cost due to symmetric approach for encryption of temporarily generated key. Therefore, this scheme is more appropriate for application in wireless Sensor Networks.

Thus for the purpose of our thesis work, we have proposed a scheme which optimize the Key-pre Distribution strategy by using Deterministic technique of merging blocks to form node and hence facilitates less communication cost for pair-wise common key establishment. Again, for securing temporary key during Path Key Establishment, we have proposed two schemes which provide full security to the temporary key.

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Chapter 1

INTRODUCTION

1.1 INTRODUCTION

Wireless Sensor Network is a distributed network of a collection of sensor nodes. Sensors are inexpensive, low-power devices which have limited resources. They are small in size, and have wireless communication capability within short distances. A sensor node typically contains a power unit, a sensing unit, a processing unit, a storage unit, and a wireless transmitter / receiver. The application of sensors is in various areas. Some of them are as

(i) To detect and characterize Chemical, Biological, Radiological, Nuclear, and Explosive attacks and material.

(ii) To detect and monitor environmental changes in plains, forests, oceans, etc.

(iii) Wireless surveillance sensor networks for providing security in shopping malls, parking garages, and other facilities.

(iv) Military sensor networks to detect and gain as much information as possible about enemy movements, explosions, and other phenomena of interest.

The applications guiding us that wireless sensor network provides certain capabilities and enhancements in operational efficiency in civilian applications as well as assist in the national effort to increase alertness to potential terrorist threats.

A wireless sensor network (WSN) is composed of large number of sensor nodes with limited power, computation, storage and communication capabilities. These resource constraints make implementation of every application on these networks a challenging task. Sensor network has wide application in military as well as civilian purposes. Sensor nodes within the network communicate and they exchange vital secure information with each other. Therefore, sensor network should provide a secure environment for these communications. There are several solutions [1] for secure communication in the network is as

1. Single key for the whole network

   All the nodes in the network share a single key. However, the limitation of this solution lies in the fact that if any one node becomes compromise, the whole network becomes compromise. Therefore, the resiliency of this scheme is very low which make it inapplicable.

2. Shared key for every other node in the network

Each node in the network keep shared key for every other nodes. During communication between a pair of node, the shared key corresponding to that pair would be used for encryption or decryption. Therefore, each node need to keep *n-1* number of keys whereas the total number of nodes in the whole network is *n*. Compromise of any node merely disconnect that node from the network. However, this does not affect the rest of the network. The limitation of this strategy is that it can not be scalable i.e. if the number of nodes increase, the parameter *n-1* also increases and we know that there is limited memory capacity in each sensor node. Therefore, the large value of *n-1* may not be accommodated in the memory of a sensor node. Although this solution gives best resiliency to the network till it can not be applicable for a large network.

3. Public Key cryptosystem

Use of public key cryptosystems is not efficient since implementation of public key framework demands processing power at the high end [1]. Implementation of RSA and ECC on 8-bit CPUs has been proposed in [10] recently. However, a closer scrutiny reveals that the algorithms execute in seconds (the range being 0.43-83.2 s); whereas the key pre-distribution (discussed next) involves much less time to execute. Therefore, Public key cryptosystem is not suitable in comparison to key pre-distribution.

4. Pre-key Distribution Scheme (KPS)

This is an optimum scheme [2] which provides a balance between security issues and resource requirements. A set of key (key pool) has been chosen and each key assigned a unique key Id. Keys from this key pool is distributed to the nodes where each node gets same number of keys. Distribution of keys to the nodes is in such a way that any node having common key with maximum number of other nodes. This technique is scalable. Because, even if increase of nodes in the network does not increase the keys in each node.

This scheme has three steps are

i) Key Pre-distribution

ii) Shared-key Discovery and

iii) Path key Establishment

The first step is carried out before deployment of the network. During this step keys are distributed to the nodes. After deployment of the network, any pair of node can communicate using a common key between them. The Discovery of existence of common key between a pair of node is carried out at second step. This step is called *Shared-key Discovery*. If there is no common key between a pair of node u and v, a sequence of intermediary nodes $n_1, n_2, \cdots, n_t$ are found such that every pair of adjacent nodes in the path $u, n_1, n_2, \cdots, n_t, v$ share a pair wise key. This step is called *path key establishment* [3]. Once the common key discovered, all communication occurs using one or more common keys.

Several schemes [2] have been proposed for key pre-distribution in wireless sensor network. In random scheme [9] of key pre-distribution, randomly generated key are selected from the key pool and assigned to the nodes. Each node broadcast their key ids to other nodes. Receiving nodes on the other side compare their own key ids to find out common keys. Due to the randomness of distribution of key ids, this scheme yields an amount of communication effort, as all the key ids need to send for comparison. Deterministic scheme [4] of key pre-distribution to the nodes yields a pattern. Therefore, common key discovery becomes an easier task. Again, distributing a few numbers of keys to the different nodes does not provide a good connectivity between nodes i.e. perhaps all nodes may not communicate directly to other nodes. Decreasing the size of the key pool merely increase the connectivity but compromising of one node would result compromising many other victim links and so that the victim nodes, which are not secure at all.

1.2 MOTIVATION

A Key Pre-distribution scheme proposed by Chakrabarti, Roy and Maitra is merging a number of nodes to produce a new node which is having more keys [1]. This scheme provides a good resiliency of the network. Again, because this scheme is deterministic, it incurs very less communication and computation effort. However, they select z number of blocks randomly to form a node. Due to this randomness, an amount of communication cost incurs during common key establishment. This is because nodes in the pair have block ids which does not have any pattern due to random selection of block. Therefore, common key can be found deterministically if and only if block ids on the other node is known. It suggests the broadcast of block ids of one node to other node which yields an amount of communication cost. Our proposal is to merging of nodes

deterministically so that there will be a pattern of block ids in each node. This requires much less communication during common key establishment.

A pair of node can communicate with each other directly if they have common key between them (assuming the pair is within radio range), otherwise they establish a temporary common key and exchange that key through one or more intermediate node. For example, if there is no common key between a pair of node u and v, a random key is generated and a sequence of intermediary nodes $n_1, n_2, \cdots, n_t$ are found such that every pair of adjacent nodes in the path $u, n_1, n_2, \cdots, n_t, v$ share a pair wise key and the newly generated key is exchanged through the same path by encrypting and decrypting alternatively until reaching to the destination. This phase is known as *path key establishment* [3]. Once the common key established, all communication between u and v occur using that common keys. Therefore, no intermediary nodes require for actual communication. However, as the newly generated random key is accessible to the intermediary nodes, compromise of any of them results of compromising the newly generated key. Hence although the compromise node is not involved in actual communication, still newly generated link becomes victim. Some researchers [11, 12, 13] have proposed modification of routing strategy to merely avoid the attack. However, till now, no proposal has been made to defend the attack. We for the first time proposed two schemes for providing full security to this key. Our first proposal is based on Id based public key cryptography. However, public key cryptography is not suitable to Wireless Sensor Network. Hence, we proposed another scheme which is based on Id based symmetric key cryptography. Although public key cryptography is nowadays suitable to Wireless Sensor Network, sill it will be more appropriate to use symmetric key cryptography to this type of network. Hence our second proposal is more suitably applicable to Wireless Sensor Networks.

1.3 LITERATURE SURVEY

Key Pre-Distribution is an optimum scheme for distribution of keys to the nodes in Wireless Sensor Network. Many researchers have proposed various schemes for key pre-distribution [2]. Eschenauer and Gligor [9] have proposed a basic scheme which applied for the first time to distribute keys into sensor nodes. q-composite is another scheme which requires minimum q number of common keys between a pair of node to generate paire-wise common key between the same pair of node [2]. Combinatorial Design is a tool for distributing keys into various nodes provides a balance of

distribution. Camtepe and Yener [3] have first time applied combinatorial design technique for key pre-distribution. Transversal design is a type of combinatorial design which applies deterministic technique for key pre-distribution. Use of deterministic scheme yields a pattern of key ids in each node. This helps to establish common key between a pair without brad casting key ids into the network. Lee and Stinson [4] have proposed the use of Transversal design for key pre-distribution. However, the resiliency of the network for this scheme is less and hence Chakrabarti, Maitra and Roy [1] have proposed a scheme which merges *z* number of randomly selected blocks to form a node. However, Chakrabarti, Maitra and Roy also have used Transversal design for distribution of keys to different blocks. Due to merging block to form node Chakrabarti, Maitra and Roy have achieved much better resiliency of the network. We have studied these schemes and proposed a scheme which is basically the modification of Chakrabarti, Maitra and Roy's scheme. We have described these schemes in later chapters in detail.

Pair-wise shared Key establishment is a technique to establish pair-wise common key between a pair of node. However, in case of no common key between a pair of node requires establishment of a temporary key and needs to exchange securely through one or more intermediary nodes. In order to enhance the security of symmetric key establishment, Hui Ling and Taieb Znati [12] proposes an End-to-End Pair-wise Key Establishment scheme which leverages multiple paths for key negotiation and establishment However, they have assumed that an attacker can randomly compromise at most *x* out of *n* nodes. This is clearly limiting the security against any huge attack. It is also vulnerable to stop forwarding or Byzantine attacks. Another multiple path scheme has been proposed by Guanfeng Li, Hui Ling and Taieb Znati [13]. This scheme also suffers form the same limitation. A new strategy proposed by Jang-Ping and Sheu Jui-Che Cheng [11] introduce hop by hop authentication scheme for path key establishment. This clearly consumes an amount of computation which may not be applicable to sensor nodes. Thus a few researchers modified the routing strategy [11, 12, 13] for avoiding attacks on key, However till now no proposal have been made to ensure the full security to this temporarily generated key and hence the newly established communication. We for the first time have proposed two schemes for securing this temporary key.

1.4 CONTRIBUTION

We have studied various scheme of Key Pre-Distribution scheme. Merging of nodes proposed by Chakrabarti, Roy and Maitra [1] requires an amount of

communication cost during common key establishment. We have proposed a deterministic merging strategy [22] which yields a particular pattern of block ids in a node. Therefore, no communication require for common key establishment. We have simulated the Key Pre-Distribution scheme proposed by Lee and Stinson, Merging scheme by Chakrabarti, Roy and Maitra. We also have simulated our own proposed scheme. All the simulation is done using C Language. We have simulated the schemes mention earlier to find out the resiliency of the distribution. Here resiliency means the robustness under adverse situation which is measured by a parameter which is *E(s): Fraction of links become compromise on compromise of s number of nodes.* We have compared this parameter for various schemes including our proposed scheme. We have also compared the communication cost during common key establishment for all the schemes including our proposed scheme [22].

