PRIMITIVE PYTHAGOREAN TRIPLES IN KEY MANAGEMENT OF SENSOR NETWORKS

By

KOTHAPALLI YASHWANTH

Bachelor of Technology in Computer Science

Jawaharlal Nehru Technological University

Hyderabad, India

2009

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2011

PRIMITIVE PYTHAGOREAN TRIPLES IN KEY

MANAGEMENT OF SENSOR NETWORKS

Thesis Approved:

Dr. Subhash Kak

Thesis Adviser

Dr. Johnson Thomas

Dr. Michel Toulouse

Dr. Mark E. Payton

Dean of the Graduate College

ACKNOWLEDGMENTS

It gives me immense pleasure in acknowledging all those people without whom, this thesis would have been impossible. I would like to offer my deepest gratitude to my Thesis advisor Dr. Subhash Kak, who has supported me throughout my thesis with his patience and knowledge. It is his encouragement and effort without which, this thesis would not have been completed. His strong support and timely advice has always been a boosting energy for me to further explore this topic. I could not wish for a better advisor.

I would like to thank Dr. Johnson P Thomas and Dr. Michel Toulouse for their valuable comments which helped me to give a good shape to my thesis.

I would like to thank my parents, RamaKrishnam Raju and Lalitha and my brother Venkatapathi Raju for their support and encouragement. I would also like to thank my friends Singi Reddy Rohith Reddy, Aileni Anvesh Reddy, Venkat Ravinder, Pradeep Dantala, Naren Dhupati, Vijay Pratap, Krishnakanth, Praveen Chandrahas, Mehera, Tejaswi, Vijay Singh, Hima Bindu, Shashank Sadalia, Parmeshwar Reddy, Chakradhar Reddy, Nitesh Reddy, Siddharth Echampati, Hari Kishan Kotha Sagar Kodukula, Rohith Vaidya, Ansih Koppula and Sujith Reddy Beemireddi for supporting me throughout my research.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

Wireless sensor networks act as building blocks for smart environments which help in automating various systems like transportation, home, utilities and industries. Smart environments require sensory data from real world. This sensory data is collected by distributed wireless sensor networks and is processed to retrieve important information. A wireless network consists of hundreds or thousands of autonomous devices, called sensor nodes, which are spatially distributed. Each sensor node has a radio transceiver for communicating at short ranges and a microcontroller that is capable of processing data. Various applications of wireless sensor networks are tracking, environmental monitoring, patient health monitoring and traffic monitoring and animal monitoring. Sensor nodes also find applications in battle field surveillance.

Each node has limitations on their resources such as memory, energy, computational power and communication range. Each sensor node should have the ability to self-organize, coordinate with other sensor nodes and perform in extreme weather conditions. A sensor network should be in a position to add new nodes and also withstand the loss of some nodes.

In a sensor network, each node collects data from the surroundings and tries to send it to the base node from which it is sent to satellites and supercomputers for analyzing and processing data.

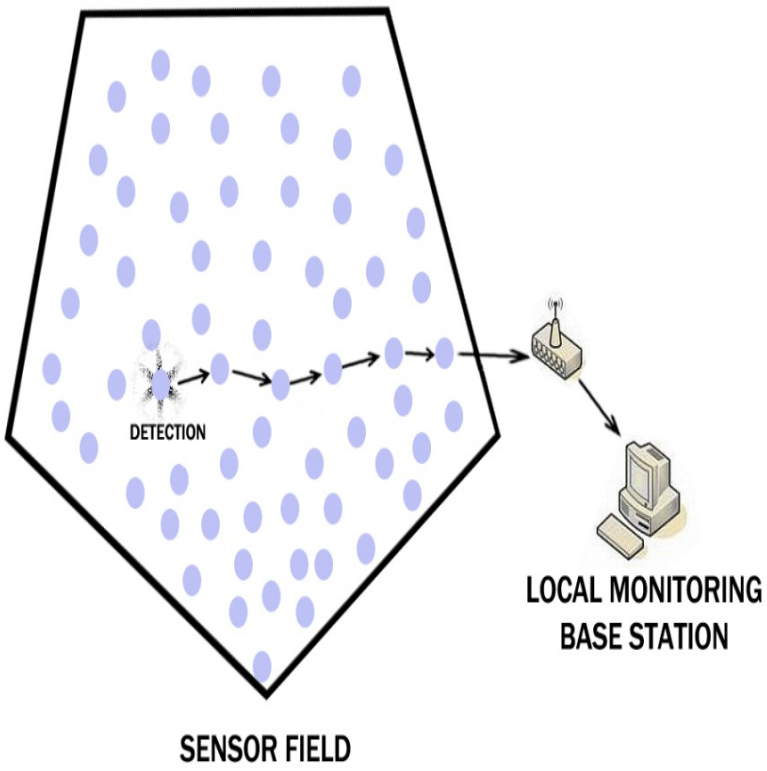


Figure 1. A Wireless Sensor Network

The wireless communication between the nodes takes place through radio transceiver introduces a possibility of interception of messages by an adversary. Moreover, due to cost consideration, tamper resistant hardware is not used to protect the keys or other critical data. With the threat of interception, one has to take care to minimize this impact of such loss on the neighboring nodes and the network.

Since these networks can be deployed in hostile environments, they may have to face attacks such as masquerade attacks, spamming with erroneous information, and information retrieval by listening to the traffic and even physical attacks. In order to minimize the effect of these attacks, the messages transferred between two communicating nodes must be properly encrypted and authenticated. In order to start a communication, two nodes must share a common key and thus the problem reduces to that of key management. Much research has been done in how efficiently the keys are to be generated [1] or pre-distributed [2],[3] by considering the limitations on energy, memory and cost of sensor nodes. Since nodes are battery powered, key management scheme should have low communication costs. Nodes have limited memory, so key management scheme should be designed to minimize memory requirements. Another consideration is node resiliency.

In this thesis we propose a key self-establishment scheme using Primitive Pythagorean triples (PPT) as suggested by Kak [4]. Pythagorean triples are the lengths of sides of a right angled triangle (a, b, c). The generation of events of specific probability is of importance in cryptography and in applications such as e-commerce [5],[6]. Such events may be generated by a variety of methods that include prime reciprocals [7],[8],[9] or by the use of specific modular operations [10]. The generation of random events is also tied up with the question of algorithmic probability [11]. Pythagorean triples can generate probability events. We can generate infinite number of Pythagorean triples i.e. x (a, b, c) where x>1. A primitive Pythagorean triple is the one in which neither of the three numbers have any common factor.

Euclid [12],[13] proposed that Pythagorean triples may be obtained using the formula

$a = m^2 - n^2$

$c = m^2 + n^2$

$b = 2mn$

Here m, n are two positive integers where one of them is odd and the other is even.

Let us consider $m = 10$ and $n = 5$

$a = 10^2 - 5^2 = 75$

$b = 2(10)(5) = 100$

$c = 10^2 + 5^2 = 125$

Since $125^2 = 100^2 + 75^2$, we know that (a, b, c) is a Pythagorean triple.

Using an odd number we could generate PPTs and using their class sequences we could generate a peculiar string, which can be used as a key. In this scheme, much importance is given to optimal utilization of memory available in a node. If two nodes share an odd number, they could generate a class sequence which can be utilized as a key for building a communication link between them. This scheme just stores the odd numbers in sensor nodes instead of keys for establishing a safer communication channel. So, more odd numbers which are 32 bit in length could be stored than keys which are 128 bit in length. The proposed scheme increases the connectivity of the graph without compromising with node resilient nature. Even the overhead of communication between the nodes to generate a common key is minimized, unlike the case of Blom's scheme [1].

The rest of the thesis is organized as follows, Chapter 2 describes the key pre distribution and key establishment as proposed earlier, Chapter 3 describes the scheme of this thesis, Chapter 4 describes the results of simulation, and Chapter 5 presents the conclusions of the thesis.

CHAPTER II

LITERATURE REVIEW

Eschenauer and Gligor [2] proposed a key pre-distribution scheme in which a large number of keys, say N, are generated and each key is given a unique identifier. Then M keys which are to be placed in a key ring are chosen out of N generated keys and given to each sensor node. This scheme decreases the number of keys to be stored in the sensor memory by considering the memory overhead of a sensor node. This scheme is different from the method of storing (N-1) keys in each sensor node and also the method of storing only one key in each sensor node in a tamper resistant hardware. It is required in this scheme that two neighboring nodes should contain at least one common key. In the case of neighboring nodes containing only one common key, the adversary has to gain control over one node to break the network. Though this scheme decreases the number of keys to be stored in the sensor memory, it doesn't achieve enough security and is vulnerable to smaller scale attacks.