During path key establishment, temporarily generated key does not have any security. A few researchers [11, 12, 13] have provided scheme to merely avoid any attack by modifying routing strategies. Hence we proposed a scheme using ID based public key cryptography [23], for providing security to this key. The scheme uses public key encryption and decryption only once for a particular session of communication. However, public key cryptography may not be applicable to Wireless Sensor Network. Therefore, we have modified this scheme and proposed another scheme [23] where any encryption or decryption is done using symmetric key cryptography. We have used ηT pairing to find out a common key between a pair of node. We have analyzed this scheme and found that this scheme can be applicable to such a network which is limited by computation ability.

1.5 THESIS OUTLINE

The organization of rest of the thesis is as follows. In Chapter 2, we have discussed Basic Combinatorial design which includes Balanced Incomplete Block Design (BIBD), Projective Plane and Transversal Design. In Chapter 3, we have described the various Key pre-distribution schemes. In Chapter 4, we have introduced our proposed scheme for Key pre-distribution and provide various simulation results. Chapter 5 has been used to introduce the theoretical concept of Identity based Encryption scheme with Tate and ηT pairing. In Chapter 6, we have introduced our proposal for securing temporary key during Path Key Establishment phase and analyze various security and performance requirements. We have concluded our thesis with future scopes followed by a number of references.

Chapter 2

Basic Combinatorial Design

Chapter 2

BASIC COMBINATORIAL DESIGN

2.1 INTRODUCTION

Combinatorial design theory is interested in arranging elements of a finite set into subsets to satisfy certain properties [7, 8]. It is the study of families of subsets with various prescribed regularity properties. Members of the universal set *S* in a combinatorial design are usually called treatments, or varieties, and the subsets chosen are called blocks. A regular design based on a set *S* having *v* number of elements is a collection of a number of subsets or blocks having *k* number of elements in each subset from *S* such that every member of *S* belongs to *r* number of subsets or blocks. It is usual to write *b* for the number of blocks in a design. Therefore, a regular design has four parameters *v, b, r, and k.* However these parameters are not independent. A regular design is represented as (v,b,r,k)-*design, where, bk = vr.*

A block design is proper if all block have same length. The number of blocks that contain a given treatment is the replication number *r*. If all *v* treatments occur in a block of a design, then the block is called complete. If a regular design has this property, this is called complete. Complete design is of very little interest unless some further structure is imposed (such as in Latin Square). Given a design if at least one block is incomplete then also the design is incomplete. If *v = b*, the design is called symmetric. If *x* and *y* are any two different treatments in an incomplete design, we shall refer to the number of blocks that contain both *x* and *y* as the co valiancy of x and y, and write it as λ_{xy}. Many important designs are concerned with this co valiancy function. For BIBD (discussed below) λ_{xy} is constant.

2.1.1 BALANCED INCOMPLETE BLOCK DESIGN (BIBD)

A *Balanced Incomplete Block Design (BIBD)* [6, 8] is an arrangement of *v* distinct objects into *b* blocks such that each block contains exactly *k* distinct objects, each object occurs in exactly *r* different blocks, and every pair of distinct objects occurs together in exactly λ blocks. The design can be expressed as (*v, k,* λ), or equivalently (*v, b, r, k,* λ), where: $\lambda\ (v-1) = r\ (k-1)$ and $b.k = v.r$.

Following is an example of (*9, 12, 4, 3,1*)-*BIBD :*

1	2	3
4	5	6
7	8	9
1	4	7
2	5	8
3	6	9
1	6	8
2	4	9
3	5	7
1	5	9
2	6	7
3	4	8

Fig 2.1: Key arrangement for BIBD design

The word incidence is used to describe the relationship between blocks and treatments in a design. A block design may be specified by its incidence matrix. If the design has b blocks $B_1, B_2, .. , B_b$ and v treatments $t_1 , t_2, ..., t_v$ define a $v \times b$ matrix A as:

$$a_{ij} = \begin{Bmatrix} 1 & if\ t_i\ belongs\ to\ B_j \\ 0 & otherwise \end{Bmatrix}$$

A BIBD is called Symmetric BIBD or Symmetric Design when $b = v$. A Symmetric Design has four properties:

1. Every block contains $k = r$ elements
2. Every element occurs in $r = k$ blocks
3. Every pair of elements occurs in λ blocks
4. Every pair of blocks intersects in λ elements.

Example: Consider $(v, k, \lambda) = (7, 3, 1)$, or equivalently $(v, b, r, k, \lambda) = (7, 7, 3, 3, 1)$, Symmetric Design. Let $S = \{1, 2, 3, 4, 5, 6, 7\}$ be the set of $|S| = v = 7$ objects. There are $b = 7$ blocks and each block contains $k = 3$ objects. Every object occurs in $r = 3$ blocks.

Every pair of distinct objects occurs in $\lambda = 1$ blocks and every pair of blocks intersects in $\lambda = 1$ objects.

The blocks of the Symmetric Design are:

{1, 2, 3}
{1, 4, 5}
{1, 6, 7}
{2, 4, 6}
{2, 5, 7}
{3, 4, 7}
{3, 5, 6}

2.1.2 PROJECTIVE PLANE

A Finite Projective Plane [8] consists of a finite set P of points and a set of subsets of P, called lines. For an integer n where $n \geq 2$, there are exactly $n^2 + n + 1$ point, and exactly $n^2 + n + 1$ line. If we consider lines as blocks and points as objects, then a Finite Projective Plane of order n is a Symmetric Design with parameters $(n^2 + n + 1, n + 1, 1)$ Finite Projective Plane of order n has four properties [8]:

1. Given any two distinct points, there is exactly one line incident with both of them.
2. Given any two distinct lines, there is exactly one point incident with both of them.
3. Every point has $n+1$ line through it.
4. Every line contains $n+1$ point.

(Note that some of these properties are redundant.) A projective plane is therefore a symmetric $(n^2 + n + 1, n+1, 1)$ block design.

A finite projective plane [8] exists when the order n is a power of a prime, i.e., for $n = p^1$. It is conjectured that these are the only possible projective planes, but proving this remains one of the most important unsolved problems in combinatorics. The smallest finite projective plane is of order $n = 2$, consists of the configuration known as the *Fano plane*. This *Fano plane*, is denoted *PG (2, 2)*.

It has incidence matrix

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Fig 2.2: Incidence matrix of *Fano Plane PG (2, 2)*

Every row and column contains three *1's*, and any pair of rows /columns has a single *1* in common. The idea will be clearer if we see graphically.

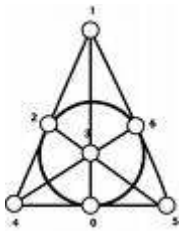

Fig 2.3: Fano plane

From the above figure we see that

1. Given any two distinct points, there is exactly one line incident with both of them.
2. Given any two distinct lines, there is exactly one point incident with both of them.
3. Every point has *3* line through where $n = 2$.
4. Every line incidence with *3* points as $n = 2$.

2.1.3 TRANSVERSAL DESIGN

A transversal design *TD (k, n) [$k \geq 2$ and $n \geq 1$]* is a triple (*X, G, B*) such that the following properties are satisfied:

1. *X* is a set of *k.n* elements called points,
2. *G* is a partition of *X* into *k* subsets of size *n* called groups,
3. *B* is a set of *k-subsets* of *X* called blocks,
4. Any group and any block contain exactly one common point, and
5. Every pair of points from distinct groups is contained in exactly one block.

Note that the "groups" in a transversal design are just subsets of points; they are not algebraic groups. Also, a *TD (2, n)* exists trivially for all integers $n \geq 1$. Later we will give one easy method describing construction method of a Transversal Design with example.

2.1.3.1 AN ALGORITHM TO CONSTRUCT CLASS OF TRANSVERSAL DESIGNS

Suppose p is a prime number and $2 \leq k \leq p$. Then there exist a *TD(k, p)*.

Step 1: Define $X = \{0, \ldots, k-1\} \times Z_p$

Step 2: For $0 \leq x \leq k-1$, define $H_x = \{x\} \times Z_p$.

Step 3: Define $H = \{H_x \mid 0 \leq x \leq k-1\}$.

Step 4: For every ordered pair $(i, j) \in Z_p \times Z_p$,

Define a block $A_{i,j} = \{(x, i*x + j \bmod p) \mid 0 \leq x \leq k-1\}$.

Step 5: $A = \{A_{i,j} \mid (i, j) \in Z_p \times Z_p\}$

Step 6: (X, H, A) is a *TD (k, p)*.

Fig 2.4: Algorithm of Transversal Design

Example: To understand the above algorithm, we present following example how the blocks are formed. In this example a *prime power p = 3, block size k = 3,* which is less than *p* and greater than *2, No. of blocks = 49*. Now the blocks are constructed as below:

Block Id: (0, 0): (0, 0) (1, 0) (2, 0)
Block Id: (0, 1): (0, 1) (1, 1) (2, 1)
Block Id: (0, 2): (0, 2) (1, 2) (2, 2)
Block Id: (1, 0): (0, 0) (1, 1) (2, 2)
Block Id: (1, 1): (0, 1) (1, 2) (2, 0)
Block Id: (1, 2): (0, 2) (1, 0) (2, 1)

Block Id: (2, 0): (0, 0) (1, 2) (2, 1)

Block Id: (2, 1): (0, 1) (1, 0) (2, 2)

Block Id: (2, 2): (0, 2) (1, 1) (2, 0)

Here each block represents the key ids of a particular node in a sensor network. The numbers in the blocks represents the set of key ids. Here, the common key between blocks *(0, 1)* and *(2, 2)* is *(1, 1)*. Transversal Design has the property that maximum number of shared key between two nodes is *1*. Therefore, it may happen that there is no shared key between two nodes. For example, nodes *(0, 0)* and *(0, 1)* do not share any key. In this case if they want to communicate, they have to find one or more intermediate nodes. In the above situation block *(1, 0)* has the common key with both the blocks. Therefore, block *(0, 0)* and *(0, 1)* have to communicate through block *(1,0)* using key (0, 0) and (1, 1) respectively.

Chapter 3

Key Pre-distribution Schemes

Chapter 3

KEY PRE-DISTRIBUTION SCHEMES

3.1 INTRODUCTION

Secure communication between sensor nodes in Wireless Sensor Networks has become an important issue nowadays. Since the sensor devices are severally constrained by their computation, communication and storage constraints, it is not a trivial task. Many schemes like public key cryptography, single key for whole network etc have proposed. However, these schemes do not provide the trade off between the requirement of resource constraints and resiliency. Key Pre-Distribution in Wireless Sensor Network is an optimum scheme for balancing between resource constraints and resiliency of Wireless Sensor Network.

Many Key Pre-Distribution Schemes [2] have been proposed. We name a few of them. The various Key Pre-Distribution Schemes are classified into the following categories.

(i) Eschenauer and Gligor's scheme

(ii) q-composite scheme

(iii) Camtepe and Yener's scheme

(iv) Lee and Stinson's scheme

(v) Chakrabarti, Maitra and Roy's scheme

In the following section, we are briefly elaborating the above schemes.

3.2 ESCHENAUER AND GLIGOR'S SCHEME

Eschenauer and Gligor [9] proposed the basics scheme for Key pre-Distribution in Wireless Sensor Network. They define three parameters (*n, k, p*), where *n* is total number of node in the network, *k* is size of each key ring and *p* (*denoted as v*) is size of the key pool. A different random *k-subset* of keys from *p* are selected and assigned to each node. If there is at least one common key between two nodes, they can securely communicate with each other. If there is more than one common key between a pair of node they can select any of the common key and use as the pair wise key for secure communication. There is a condition which relating the three parameters *n, k, and v* is $n \leq \binom{v}{k}$. Practically it has seen that for reasonable values of *n, k, and v*, the number of nodes can be very large.

3.3 Q-COMPOSITE SCHEME

This scheme was proposed by Chan et al. [2003], who stipulated that two nodes compute a pair wise key only if they share at least q common keys. The integer q is a pre specified *threshold* value which is the number of *intersection* points between two nodes. Given that two nodes have at least q intersection points, they use all their common keys to compute their pair wise key, by means of the following key derivation function:

Two nodes within wireless communication range must be able to determine the common points in the two blocks assigned to them. It is typically suggested that a node U_i would broadcast the k points in A_i to each of its neighbors. This would allow each neighbor U_j of U_i to determine the common keys it shares with U_i by searching two lists of k points (namely, Ai and $A\ j$) for common key identifiers. Now, suppose that two sensor nodes, say Ui and Uj, discover that Ai and $A\ j$ have exactly t common points, say $\{x_{a1}, \ldots, x_{at}\} \in X$, where $a_1 < a_2 < \cdots < a_t$. If the number of common points is at least q, i.e., if $t \geq q$, then they can establish a secret key,

$$K_{i,j} = h(L_{a1} \,||\, \ldots \,||\, L_{at} \,||\, i \,||\, j),$$

using a public *key derivation function h*, this has appropriate input and output sizes. Such key derivation functions are typically constructed from a suitable public hash function, (e.g., SHA-1). The case $q = 1$ is very similar to the basic scheme. The only difference occurs when two nodes share more than one common key. In a 1-composite scheme, all the common keys are used to derive the pair wise key, while in the basic scheme, any common key can be used as a pair wise key.

3.4 CAMTEPE AND YENER'S SCHEME

Camtepe and Yener's [3] first proposed the application of combinatorial design for Key Pre-Distribution to the Wireless Sensor Networks in 2004. Using Combinatorial Design they first time use set system for Key Pre-Distribution. They proposed two classes of combinatorial designs. One is *Symmetric Balanced Incomplete Block Design* or *SBIBD* (In particular finite projective planes) and other is generalized quadrangles. A sensor node associated with a block, says B, receives all the keys indexed by the points contained in B. It is possible to achieve good connectivity and resiliency by using these kinds of set systems. This scheme provides good resiliency under adverse situation. The limitation of this approach is that the network size is limited by the number of blocks in the set system. For this reason, they also presented a *hybrid approach*, where random key

rings are chosen once all the blocks in the set system are exhausted. This helps to obtain larger network sizes.

3.5 LEE AND STINSON'S APPROACH

Distribution of keys into different nodes is according to a transversal design which is a class of combinatorial design. Lee and Stinsonm [4] have brought the concept of Transversal Design for Key Pre-Distribution in Wireless Sensor Networks. This design scheme is equivalent to (v, b, r, k)-design scheme [7], where $bk = vr$ and $v - 1 > r(k - 1)$. Each blocks contain k number of keys, each key occurs in exactly r number of blocks, $b = r^2$ is the total number of blocks and v is the total number of keys in the key pool. Each pair in the arrangement has *0* or *1* shared key. Since they have used Transversal Design for distribution of keys into various nodes, there is a pattern in key ids in each node. This is because Transversal design yields a deterministic fashion of key distribution. Therefore, during common key establishment, no communications require. They have used one example of the design (*1470, 2401, 49, 30*). The important parameters yields

p1 = 0.6.

There is *60%* pair which can communicate directly and

The remaining *40%* node can communicate via intermediate nodes.

Assuming, the deployment of the network is random, it has seen according to the given example that the expected proportion such that two nodes are able to communicate either directly or through an intermediate node is as high as *0.99995*. If any attacker compromise one or more number of nodes, then all the keys within these nodes will get compromised. Therefore, compromising of *s* number of nodes will affect a given link with probability roughly equals to *fail (s) = 1- (1-(r-2)/b-2).* For the above example, *fail (10) = 0.17951.* Therefore any given link is affected with a probability of about *18%* when *10* random nodes are compromised.

3.6 CHAKRABARTI, ROY AND MAITRA'S SCHEME

Lee Stinson's approach is clearly showing that the connectivity between a pair directly or through an intermediate node is high is 0.99995 i.e., almost 100%. However the probability of failure during and adverse condition is 18% when number of compromise node is only 10. This is because the common key between a pair of node is maximum 1. Chakrabarti, Roy and Maitra [1] follows the same direction of key

distribution. However, instead of immediately considering the formed blocks after distribution as sensor node, they use the concept of merging of blocks to form a sensor node. Initially they do not specify any merging strategy and consider that blocks will be merged randomly avoiding intra-node common keys as much as possible. They revised their scheme by introducing *MOVE* function [1] to increase the connectivity between different pairs. In this direction they have presented the following result as an example

Example

Consider a ($v = kr$, $b = r^2$, $r=101$, $k=32$) configuration. If one merges $z = 4$ many blocks to construct a node, the following is obtained (theorem 3 in [1]).

There will be $\left\lfloor \frac{10201}{4} \right\rfloor = 2550$ sensor nodes.

The probability that two nodes do not share a common key is approximately $\left(1-\frac{32}{102}\right)^{16} = 0.0024$.

Expected number of keys shared between two nodes $= \frac{16.32}{102} \geq 5.$

Each node will contain on an average $\hat{M} = 4\times 32 - \binom{4}{2}\frac{32}{102} \approx 126$ many distinct keys and at most 128 many keys.

$Fail(10) = 0.019153 \approx 2\%$ and $Fail(25) = 0.066704 \approx 7\%.$

The example clearly uses more keys (≤ 128) per sensor node. Note that directly from (v, b, r, k) configuration, it is not possible to have $k > r$. However, in a merged system that is possible. Moreover, the average number of keys shared between any two nodes is ≈ 5.

Chapter 4

Deterministic Merging of Blocks for KPS in WSN

Chapter 4 DETERMINISTIC MERGING OF BLOCKS FOR KPS IN WSN

4.1 INTRODUCTION

Application of combinatorial design [3] offers a balance of key content in various sensor nodes. The maximum number of pair of nodes can communicate directly using pair wise common key. Transversal Design is such a combinatorial Design which offers a deterministic nature of key distribution [4]. A pattern of key ids can be seen in this type of distribution of keys. Lee and Stinson first time proposed the application of Transversal Design for Key Pre-Distribution in Wireless Sensor Network [2]. The result is less communication including a balance distribution for the establishment of secure communication. However, Transversal Design yields maximum *1* pair wise key is possible for a pair of node. Therefore, compromise of a node yields compromise of all the keys underlying that node and so that compromise of all the links carried by those keys. Here, as the number of common key between a pair is maximum one, the compromise of that key on compromise of any other node having the same key yields breaking of the link between the pair under consideration. This clearly shows that the resiliency in adverse condition is less due to the fact that the number common key between a pair of node is maximum one. Here the resiliency is the protection measure of a particular design. This is measured by the parameter *fail(s): fraction of links become compromise on compromise of randomly selected s number of node* [6]. Chakrabarti, Roy, and Maitra has modified this scheme and proposed that instead of immediately considering each blocks as sensor node after distribution of keys using Transversal Design, merging of a number of blocks to form a node yielding the probability of more than one common key between a pair of nodes. Therefore, during any adverse condition the probability of breaking link between a pair of node becomes very less. However, it increases a bit of memory requirement which can be accommodated [1]. Moreover this scheme increases the resiliency. Selection of blocks for merging to form a node is purely random. Due to this randomness, the content of blocks in a node is random i.e. unpredictable. During common key establishment between a pair of node introduces an amount of communication cost *O(z),* if number of blocks in each node is *z.* We modified this part and proposed a deterministic scheme [22]. In order to select blocks for merging to form a node we follow a particular rule. As the selection of block is deterministic,

there is a pattern of blocks in each node. Therefore, to find out blocks for a particular node, no extra communication require during common key establishment. We have simulated and analyze the various parameters as stated earlier in the following sections. For simulation we have used C Language as the platform.

4.2 SIMULATION AND STUDY OF LEE AND STINSON'S SCHEME

We have simulated the scheme provided by Lee and Stinson. For simulation we have used C Language for programming. We have studied some important parameters like F*(s)* [6]: Fraction of links which has been compromised due to the compromise of s number of nodes. The results we obtain

Use (*v, b, r, k*) based *transversal design*, where *v = 3232, b = 10201, r = 101, k = 32.*

Average number of common keys between two nodes = 1.000000.

Maximum number of connection could be *104050200*.

Number of initial links detected = *16070800.*

Therefore *connectivity of the design* is *0.154452*, i.e. almost *15%.*

The average value of *f(s) = 0.3259*, i.e. almost *33%* where *s = 40*.

	S = 4	S = 8	S = 12	S = 16	S = 20	S = 24	S = 28	S = 32	S = 36	S = 40
f(s)	0.0391	0.0765	0.1117	0.1494	0.1800	0.2121	0.2464	0.2721	0.3020	0.3259

Table 4.1: Result of E(s) for Lee and Stinson's scheme

4.3 SIMULATION AND STUDY OF CHAKRABARTI, ROY AND MAITRA'S SCHEME

According to Lee and Stinson's scheme, any pair of nodes can share *0 or 1 key* [4]. Merging of nodes to form a new node increases the number of common keys between a pair. Chakrabarti, Roy, and Maitra provide one scheme where they randomly choose z number of blocks and merged to form a new node. They have chosen the blocks in such a way that there will be no inter node connectivity. As they have chosen randomly, for some cases they could not avoid the occurrence of inter node connectivity. After forming a number of nodes they revised their scheme by introducing MOVE function [1]. MOVE is basically increasing the connectivity of the network by swapping blocks between maximum linked pair with zero linked pair.