Chan, Perrig and Song proposed a key pre-distribution scheme [3], which states that a large number of keys are generated say N and some set of keys are selected out of these keys. These selected keys are sent to each sensor node as in the case of Eschenaeur and Gligor scheme [2]. But in this scheme two neighboring nodes must have at least have q common keys in order to communicate. It is not enough for an adversary to gain control over one node, he has to know all q keys which are used to build a communication link between two nodes. This scheme could achieve security in smaller scale attacks but couldn't with stand the larger scale attacks. This scheme could achieve more node resilient nature when compared to Eschenaeur and Gligor scheme [2].

Chan and Perrig in their PIKE [15] protocol proposed a pair-wise key establishment scheme. In this scheme, the keys are distributed in such a way that any two nodes A, B which want to communicate with each other can find an intermediate node (peer node) C that has unique pair wise keys in common with both A, B. So, A can have a safe communication link and send messages to B through C. Thus in this scheme, each node shares unique pair wise keys with $O(\sqrt{n})$ intermediate nodes. Here n represents the total number of nodes in the sensor network. In this key pre-distribution scheme, unique pair wise keys are distributed such that they are shared by two nodes. If the peer node is compromised, the key established will no longer be secure.

Du et al. [14] proposed a key distribution scheme using deployment knowledge. In this scheme, the assumption is that the sensor nodes are grouped into smaller groups and are deployed out of plane one after the other. So, the groups which are deployed in sequence to each other will be neighbors. This whole scheme is divided in to three phases namely key pre-distribution phase, shared-key distribution phase and path establishment phase. In the key pre-distribution phase, the key pool is divided into parts and each part is assigned to each group of sensor nodes. The main idea behind this is that the neighboring groups of sensor nodes should have key pools which have more keys in common. The groups which are far away might have key pools which share less

number of keys or no keys at all. The next two phases are applied in each group which is same as Eschenauer-Gligor scheme [2].

Kak, in his research paper [4] stated the importance of PPTs (a, b, c) in the early Vedic periods in India and their description in geometry books of India. He proposed various ways to generate PPTs such as the following:

1. Using Gopala-Hemachandra quadruple (g, e, f, h)

In this, he states that if $a = gh,\ b = \frac{(c-a)f}{e}$ and $c = eh + fg$, then (a, b, c) is a Pythagorean triple. If g is odd and g, h, e, f are co primes, then the resulting (a, b, c) will be a Primitive Pythagorean triple.

2. Using two odd numbers s, t which are co-primes

$$a = st$$

$$b = \frac{s^2 - t^2}{2}$$

$$c = \frac{s^2 + t^2}{2}$$

He proposed some properties of PPTs that if (a, b, c) represents a PPT, then all three integers can't be even, both a and b cannot be odd and c even. He classified each PPT into different classes according to the divisibility of a, b, c with 3 and 5 respectively.

1. Class A, in which a is divisible by 3 and c by 5

div	a	b	c
3	X		
5			X

2. Class B, in which a is divisible by 5 and b by 3

Div	a	b	C
3		X	
5	X		

3. Class C, in which a is divisible by 3 and 5

div	a	b	c
3	X		
5	X		

4. Class D, in which b is divisible by 3 and c by 5

div	a	b	c
3		X	
5			X

5. Class E, in which a is divisible by 3 and b by 5

div	a	b	c
3	X		
5		X	

6. Class F, in which b is divisible by both 3 and 5

div	a	b	c
3		X	
5		X	

He presented first 34 PPTs when arranged in increasing order of hypotenuse along with their classification

PPT	w(x)
(3,4,5)	A
(5, 12, 13)	B
(15, 8, 17)	C
(7, 24, 25)	D
(21, 20, 29)	E
(35, 12, 37)	B
(9, 40, 41)	E
(45, 28, 53)	C
(11, 60, 61)	F
(33, 56, 65)	A
(63, 16, 65)	A
(55, 48, 73)	B
(13, 84, 85)	D
(77, 36, 85)	D
(39, 80, 89)	E
(65, 72, 97)	B