They define MOVE function is as follows.

1. Start **move**;
2. Copy the current configuration in a temporary configuration and work on the temporary configuration;
3. From the list of pairs of nodes sharing more than one common keys, select one pair of nodes randomly; call them *a* and *b*;
4. From the list of pairs of nodes sharing no common key, select one pair of nodes randomly; call them *c* and *d*.
5. Select one block each from *a* and *b* (say block α from node *a* and block β from node *b*) and remove them such that α and β intersect each other and nodes *a* and *b* are still connected after the removal of α, β, respectively; if this condition is not satisfied then go to step 9;
6. Select one block each from nodes *c* and *d* and remove them; let the removed blocks be γ and δ, respectively;
7. Put γ in *a*, δ in *b*, α in *c* and β in *d*;
8. Store this temporary configuration in some container;
9. End **move**.

Fig 4.1: MOVE function

The MOVE algorithm is just exchanging blocks between two pairs such that one pair is having maximum common intersection point and another pair has no intersection point at all. Now one block from each node selected randomly from the first pair such that there is an intersection between selected blocks and ensures that the removal of those blocks still left one or more common intersection point. These two blocks are exchanged with the blocks selected randomly from the second pair. Thus second pair gets a common intersection point and so that common key and first although looses a common point still able to have at least one common point so that common key. Hence MOVE is executed hundred times to balance the design which helps to increase the connectivity and Resiliency as well. We have simulated this scheme using *C language* and studied the parameters *E(S): Fraction of links get compromise on compromise of s number of nodes*

[6], and *Average number of common keys between a pair*. The experiment result shows that the *resiliency* is much *higher* than the scheme provides by Lee and Stinson. But to store keys for each nodes need more storage. However, they have shown that consumed storage space is within the limits of a sensor node. The results we have obtained for various parameters are

Average number of common keys between a pair is *5.019608.*

Maximum number of connection could be *3249975*.

Number of initial links detected *is 3242106*.

Therefore, *connectivity of the design* is *0.997579*, i.e. almost *100%.*

The average value of $E(s) = 0.0193$, i.e. almost *2%*, where $s = 10$ and equivalent to *40 blocks*.

	S = 1	S = 2	S = 3	S = 4	S = 5	S = 6	S = 7	S = 8	S = 9	S = 10
E(s)	0.0007	0.0017	0.0027	0.0042	0.0059	0.0080	0.0103	0.0130	0.0162	0.0193

Table 4.2: Result of E(s) for Chakrabarti, Ray and Matra's scheme

4.4 DETERMINSTIC MERGING OF BLOCKS FOR KEY PRE-DISTRIBUTION IN WSN

4.4.1 KEY DISTRIBUTION

Chakrabarty, Roy and Maitra's scheme improves some parameters. However, we observe that they have used random scheme for selection of blocks to merge for forming node. Therefore, a particular node will be having no specific block id. On the time of shared key discovery between a pair of nodes, they have to broadcast all the block ids to the other nodes. This is yielding a communication cost *O(z)*[1], (*z* is the number of blocks to be merged to form a node) in addition to the request for communication which is *O(1)*. Sending all the block ids can not be avoided due to the randomness of the scheme. Observing this limitation, we propose a deterministic scheme for merging of block to form a node.

The property of Transversal design for arrangement of a set of elements into a number of subsets focuses the fact that the probability of repeating an element for consecutive blocks is much less. Therefore, merging $z\ (1 \leq z \geq p)$ number of blocks to form a node implies much less probability for occurrence of intra-node repetition of

same element. On the basis of this assumption, we have considered *z* number of consecutive blocks for merging to form a node which helps to avoid any intra-node common key. This increases the connectivity of the entire network as well. Again as we merge *z* number of consecutive blocks, there is a pattern of block ids in a particular node. Therefore, to find out block ids for a particular node id we does not need to exchange block id which consumes an amount of communication effort. Nodes can themselves compute block ids of their counterparts. As our scheme is a deterministic, the communication cost is only *O(1)*, that needs to request for communication by any of the node in the pair, which is much less than *O(z)*. Note that the communication cost in our scheme is a constant value in comparison with scheme by Chakrabarti, Roy and Maitra where communication cost is a variable figure. On getting the node id of the requesting node, a node can easily determine the block ids of the other node which will take *O(z)* cost for computation time in average. After obtaining the block ids rest is to discover the shared keys, would take $O(z^2 log_2^2 r)$[8] time. Therefore, average computation cost for key establishment is $O(z) + O(z^2 log_2^2 r)$, i.e. $O(z^2 log_2^2 r)$, which is same as the scheme proposed by Chakrabarti, Roy, and Maitra. However, communication cost is much less which is one of the key requirements for these computational intensive devices. The algorithm for merging nodes is as follows:

```
/* Input: A set of block ids after conversion into one dimensional
   form
   Output: A set of node ids */
Begin of Merging blocks
    Count = 0;
    For i = 0 to bnumber-1do
    /* bnumber is the total number of blocks */
    Begin
        For j = 0 to z-1 do
        /* z is the number of blocks to be merged */
```

Continue……

```
Begin
  For s = 0 to k-1 do
  /* k is the number of keys stored by each block */
  Begin
    Noderepository[i][j*k+s].first = block[count][s].first;
     /*Storing first part of the key id*/
    Noderepository[i][j*k+s].second
    = block[count][s].second;
    /*Storing second part of the key id*/
   End For
 End For
 Count++;
End For
End of Merging blocks
```

Fig 4.2: Algorithm for merging blocks in deterministic scheme

The experimented result we have obtained using the design ($v = 3232, b = 10201, r = 101, k = 32$) and $z = 4$, is given below.

The total number of nodes which has formed is *2550 e*ach having *128* number of keys.

Average number of common keys between two nodes = 5.520681.

Maximum number of connection could be *3249975.*

Number of initial links detected 2955867.

Therefore, *connectivity of the design* is *0.909505,* i.e. almost *91%.*

The average value of $E(s) = 0.1562$, i.e. almost *16%*, where $s = 10$ and equivalent to *40 blocks*.

	S = 1	S = 2	S = 3	S = 4	S = 5	S = 6	S = 7	S = 8	S = 9	S = 10
E(s)	0.0080	0.0179	0.0315	0.0443	0.0586	0.0807	0.0976	0.1141	0.1413	0.1562

Table 4.3: Result of *E(s)* for proposed scheme

4.4.2 KEY EXCHANGE

Any pair wishes to communicate with each other send a request massage to its counterpart, which then including the sender discovers the common key between them. According to our scheme, they don't need to send any extra information. They generate the block ids of the others using the following algorithm which needs the node id only of the other node.

```
/* Input: Node id
   Output: z number of block ids */
   Begin of block discovery
   For i = 0 to z do
   /* Number of blocks have been merged for each node */
    Begin
       Block[i].first = (Node id * z + i) / (p – 1);
       /*p is the prime number taken at the time of transversal
         design where total number of blocks = p * p */
       Block[i].second = (Node id * z + i) % (p + 1);
   End for
End of block discovery
```

Fig 4.3: Algorithm to discover block ids for a given node

On discovering the block ids, they can compare all the blocks with their own blocks for finding any common key using the algorithm proposed by Lee and Stinson. After discovering the common key, if any, they can start communication using that key. In case of a pair which does not have any common key, they have to generate a key temporarily and need to exchange through one or more intermediate nodes.

4.5 COMPARISON

If we compare between previously discussed three schemes we can find that the scheme proposed by us being a deterministic merging technique offers much less communication cost i.e. *O(1)*, which is one of the key requirements of Wireless Sensor Network. However scheme proposed by Chakrabarti, Roy, and Maitra consumes a

variable communication cost *O(z),* where z is the number blocks to be merge to form a node. Again, although the scheme proposed by Lee and Stinson consumes *O(1)* as the communication cost, however, it suffers from less Resiliency. Computation for key discovery is same i.e. $O(z^2 log_2^2 r)$[8] in our scheme as well as for Chakrabarti, Roy and Maitra and Lee & Stinson. The average number of common keys in each pair of node is almost *5* in our scheme as well as in Chakrabarti, Roy, and Maitra's scheme [1], whereas scheme proposed by Lee and Stinson[5] has only *1* key. This is the main advantage of merging blocks to form node. Connectivity of ours scheme is almost *91%* which is almost same with the scheme proposed by Chakrabarti, Roy and Maitra. However, connectivity of our scheme is much better than the scheme proposed by Lee & Stinson. The resiliency is best in Chakrabarti, Roy and Maitra's scheme. However, our scheme has also achieved a reasonable efficiency for this parameter.

4.6 CONCLUSION

The comparative study and simulation of the scheme proposed by Lee and Stinson and the scheme proposed by Chkrabarti, Roy and Maitra shows that merging of blocks to form node improves a number of parameters such as *Resiliency*, *average number of common key between a pair of node*, *connectivity of the network* etc. Although there is a limitation that the memory consumption is more, but it has seen that this requirement for merging is easily adaptable [1]. However, the scheme proposed by Chakrabarti, Roy and Maitra, as they merge randomly, the communication cost for key discovery is *O(z)*, which we have tried to minimize by proposing a deterministic scheme. This is one of the *major requirements for a wireless sensor network*. More improve can be done by revising the merging strategy for obtaining better resiliency.

Chapter 5

Elliptic Curve and Pairing

Chapter 5

ELLIPTIC CURVE AND PAIRING

5.1 INTRODUCTION

The pairing is a function which is a product of two subgroups over elliptic curve maps to a finite multiplicative group [18]. One of the first well known applications of cryptographic pairings is the transformation of an elliptic curve discrete logarithm problem (ECDLP) instance into an instance of discrete logarithm problem (DLP) in the finite field. The two types of pairing over elliptic curve is known commonly as Tate pairing and Weil pairing. Weil pairing was introduced to cryptography by Menezes, Okamoto and Vanstone and the Tate pairing was introduced by Frey and R¨uck [18], called F R reduction, and is about twice faster than Weil pairing.

These applications were examples of a "negative" use of pairings in ECC. Shamir for example in 1984 posed as a challenge the concept of an Identity-Based Encryption scheme. Boneh and Franklin [16] proposed a solution to this challenge based on a bilinear map using the Weil pairing. Many more state of-the-art protocols using bilinear pairings were proposed such as short digital signatures. The best known method for computing pairings is based on Miller's algorithm [21]. This is a standard method and many researchers have been trying to improve its efficiency. The emphasis mainly has been to optimize the Miller's loop and the final exponentiation in the algorithm [21].