(99, 20,101) E

(91, 60,109) F

(15,112,113) C

(117, 44, 125) A

(105, 88,137) C

(17,144,145) D

(143, 24,145) D

(51,140,149) E

(85,132,157) B

(119,120,169) F

(165, 52,173) C

(19,180,181) F

(57,176,185) A

(153,104,185) A

(95,168,193) B

(195, 28,197) C

(133,156,205) D

(187, 84, 205) D

Blom [1] proposed a key establishment scheme which helps to generate a unique shared key by two nodes for communication. Blom's secure parameter t says that unless there is a capture of t+1 node out of N nodes in the network, an adversary could not know the keys held by the other users. Increasing the value of t increases the node resiliency nature of the network, but it also increases the memory requirement for storing key information. In the initial stages, the base node generates a public matrix P of order $(t+1) \times N$ over a finite field GF(q), using Vandermonde matrix.

$$P = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ n_1 & n_2 & n_3 & \cdots & n_N \\ {n_1}^2 & {n_2}^2 & {n_3}^2 & \cdots & {n_N}^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ {n_1}^t & {n_2}^t & {n_3}^t & \cdots & {n_N}^t \end{bmatrix}$$

Then the base node selects a symmetric matrix S of order (t+1) × (t+1), which is only known to itself. Then matrix A of order N × (t+1) is generated using matrices S and P, i.e

$A=(S.P)^T$

Then K matrix is calculated using A and P matrices i.e

$K = A.P = (S.P)^T.P = P^T.S^T.P$

$= P^T.S.P = (A.P)^T = K^T$

The Central node sends j-th row of matrix A and j-th column of matrix P to each j-th node in the network, where j varies from 1, 2…N. Two nodes (node j, node k) could generate a common key using j-th row of A matrix and k-th column of P matrix. So, two nodes exchange their columns of P matrix received from base node in order to generate a shared a key and to establish a communication path between them.

CHAPTER III

PROPOSED APPROACH

If two nodes want to communicate securely with each other, both have to share a key. In order to pre-distribute or establish keys between nodes, one has to consider the limitations of sensor nodes such as low energy resource, low memory availability and resilient nature against node capture. The best way of key management would be to store N-1 keys in all the nodes, since the memory of a sensor node is limited, this scheme is not practical. If we store one key in all nodes, then capture of node would compromise whole network. It would be effective to store as many keys as possible in a sensor node, given its memory constraints. This is accomplished in this thesis by generating keys using the properties of PPTs.

3.1. Properties of PPTs

Any odd number could be expressed as sum of an even number and an odd number. A PPT could be generated if these constituents form a co-prime pair.

Algorithm:

For i=2 to n

If (i=odd number)

$$m = \frac{i}{2} + 1$$

$$n = i - m$$

End if

While(n>=1)

If (m, n are co-primes)

$$a = m^2 - n^2$$

$$c = m^2 + n^2$$

$$b = 2mn$$

End if

End while

End for

An explanatory example is given below in Table 1.

.

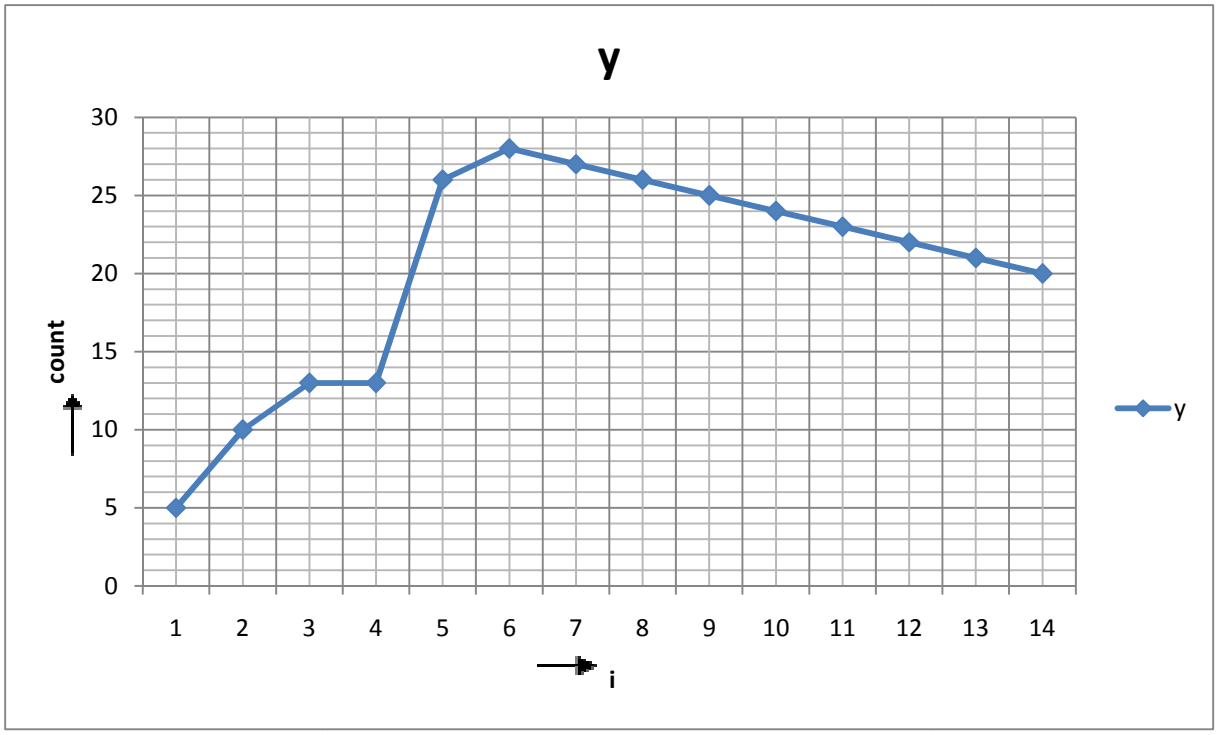
Odd number	Co-prime pair	PPT generated	class
3	(2,1)	(3,4,5)	A
5	(3,2)	(5,12,13)	B
	(4,1)	(15,8,17)	C
7	(4,3)	(7,24,25)	D
	(5,2)	(21,20,29)	E
	(6,1)	(35,12,37)	B
9	(5,4)	(9,40,41)	E
	(7,2)	(45,28,53)	C