5.2 ELLIPTIC CURVE

Elliptic curves are considered primarily as an alternative group structure, with certain advantages when it comes to the implementation of common cryptographic protocols [17]. The main advantage is that much smaller keys can be used, as there is no known polynomial-time algorithm for the discrete logarithm problem for the great majority of such curves. Given a point P on a curve E defined over a finite field F_q where $q = p^m$ (where p is a large prime) this is the problem of determining a given aP. In most circumstances the points on such a curve form a simple cyclic group. Each point on the curve has an order. This is the smallest positive integer r such that $rP = O$, where O is the identity point of the group, the so-called point at infinity. The number of points on the curve, the order of the curve, is referred to as $\#E$. Every valid r divides $\#E$. Here is an example of elliptic curve [17].

$$E(F_q) : y^2 = x^3+Ax+B$$

The complete set of curve points is called *G*, of order *#E*. The set of all points that are transformed to *O* by multiplication by *r ("killed by r")* is called *G[r]*. These are the *r-torsion* points. Since *r* is a prime, this is all the points of order *r* plus *O*. There are r^2 such points, and these r^2 points can be organized as *r+1* distinct cyclic subgroups of order *r*. They all share *O*. Note that one of these subgroups is *S[r]* and consists of all those *r-torsion* points from the original curve $E(F_q)$ - points of the form *Q [(a, 0), (c, 0)]*, which are of course on both curves. Let $h = \#E/r^2$. Then a random point on the curve can be mapped to a point in one of these sub-groups of order *r* by multiplying it by this co-factor *h*. For simplicity we assume that *r* does not divide *h*. For our example curve *r = 11* and *h = 140*. The set of distinct points generated by multiplying every element of *G* by *r* is called *rG*. The number of elements in *rG* is *h*. This is called a coset. Consider the partitioning of the *#E* points into distinct cosets. This can be done by adding a random point *R* to every element of *rG*. There are exactly r^2 such distinct cosets, each with h elements. The original coset *rG* is the unique coset that contains *O*. Every coset contains exactly one *r-torsion* point. Elements of these cosets are not all of the same order. They do not form a group. The quotient group $G/r \in G$ is the group formed of all these cosets.

5.3 TATE PAIRING

The Tate Pairing [17, 19, 21] operates on a pair of points, *P* of prime order *r* (a member of *G[r]*) and a point *Q* which is a representative member of one of the cosets. It is denoted $e_r(P,Q)$. It evaluates as an element of the finite field F_{p^2} of order *r*. Observe that *r* divides p^2-1. Its value is the same irrespective of which element of a particular coset is chosen. Recall that each coset has exactly one r-torsion point. For convenience, [17] choose *P* to be a member of *S[r]* - as it also lies on $E(F_p)$, this makes the Tate Pairing calculation much faster. However the Tate pairing can evaluate as *1*. This will occur if *P* is a multiple of *Q*, which is the case if *Q* is chosen from a coset whose *r-torsion* point is also a member of *S[r]*. For a randomly chosen *Q* and for large *r* this is extremely unlikely - the odds are *1/r*. The Tate Pairing is non-degenerate as for any given *P* not equal to *O*, we can always find a *Q* such that $e_r(P,Q)$ is not *1*. Also $e_r(P,P) = 1$ for *P* in *S[r]* (and $k > 1$). However, the most important property of the Tate pairing is bilinearity.

$$e_r(aP, bQ) = e_r(P,Q)^{ab}$$

Note that *P* must be of order *r*, but *Q* need not be.

Which coset to choose *Q* from? There are computational advantages in choosing points of the form *Q [(a, 0), (0, d)]*. Call the set of points of this form *T*. It is not difficult to see that if there are *p+1-t* points of the form *Q [(a, 0), (c, 0)]* then there will be *p+1+t* points of the form *Q [(a, 0), (0, d)]*. Substitute all *a<p* for *x* in the curve equation. Then if the RHS is a *QR*, the point is *Q [(a, 0), (±c, 0)],* otherwise it is *Q [(a, 0), (0, ±d)]*. There will always be a subgroup of order *r*, consisting of points of this form. *Q* can therefore be chosen as an element of *T*. Note that points of this form stay in this form under point multiplication, so such a *Q* will be in a coset supported by an element of *T[r]*. However, there are also *p+1+t* points on the twisted curve. Is there a connection between the group of points of the form *Q [(a, 0), (0, d)]* and the group of points on the "twisted" curve? Yes there is - they are isomorphic. For every point of the form *Q [(a, 0), (0, d)]* on the curve defined over the quadratic extension field F_{p^2}, there is a point *Q (-a, d)* on the twisted curve defined over F_p. This is convenient as it means that multiplication of such points can be done on the twisted curve using regular *E* (F_p) methods. A diagram [17] might help.

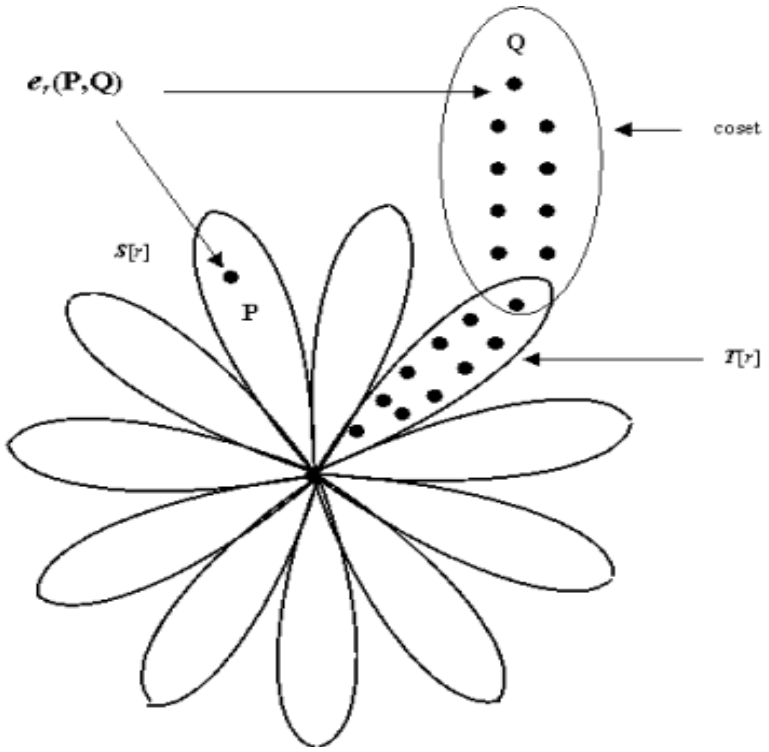

Fig 5.1: Trace Map (Adopted from [17])

The point-at-infinity is in the centre. Twelve subgroups of order *11* radiate out from it. Each of the points in each subgroup supports a coset. The point-at-infinity supports the coset *rG.* An alternative idea is to use a super singular curve. For example

$$E\ (F_p)\colon y^2 = x^3 + x$$

with *p =131, r =11, t=0, P(6,22), #E=132.* There are points on the curve of order *132*, and the group is cyclic. There is a subgroup of order *r*. This same curve taken over the extension field $E\ (F_{p^2})$ has *17424 (=132.132)* points on it. As before, there are no points on the curve of this order - it is not cyclic. In this case the sets *S* and *T* are of the same order. But more than that - every point in *S* can be mapped directly to a point in *T* of the same order via the auto-morphism *f (x, y) = (-x, 0), (0, y)*. For every point *Q [(a, 0), (c, 0)]* in *S*, there is a point *Q [(-a, 0), (0, c)]* in *T*. This allows the introduction of the alternative function $\hat{e}_r\ (P, Q) = e_r\ (P, f\ (Q))$, where *P* is a member of *S[r]* and *Q* is a member of *S*. Note that $\hat{e}_r\ (P, P)$ is not *1*.

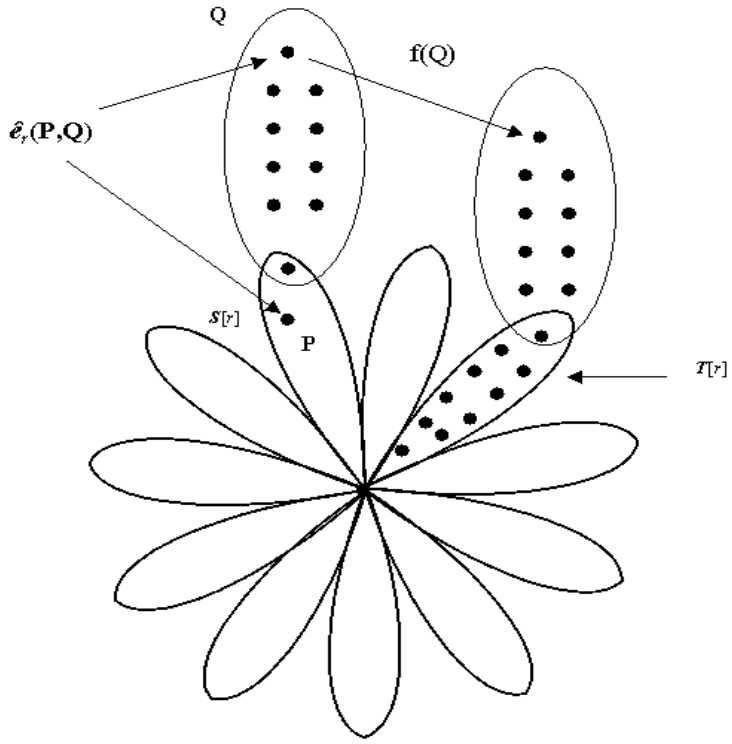

Fig 5.2: Trace Map for Alternative curve equation (Adopted from [17])

What about all those other subgroups of order *r?* In fact they are of little interest. Any general point *P [x, y] = P [(a, b), (c, d)]* of order r on the curve can be written as the

sum of a point from *S* and a point from *T* using the Trace Map. $P[x, y] = P_S + P_T$. Where $P_S = Trace\ (P)/k$, and $P_T = P - P_S$. In case of [17], $P_S = ((a + ib, c + id) + (a – ib, c - id)) / 2$ (an elliptic curve point addition followed by elliptic curve point division by 2). For a general point $P\ [x, y] = P\ [(a, b), (c, d)]$ of order *r*, $e_r\ (P, P) = e_r\ (P_S, P_T)$. $e_r\ (P_T, P_S)$ which is NOT equal to *1*.

The Tate pairing can be computed using Miller's algorithm [21]. We show Miller's algorithm as follows.

1. Choose a random point $Q' \in E(F_{q^k})$ and computes $S = Q + Q' \in E(F_{q^k})$
2. Set $t = \lfloor \log_2(m) \rfloor$, and let $(m_t, \ldots, m_0)_2$ be the binary representation of m.
   Set $f = 1$ and $V = P$
3. For $i = t - 1$ to 0 do
   Set $f = f^2(gV, V(S)g^2V(Q'))/(g^2V(S)gV, V(Q'))$ and $V = 2V$
   If $m_i = 1$ then
   Set $f = f(gV, P(S)gV + P(Q'))/(gV + P(S)gV, P(Q'))$
   and $V = V + P$
4. Return f

Fig 5.3: Miller's Algorithm for Tate pairing

The trace is the value *t* satisfying $\#E(F_q) = q + 1 - t$. The elliptic curve is said to be supersinglur curve if the characteristic *p* of F_q divide *t*. According to Washington, $E(F_q)$: $y^2 = x^3 + ax$ is supersinglur curve and $\#E(Fq) = p + 1$ if q = p > 3, p = 3 mod 4. And if $E(F_q)(q = p^k)$ is supersinglur curve $E(F_q) = p^k + 1\ (k : odd)$, $E(F_q) = ((-p)^{\frac{1}{2}} - 1)^2 (k : even)$. A distortion map with respect to *P* is an endomorphism φ that maps the point *P* to a point $\varphi(P)$ that is linearly independent from *P*. The image $\varphi(P)$ of the point *P* will have coordinate in an extension field of F_q.

5.4 ηT PAIRING

Miller proposed in 1986 the first algorithm for computing Weil and Tate pairings [24]. Barreto et al. introduced the ηT pairing, which extended and improved the Duursma-Lee techniques. It makes it possible to efficiently compute the Tate pairing. An iterative implementation of the ηT pairing is given in [24] over the underlying field F_3^m. The study of ηT pairing in [24] shows the efficiency in comparison to Tate pairing.

5.5 IDENTITY BASED NON-INTERACTIVE KEY DISTRIBUTION (ID-NIKD)

5.5.1 INTRODUCTION

In 1984 Shamir asked for a public key encryption scheme in which the public key can be an arbitrary String. Shamir's original motivation for identity-based encryption was to simplify certificate management in e-mail systems [16]. When Alice sends mail to Bob at bob@company.com she simply encrypts her message using the public key string bob@company.com". There is no need for Alice to obtain Bob's public key certificate. When Bob receives the encrypted mail he contacts a third party, which we call the Private Key Generator (PKG). Bob authenticates himself to the PKG in the same way he would authenticate himself to a CA and obtains his private key from the PKG. Bob can then read his e-mail. Note that unlike the existing secure e-mail infrastructure, Alice can send encrypted mail to Bob even if Bob has not yet setup his public key certificate. Also note that key escrow is inherent in identity-based e-mail systems: the PKG knows Bob's private key.

5.5.2 NON-INTERACTIVE KEY DISTRIBUTION (ID-NIKD)

Tate pairing [14, 15, 18, 19, 20] is a tool which can be applicable to establish Identity based Non-Interactive Key Distribution [16]. The computation of pairing involves two groups, G_1 and G_2 which are finite cyclic additively-written groups and at least one of which is of prime order r. The pairing takes an element from each of the two groups and maps them to the third group G_T, which is a finite cyclic multiplicatively written group also of prime order r. The properties which make the pairing applicable [16, 20, 15] are as follows

Bilinearity: For all $P, P' \in G_1$ and all $Q, Q' \in G_2$, one has: $t(P + P', Q) = t(P, Q) \times (P', Q)$ and $t(P, Q + Q') = t(P, Q) \times t(P, Q')$.

Non-degeneracy: For all $P \in G_1$ with $P \neq 0$, there is some $Q \in G_2$ such that $t(P,Q) \neq 1$. For all $Q \in G_2$ with $Q \neq 0$, there is some $P \in G_1$ such that $t(P,Q) \neq 1$.

Computable: t can be easily evaluated.

Identity based non-interactive key distribution involves three algorithms: *Setup, Extract and Shared Key*. An offline Trusted Authority is responsible for algorithm Setup and Extract. The third algorithm i.e. Shared Key is for discovery of shared key between two parties and hence would be under control of two parties involve in secured communication.

The Algorithm SETUP is as follows.

SETUP

The offline Trusted Authority (TA) select

1. The public group G_1 (with generator p) and G_2, size of q depending on security parameter k.
2. The pairing scheme is Tate Pairing t.
3. A random secret r, $s \in Z_q^*$.
4. A master public key P and $K_{pub} = sP$, $u = rP$.
5. A public hash function $H_1: \{0, 1\}^* \rightarrow G_1^*$
6. A public hash function $H_2: G_2 \rightarrow \{0, 1\}^n$ for some fixed n
7. The massage space and the cipher space $M = \{0, 1\}^n$, $C = G_1^* \times \{0, 1\}^n$

Fig 5.4: Algorithm for SETUP

Setup algorithm executes by the Key Generation Centre or Trusted Third Party. Executing this algorithm, the TA sets various parameters for generation of private key along with public key.

EXTRACT

To create private key for $ID \in \{0, 1\}^*$, the TA computes

1. $Q_{ID} = H_1(ID)$ and
2. Private Key $d_{ID} = sQ_{ID}$, which would be loaded into each node.

Fig 5.5: Algorithm for Extract

The TA generates private keys for corresponding ids of each node and these private keys along with master public key K_{pub} are loaded into the respective nodes. During shared key establishment, pair of nodes involve in communication executes the algorithm Shared Key.

SHARED KEY

On input K_{pub}, a private key d_{ID_A}, and an identifier of other node $ID_B \in \{0, 1\}^*$, where $ID_A \neq ID_B$, a Shared key K_{shared} generates according to the constraint $t(k_{pub}, d_{ID_A}, Q_{ID_B}) = t(k_{pub}, d_{ID_B}, Q_{ID_A})$. Where $Q_{ID_A} = H_1(ID_A)$, and $Q_{ID_B} = H_1(ID_B)$.

Fig 5.6: Algorithm for finding shared key

Correctness of the above algorithm [16] lies on the bilinearity property of pairing function t. where $t(k_{pub}, d_{ID_A}, Q_{ID_B})$

$= t(sP, sH_1(ID_A), H_1(ID_B))$

$= t(sP, H_1(ID_A), H_1(ID_B))^s$

$= t(sP, H_1(ID_A), sH_1(ID_B))$

$= t(k_{pub}, d_{ID_B}, H_1(ID_A))$

$= t(k_{pub}, d_{ID_B}, Q_{ID_A})$

This clearly shows that no need of communication between the pair of nodes require for establish a shared key. The pairing function *t* is basically *Tate pairing* which is efficient and it has proved [14] that it can be applicable to computation intensive devices like smartcards. Therefore the above described algorithm can also be applicable to wireless sensor nodes. We have used this tool for path key establishment.

Chapter 6

SECURED PATH KEY ESTABLISHMENT

Chapter 6 SECURED PATH KEY ESTABLISHMENT

6.1 KEY PRE DISTRIBUTION

Before deployment of the network, a set of keys along with key ids are generated. These key ids along with the keys are distributed to the nodes randomly or according to any pre defined rule. Each node may contain keys for every other node i.e. if there is n number of nodes in the network, each node need to contain n-1 number of key. This distribution strategy is best for resiliency. However, this strategy suffers from scalability. This is because sensor nodes are limited with memory requirement. If numbers of nodes in the network get increases, the value of n also is increases. After a certain increase of the number of sensor nodes, it is not possible to keep keys for each node in the network Therefore Scalability can not be guaranteed in this strategy. Again public key cryptography requires a huge computation which may not be supported by sensor devices. The only optimum solution is Key Pre-Distribution which provides trade off between resiliency and resource limitation. According to this strategy, a set of key has been chosen and unique key ids are assigned to each key. A fixed number of key ids for each node are taken from previously chosen set and distributed to each node randomly or by any pre defined rules. Therefore increase of node in the network does not indicate the increase of key ids in any node and hence the scheme is scalable. However, each node may not have shared key with the other nodes. There would be some pair of nodes which does not have any common key for secure communication. We are describing in the next section how to deal with that situation.

6.2 PATH KEY ESTABLISHMENT

Pair-wise Key establishment process establish common key between a pair of nodes and then the pair can communicate with each other securely using the same common key with the assumption that they both are within the same communication range. They use symmetric key algorithm for encryption or decryption process which is admissible to sensor nodes. However, if there is no common key between a pair of node, they have to communicate through one or more intermediate nodes [5, 11, 12, 13]. The idea is that if there is no common key between a pair of node u and v, a random key generates and a sequence of intermediary nodes $n_1, n_2, \cdots, n_t$ are found such that every pair of adjacent nodes in the path $u, n_1, n_2, \cdots, n_t, v$ share a pair wise key and exist within the communication range with each other and then the newly generated key

propagates through the same path by encrypting and decrypting alternatively until reaching to the destination. Once the common key establish, all communication between u and v occurs using that common key with the assumption that they both are within the same communication range. Therefore, no intermediary nodes require for actual communication.

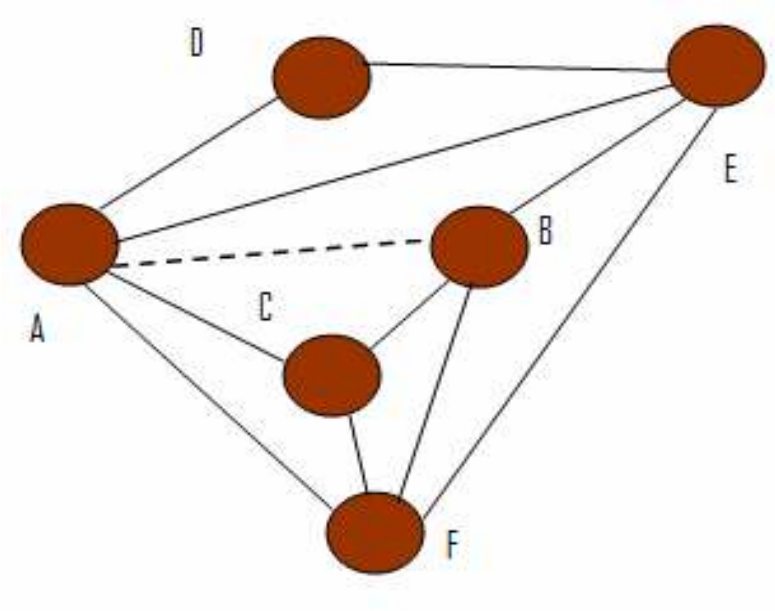

Fig 6.1: Path key establishment

For example, from the above figure let us suppose node *A* want to communicate with node B. We see that there is no shared key between node *A* and B (A solid line between a pair of nodes indicates the existing of common key between them). However, node C is having shared key between both *A* and B. Therefore, *A* generates a key K_{ran}, encrypt it using shared key $K_{AC,}$ and send to *C*. *C* decrypts using K_{AC}, to recover K_{ran}. *C* again encrypts using $K_{CB,}$ and send to *B*, i.e. the intended recipient. Now, node *B* decrypts it using K_{CB} to obtain $K_{ran.}$ Now, they can exchange any data using the key K_{ran} (Dotted line shows the newly generated path). Thus nodes A and B does not need any intermediate nodes for exchange any information.