	(8,1)	(63,16,65)	A
11	(6,5)	(11,60,61)	F
	(7,4)	(33,56,65)	A
	(8,3)	(55,48,73)	B
	(9,2)	(77,36,85)	D
	(10,1)	(99,20,101)	E

Table 1. Generating PPTs and their classification into their respective classes for a given odd number

Consider first 34 PPTs by arranging them in increasing order of hypotenuse. If the hypotenuse is equal, the PPTs are arranged in increasing order of the first side. The corresponding class sequence of these first 34 PPTs is ABCDEBECFAABDDEBEFCACDDEBFCFAABCDD. This sequence does not include all possible pairwise transitions. To obtain all pairwise transitions, we need to consider up to 8182 PPTs. If we consider w(x)'s continuously 2, 3, 4 or 5 at a time, it

is observed that the class sequences are repeated before 26 PPTs. If we consider i=6 then we obtain 28 unique combinations. The graph of this behavior is given below.



Graph 1**.** Graph between count of unique sequences and different values of i.

Here the x-axis is i, where as Y-axis gives the no: of unique combinations for that i.

Property1.

For any prime number *y*, *n* number of co prime pairs exist, whose sum is the number itself. These generate n PPTs where y = 2n+1.

Proof: Let *y* be a prime number which is a sum of two numbers r and s. Assuming that r and s are non zero and non co primes, there exists a common factor z, between them. Since z divides both r and s, it also divides *y* which is a contradiction to *y* being a prime number. Therefore, all pairs of

numbers (r, s) whose sum is prime numbers are co primes. There will be n such co prime pairs where $y = 2n + 1$. Therefore, n PPTs could be generated from a prime number.

For example y =13

(m, n)	PPT (a, b, c)	class
(7,6)	(13, 84, 85)	D
(8,5)	(39, 80, 89)	E
(9,4)	(65, 72, 97)	B
(10,3)	(91, 60, 109)	F
(11,2)	(117, 44, 125)	A
(12,1)	(143, 24, 145)	D

Table 2. PPTs generated out of 13 and their classes

The number of PPTs is 6 i.e n=6.

Property 2

For all the odd multiples of 15, PPTs generated belong to class C.

Property 3

All prime numbers except 3 ending in 3 or 7 follow a cyclic sequence DEBFADFCFDAFBED.

Property 4

All prime numbers except 5 ending in 1 or 5 follow a cyclic sequence FABDEFDCDFEDBAF.

Indexing of PPTs

PPTs may be indexed in a variety of ways. The most obvious ones with the total length of the sequence in which all transitions between the six classes occur are listed below:

- Arranged in increasing order of a in (a, b, c). All transitions occur in sequence length of 300. The sequence begins as: ABDEFDCCDFEABECEBAABDBFBECAFDDACDC.

- Arranged in increasing order of b in (a, b, c). All transitions occur in sequence length of 4037. The sequence begins as: ACBBAEEDDCCDEABCAFFBEDDCACFBEDDBAF.

- Arranged in increasing order of c-b in (a, b, c). All transitions occur in sequence length of 132. The sequence begins as: ABDEFDCDFCEAEEABCBBBBADFCFDEAADCDC.

- Arranged in increasing order of c-a in (a, b, c). All transitions occur in sequence length of 504. The sequence begins as: ACBAEDCBECDACDBFDEABCAFEFDBDCEBDAF.

- Arranged in increasing order of b-a in (a, b, c). All transitions occur in sequence length of 147. The sequence begins as: CDCDEAAADFBCCCBEAFBBDADEEBFDBECADF.

Other indexing schemes may be used by placing conditions on the three parameters or by imposing uniform or non uniform decimation of the specific PPT sequence. If the sender and the recipient knew the indexing scheme used, then the corresponding class sequence would be used as the key by two parties in order to establish a secure communication link between them.

3.2 Key Management:

A set of odd numbers are chosen at random and distributed to the nodes. The node sends hash values of the odd numbers in order to set a path with other node. Every node gets the information about the number of odd numbers it shares with its neighboring node and a path is established between them which have common odd numbers. Keys are selected based on the class sequences generated by the shared odd numbers. To accomplish this, the key length needs to be selected first. Assuming a key length of 10, the minimum number in a key pool is 23. This is because all the odd numbers greater than 23 generate a class sequence of length 10 or higher.

Comparing the class sequence produced by 13 in Table 2 with the restricted class sequence of length 6for 23 in Table 3, it is observed that the key generated in both the cases is DEBFAD. Thus prime numbers are not included in the key pool. Similarly, all odd multiples of 15 generate class C PPTs alone in the sequence which has to be avoided. Therefore the key pool used in this scheme consists of all the odd numbers excluding prime numbers and odd multiples of 15.

(m, n)	PPT (a, b, c)	class
(12,11)	(23,264,265)	D
(13, 10)	(69,260,269)	E
(14,9)	(115,252,277)	B
(15,8)	(161,240,289)	F
(16,7)	(207,224,305)	A

(17,6)	(253,204,325)	D
(18,5)	(299,180,349)	F
(19,4)	(345,152,377)	C
(20,3)	(391,120,409)	F
(21,2)	(437,84,445)	D
(22,1)	(483,44,485)	A