6.3 SECURITY THREATS IN PATH KEY ESTABLISHMENT

In the process of Path Key establishment, the newly generated key propagates through one or more intermediate nodes generate a channel which is not secured [11, 12, 13]. This is because compromise of any intermediate node results compromise of newly

generated temporary key which would cause the compromise of newly established direct communication between the pair of nodes. In order to enhance the security of symmetric key establishment, Hui Ling and Taieb Znati [12] proposes an End-to-End Pair-wise Key Establishment scheme which leverages multiple paths for key negotiation and establishment However, they have assumed that an attacker can randomly compromise at most *x* out of n nodes. This is clearly limiting the security against any huge attack. This is also vulnerable to stop forwarding or Byzantine attacks. Another scheme based on multiple path scheme proposed by Guanfeng Li, Hui Ling and Taieb Znati [13]. This scheme is also suffers form the same limitation. A new strategy proposed by Jang-Ping and Sheu Jui-Che Cheng introduce [11] hop by hop authentication scheme for path key establishment. From the existing three schemes we observe that an additional communication cost incurs due to multiple selection of path. Moreover, instead of using any short path for each time, the routing would be through one or more long path involving more number of intermediary nodes, this increases communication cost and also consumes an additional amount of computation by extra intermediary nodes. Thus, although a few researchers modified the routing strategy [11, 12, 13] for avoiding attacks on key, However till now no proposal have been made to ensure the full security to this temporarily generated key and hence the newly established communication. We for the first time have proposed two schemes for securing this temporary key.

6.4 IDENTITY BASED PUBLIC KEY CRYPTOGRAPHY FOR PATH KEY ESTABLISHMENT

6.4.1 EXCHENGE OF TEMPORARY KEY

This is our first proposal where we have used identity based public key cryptography for establishing temporary key and hence temporary communication path. According to our scheme, before deployment of a wireless sensor network, the *Key Generation Centre (KGC)* which also acts as a trusted authority for the entire network generates the private key for each node in respect to the node id ID_i, $i = 1, 2, ..., n$ for n number of nodes. For generating private key, *KGC* use the algorithm Setup and Extract (described in section 5.5.2). After generation of various parameters, private keys and master public key along with the other parameters are loaded along with the other keys (distributed randomly or using a particular scheme say combinatorial design) to respective nodes. This happens before deployment of the network. After the deployment

of the network, nodes in the underlying network communicate with each other if necessary. If there is no common key between a pair of nodes, path key establishment takes place. We have used public key cryptography technique for path key establishment for the time being.

The algorithm is follows

1. Node A generates a random key K_{ran}, which is a random key,
2. Encrypt K_{ran} to get C_1
   i) Compute $Q_{ID_B} = H_1(ID_B) \in G_1^*$
   ii) Compute $G_{ID} = t(Q_{ID_B}, K_{pub}) \in G_2$ and
   iii) Set $C_1 = (u, K_{ran} \oplus H_2((G_{ID})^r))$

   Note that K_{pub} is the KGC's public key and thus independent of the recipient's *ID*
3. Node *A* encrypt C_1 to get C_2 using shared key between its nearest neighbor with which it has common key and send to it.
4. After propagation of C_2 through one or more intermediate nodes it will reach to the actual recipient *B*. Node *B* decrypt C_2 using shared key between itself and its nearest neighbor node from which it came. Another decryption for recovering K_{ran} from C_1 using private key d_{ID_B} is

$$K_{ran} = C_1 \oplus H_2(t(d_{ID_B}), u))$$

Fig 6.2: Algorithm for non interactive path key establishment using public key cryptography

The temporary pair-wise key has been exchanged using the above algorithm by two parties. One of them generates the random key K_{ran} which is used temporarily and encrypt to get the corresponding cipher text (unintelligible) C_1. Note that to generate C_1 it uses public key i.e. the id of other node which is ID_B. On the other hand another party after receiving cipher C_1, decrypts K_{ran} by using its private key which has provided by the trusted authority or Key Generation Centre. They both have used ηT pairing as a tool for

pairing the ids as public key for the corresponding private key. The above algorithm generates the accurate replica of temporary key K_{ran} for the receiving node.

This has proved as follows.

$$H_2(t(d_{ID_B}, u))$$

$$= H_2\,(t(sH_1(ID_B), rP))$$

$$= H_2\,(t(H_1(ID_B), P)^{rs})$$

$$= H_2\,(t(H_1(ID_B), sP)^{r})$$

$$= H_2\,(t(H_1(ID_B), K_{pub})^{r})$$

$$= H_2\,(G_{ID})^{r}).$$

Now, ex-oring this component with C_1 will definitely give $K_{ran.}$ This has shown as follows

$$C_1 \oplus H_2(t(d_{ID_B}), u))$$

$$= K_{ran} \oplus H_2((G_{ID})^{r} \oplus H_2((G_{ID})^{r} \quad \text{Since } C_1 = K_{ran} \oplus H_2((G_{ID})^{r}$$

$$= K_{ran}$$

Hence they can start communication directly using new temporary key K_{ran} for the entire session. Thus for a particular session, a random key generates and propagates securely using public key cryptography. The pair of nodes under discussion can either store this temporary key K_{ran} or derive the key from the identities each time they need. This constitutes a memory/computation trade-off. However, computing shared keys requires a pairing computation applying the pairing function $t(\ )$. On the other hand, memory space is very cheap and growing fast, while the pre-shared keys and public keys are fairly short.

6.4.2 SECURITY ANALYSIS

The above scheme is fully secured from the compromise by any third party including intermediary nodes. Any of the intermediate nodes can not be able to decrypt C_1 to get K_{ran} as they neither have master private key s and r nor private key of B. Therefore, this scheme is fully secured from both the external or internal attacks.

6.4.3 DISADVANTAGE

Wireless sensor nodes are computation intensive device. Although generation of temporary key involves use of public key cryptography only once for a particular session of communication, however, due to limitation of wireless sensor device this strategy may

not be applicable in full satisfaction. For the above limitation we have revised our scheme and proposed another scheme. We have discussed this scheme in the next section.

6.5 IDENTITY BASED SYMMETRIC KEY CRYPTOGRAPHY FOR PATH KEY ESTABLISHMENT

6.5.1 PAIRWISE KEY GENERATION

This is our second proposal where we have used symmetric key cryptography for path key establishment. Before deployment of the network, a *Trusted Authority* or *KGC* generates private key for each node *ID*. The procedure of generating private key along with the other parameters is done by Key Generation Centre before deployment of the network. The Key Generation centre executes the procedure Setup and Extract. After generation of private key for each id, the ids are loads into the respective node. We have already discussed the procedure Setup and Extract in section 5.5.2. After deployment of the network, if any pair of node needs to establish a temporary key for communication directly, they execute the following algorithm.

Computation by node A

1. Compute $Q_{ID_B} = H_1(ID_B)$
2. Shared key $K_{A,B} = H_2(t(k_{pub}, d_{ID_A}, Q_{ID_B}))$.
3. Generate a random key K_{ran}
4. Encrypt K_{ran} by $K_{A,B}$ using any symmetric algorithm to get C_1.
5. Encrypt C_1 using shared key between itself and the nearest neighbor node to which it will send to get C_2.

Computation by node *B* to get K_{ran}

1. Compute: $Q_{ID_A} = H_1(ID_A)$
2. Shared key $K_{B,A} = H_2(t(k_{pub}, d_{ID_B}, Q_{ID_A}))$. It has already shown that $K_{B,A} = K_{A,B}$
3. Decrypt C_2 to get C_1 using common key between itself and nearest neighbor node from which C_2 came.
4. Decrypt C_1 to get K_{ran} using $K_{B,A}$.

Fig 6.3: Algorithm for non interactive path key establishment using symmetric key cryptography

For describing the above algorithm, we assume that there is a pair of node in a Wireless Sensor Network who wants to communicate with each other without having any pair-wise secret key. The nodes we are assuming are A and B where node A wants to initiate the communication with node B. Therefore node A first computes the corresponding public key Q_{ID_B} for node B where the id of node B is ID_B. It uses function $H_1( )$ which maps from one domain into the domain of elliptic curve points. After generation of public key of node B, it calculates pair-wise common key $K_{A,B}$. It uses ηT pairing as a pairing tool. After applying pairing function to generate the key, it uses another function H_2, which maps the pair-wise key into the domain appropriate for encryption using any symmetric key algorithm. Node A uses this key to encrypt the random key K_{ran} using an appropriate symmetric key algorithm which produces C_1 as the corresponding cipher text. On the other hand, after reaching C_1 to node B, it also follows the same technique for generation of pair-wise common key. To generate this key, it calculates the public key for node A and uses this public key along with its own private key $d_{ID_B,}$ and master public key K_{pub}. Thus it generates $K_{B,A}$. We have already shown in section 5.5.2 that due to billinerity of pairing function, the keys generated by both A and B are equal i.e, $K_{A,B} = H_2(t(k_{pub}, d_{ID_A,} Q_{ID_B})) = H_2(t(k_{pub}, d_{ID_B,} Q_{ID_A})) = K_{B,A}$. Therefore, node B can decrypt C_1 to get K_{ran} by using the key $K_{B,A}$. Hence they share the temporary key K_{ran} and can start communication directly with each other (assuming that they both lies within the same communication range) for an entire session which is secure. The pair of nodes under discussion can either store this temporary key K_{ran} or derive the key from the identities every time they are needed. This constitutes a memory/computation trade-off. However, computing shared keys requires a pairing computation applying the pairing function $t( )$. On the other hand, memory space is very cheap and growing fast, while the pre-shared keys and public keys are fairly short.

6.5.2 SECURITY ANALYSIS

This scheme is fully secure as any of the intermediate nodes neither have the private key of *A* nor *B*. Therefore they can not generate a pair-wise key which equals to $K_{B,A}$ or $K_{A,B}$ and hence can not decrypt K_{ran} for compromising the newly generated link.

6.5.3 PERFORMANCE ANALYSIS

As we have discussed that Wireless Sensor Network is collection of sensor nodes which are computational intensive devices, restricts us to apply any algorithm which can be suitably applicable to this network. Therefore, to propose a solution for the security requirement of temporary key generation during path key establishment, we had to consider this vital restriction. We have used identity based encryption scheme which require the use of bilinear functions defined on elliptic curve points such as the Weil Pairing or the Tate Pairing. Today, the Tate pairing is considered the most convenient pairing function in terms of computational cost [14, 19, 20, 21]. [14] shows how Tate pairing based on supersingular elliptic curves can be efficiently implemented in software in order to obtain reasonable timing on a 32-bit smartcard. Smartcard is a computational intensive device much like sensor nodes. Therefore, use of identity based encryption having Tate pairing as the bilinear function can be applicable to sensor devices. ηT pairing proposed by [24] is a pairing tool which is closely related to Tate pairing and is fastest pairing tool. [19] shows the most efficient pairing namely ηT pairing implementation on Intel Core 2 duo processor with 2.66 GHz speed. This is an efficient implementation of ηT pairing on Core 2 Duo processor. This is the most efficient pairing scheme upto 2008 Dec 21 as far they [19] claim.

We have used ηT pairing as pairing scheme for simulation of the scheme proposed by us. We have simulated in Visual C++ language. Previously we thought to simulate our proposed scheme using Tate pairing. However, as we found that ηT pairing is faster and closely related to Tate pairing, have decided to use ηT pairing as pairing tool. The performance of our scheme proposed on Intel Core 2 duo processor with 2.00 GHz speed reveals the following result.

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Avg.**
0.139	0.134	0.135	0.136	0.129	0.130	0.131	0.134	0.137	0.129	0.133

Note: Each value is in millisecond.

Table 6.1: Result of computation time of our proposed second scheme

From the above table, we can see that after 10 executions, the average computation cost is only 0.133 milliseconds. However, from [14] we observe that the use Tate pairing scheme as pairing tool consumes 41 millisecond which is more. Therefore, ηT pairing scheme is more applicable. Again, the scheme proposed by [14] was

applicable to smart card which is severely constrained with computational constraint. Therefore, the ηT pairing scheme proposed by [19] is appropriately applicable to Wireless sensor devices. This clearly validates our proposals for securing temporary key during path key establishment. However, our first proposal includes an addition computation for public key cryptography may consume an extra computational cost. But in case of our second proposal, the extra computation has optimized by introducing symmetric technique.

CONCLUSION

Wireless sensor Network is a collection of sensor devices which are severely constrained by power, computation, memory, and communication. Therefore, implementation of any type of application in this network is challenging.

Implementation of secure communication between the nodes in this network is also a very challenging task. Use of single key for the whole network can serve as a solution. However, compromises of that key reveal the whole network. Public key cryptography can provide the best resiliency to the network. However, cost of computation of public key cryptography is such that it may not be applicable to Wireless Sensor Network. Storing keys for every other node is the best solution in respect to security and computation. However, this scheme is not scalable as addition of nodes may overwhelm the memory capacity for keeping shared keys for every other node.

The optimum solution is to keep fixed number keys into each node in such a way that maximum pair of node would be having common key with each other. Combinatorial design is a tool which is used for such arrangement. Transversal Design is a combinatorial design which provides a deterministic solution for key pre-distribution which brings a pattern into the key ids in each node that make easy for finding common key between a pair of nodes. Merging of blocks to form a node provides more resiliency. However, random selection of blocks for merging raises an amount of communication cost. We have modified this scheme and proposed a scheme which selects blocks for merging to form a node deterministically and hence introduce a pattern of block ids in a node. This reduces the communication cost for finding shared keys. Merging of blocks randomly consumes *O(z)* communication cost whereas in our scheme, communication cost is O(1), a constant value. All the other parameters are almost same. However, resiliency can be improved by revising merging scheme.

There is another problem in Key pre-distribution scheme is that during path key establishment; the temporarily generated key is not secure. This is because; compromise of any intermediary nodes reveals the temporary key and hence compromise the newly generated link. A few researchers proposed schemes which modify routing strategy to merely avoid the attack. However, till now, no proposal has been made to defend the attack. We for the first time proposed two schemes for providing security to this key. Our

first proposal is based on Id based public key cryptography. However, public key cryptography is not suitable to WSN even it needs only once for a particular session. Hence, we proposed another scheme which is based on Id based symmetric key cryptography. We have used identity based encryption in both schemes which includes an efficient Tate pairing as a bilinear function. Tate pairing is applicable to computational intensive devices like smartcards and hence is applicable to this type of network. The execution of identity base encryption or decryption requires only once for a particular session and hence this is much suitably applicable to Wireless Sensor Network.

BIBLIOGRAPHY

[1] D. Chakrabarti, S. Maitra and B. K. Roy. “A key pre-distribution scheme for wireless sensor networks: merging blocks in combinatorial design”, in proceedings of 8th Information Security Conference, ISC 2005, LNCS 3803, pp. 105-114, 2006.

[2] Jooyoung Lee and Douglas R. Stinson, “On the Construction of Practical Key Predistribution Schemes for Distributed Sensor Networks Using Combinatorial Designs” ACM Transactions on Information and Systems Security, Vol. 11, No. 2, Article 5, Pub. date: May 2008.

[3] Camtepe, S.A., Yener, B.: “Combinatorial design of key distribution mechanisms for wireless sensor networks”. In Samarati, P., Ryan, P.Y.A., Gollmann, D., Molva, R., eds.: ESORICS. Volume 3193 of Lecture Notes in Computer Science., Springer (2004) pp. 293–308.

[4] Jooyoung Lee, Douglas R. Stinson "Deterministic Key Pre-distribution Schemes for DistributedSensor Networks”, in proceeding of SAC 2004, LNCS 3357, pp. 294-307.

[5] Jooyoung Lee, Douglas R. Stinson "A Combinatorial Approach to Key Predistribution for Distributed sensor Networks" in IEEE Wireless Computing and Networking Conference (WCNC 2005), New Orleans, LA, USA, 2005.

[6] Sushmita Ruj and Bimal Roy “Key Establishment Algorithms for some Deterministic Key Predistribution Schemes” ISPA 2007:pp. 431-445.

[7] A. P. Street and D. J. Street. “Combinatorics of Experimental Design”. Clarendon Press,Oxford, 1987.

[8] D. Stinson. Combinatorial Designs: Construction and Analysis. Springer, New York, 2003.

[9] Eschenauer, L., Gligor, V.B.: "A key-management scheme for distributed sensor networks". In: Proceedings of the 9th ACM Conference on Computer and communications Security, pp. 41–47. ACM CCS (2002).

[10] Gura, N., Patel, A.,Wander, A., Eberle, H., Shantz, S.C.: "Comparing elliptic curve cryptography and RSA on 8-bit CPUs. CHES", LNCS 3156, pp. 119–132 (2004).

[11] Sheu J., Cheng J. "Pair-wise path key establishment in wireless sensor networks" Computer Communications 30(11-12): 2365-2374, 2007.

[12] Hui Ling Taieb Znati "End-to-End Pairwise Key Establishment using Multi-path in Wireless Sensor Network" IEEE GLOBECOM 2005 pp 1847-1851.

[13] Guanfeng Li, Hui Ling and Taieb Znati "Path Key Establishment using Multiple Secured Paths in Wireless Sensor Networks" CoNEXT 2005.

[14] G. M. Bertoni, L. Chen, P. Fragneto, K. A. Harrison, and G. Pelosi "Computing Tate Pairing on Smartcards" Technical Report, STMicroelectronics Centro Direzionale Colleoni 20041 Agrate, Italy

[15] Matthew Baldwin "Identity Based Encryption from the Tate Pairing to Secure Email Communications" A CSMENG MAY 2002.

[16] Boneh, Matthew K. Franklin "Identity-Based Encryption from the Weil Pairing Advances in Cryptology" Proceedings of CRYPTO 2001 (2001)

[17] www.computing.dcu.ie/~mike/tate.html, A tutorial on basics of Tate pairing and its properties.

[18] Benoit Libert "The Applications of Pairings in Cryptography" Brazilian Symposium on Information and Computer Systems Security (SBSeg 2007) August 30th 2007.

[19] MITSUNARI Shigeo "A Fast Implementation of ηT Pairing in Characteristic Three on Intel Core 2 Duo Processor" Cybozu Labs, Inc, January 14, 2009.

[20] Katrin Hoeper and Guang Gong "Bootstrapping Security in Mobile Ad Hoc Networks Using Identity-Based Schemes with Key Revocation" Technical Report, Department of Electrical and Computer Engineering University of Waterloo Waterloo, ON, N2L 3G1, Canada.

[21] Seiichi Matsuda, Atsuo Inomata, Takeshi Okamoto, and Eiji Okamoto "Performance Evaluation of Efficient Algorithms for Tate Pairing" IEEE 2005.

[22] Subhasish Dhal, P. M. Khilar "Deterministic Merging of Blocks for Key Pre Distribution in Wireless Sensor Networks", in proceeding of National Conference on Recent Advances in Communication Technology, NIT Rorkela, January 2009. pp 119-123.

[23] Subhasish Dhal, P. M. Khilar " ID-NIKD based security of Temporary key during Path Key Establishment in Key Pre Distribution for WSN", accepted in international conference ICEMC^2, PESIT Bangalore July 2009.

[24] Jean-Luc Beuchat, Nicolas Brisebarre, Je´ re´mie Detrey, Eiji Okamoto, Masaaki Shirase, and Tsuyoshi Takagi, "Algorithms and Arithmetic Operators for Computing the ηT Pairing in Characteristic Three" IEEE TRANSACTIONS ON COMPUTERS, VOL. 57, NO. 11, NOVEMBER 2008.

LIST OF DISSEMINATION

[1] Subhasish Dhal, P. M. Khilar "Deterministic Merging of Blocks for Key Pre Distribution in Wireless Sensor Networks", in proceeding of National Conference on Recent Advances in Communication Technology, NIT Rourkela, January 2009. pp 119-123.

[2] Subhasish Dhal, P. M. Khilar " ID-NIKD based security of Temporary key during Path Key Establishment in Key Pre Distribution for WSN", accepted in international conference ICEMC^2, PESIT Bangalore July 2009.