Table 3. PPTs and their classes of 23

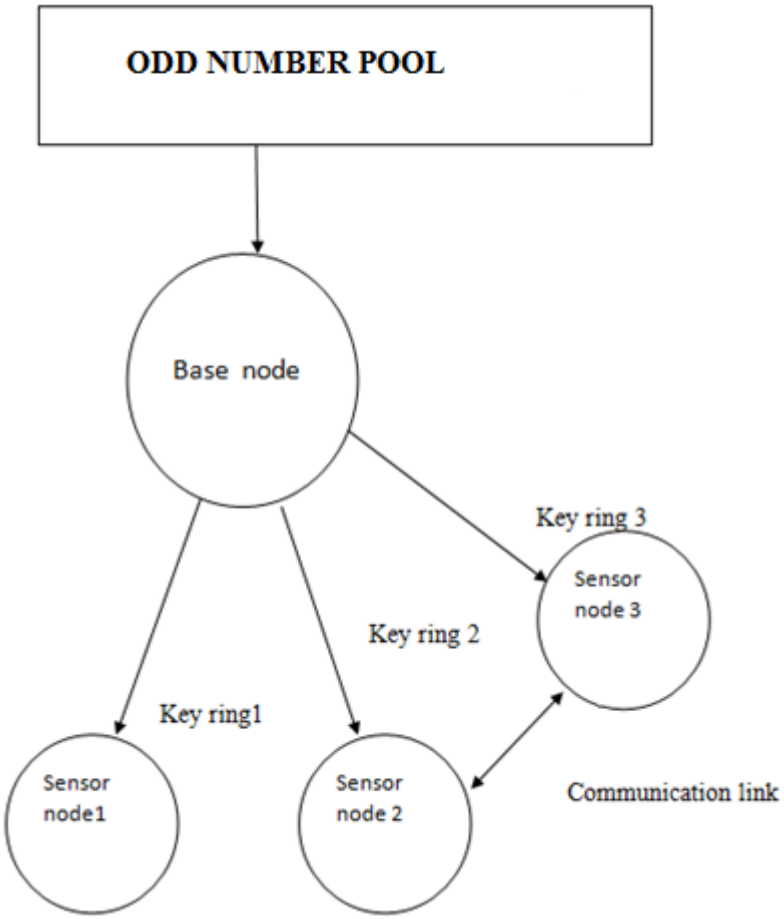


Figure 2. Key Distribution Using PPTs

In this scheme, for two nodes sharing one and only one odd number with a selected key of length k, a class sequence of same length is considered from that odd number. If two nodes share m odd numbers, where m>1, then they are arranged in increasing order and a class sequence of length $\frac{k}{m}$ from each odd number and an additional length of $k\%m$ from the maximum odd number are considered to generating a key.

Assume a key length of 10 for shared odd numbers 21, 25, 27.The corresponding class sequences are ECEAAE, BCBCBBBCB, and ACEAEEAEC respectively. The key is generated by considering class sequences of length 10/3, 10/3, 10/3 and 10/3+10%3 from original class

sequences of 21, 25 and 27 respectively. This is obtained as ECEBCBACEA. Assume the number of shared odd numbers between the nodes is x and the key length is y where x>y. In this case, the odd numbers are arranged in increasing order and only the first classes from the first y odd numbers are considered to generate the key. In this Scheme, with little computational overhead, a lot of memory is saved by storing just integers which are 32 bit long in the sensor nodes instead of keys which are of 128 bits in length. More resiliency against the node capture is achieved when compared to Eschenaeur and Gligor scheme [2], as keys are generated using more than one common odd number if available. The *q*-composite scheme proposed by Chen et. al.[3] requires *q* keys of neighbors to be shared, which increases the resilience in the network. This scheme reduces the size of the key pool but increases the key ring size at each node by repeating the keys to be stored. A more connected graph could be obtained through the proposed scheme, when compared to q-composite scheme as the nodes are connected even if they share one common odd number.

3.3 Key Pool Size Calculation

The notations used in this section are

P1	Desired probability that a sensor network to be connected
p	probability that two nodes having a communication link
n	network size, number of nodes
n1	average number of neighboring of a given node.
d	number of connected edges that a node can have with its neighboring nodes.
r	number of keys in a key ring
P	number of keys in a key pool

In these networks, it is not needed that every node has to have a connected edge with all its neighbors in order to have a communication link between any given source node and destination node. In these situations we mainly need to know, for a given network size, the number of connected edges that a node has with its neighbors, the key pool size and the key ring size.The mathematical equations given by to Eschenaeur and Gligor scheme [2], are used to answer these questions.

Let p be the probability that a share key exists between two sensor nodes, n be the network size and the equation below gives the expected degree of a node (i.e., the average number of edges connecting that node with its graph neighbors).

$$d = p \times (n - 1) \tag{3.3.1}$$

In Random Graph theory, for a given graph G(n, p), n represents the number of nodes in the graph and p represents the probability that an edge exists between any two nodes. Erdos and Renyi stated that there is a value of p between 0 and 1 such that the given graph G(n, p) is certainly connected, given desired probability P1 for graph connectivity.

$$P1 = 1 - e^{-c} \tag{3.3.2}$$

where c is a real constant

and $p = \frac{\ln(n)}{n} + \frac{c}{n}$ (3.3.3)

So, for a given desired probability (P1) that a graph to be connected, c could be calculated from equation 3.3.2. c and n could be used to calculate p from equation 3.3.3 and further d is calculated from p using equation 3.3.1.

But due to the constraints in wireless communication range the number of nodes that a node can communicate with will be n1<<n. So our new probability will be p1, which is given by

$$p1 = \frac{d}{n1-1} \ll p \tag{3.3.4}$$

Now the probability that two nodes share at least one key in their key ring sizes chosen from key pool is p1.

$$p1 = 1 - Prob[two\ nodes\ donot\ share\ a\ key]$$

The number of possible key rings that can be picked from a key pool P without replacement is

$$\frac{P!}{r!\,(P-r)!}$$

After a key ring has been picked, the number of possible ways that a ring can be picked which does not have any common key with the previous key ring is

$$\frac{(P-r)!^2}{(P-2r)!P!}$$

Therefore p1 is

$$p1 = 1 - \frac{(P-r)!^2}{(P-2r)!P!}$$

Simplifying the equation we get

$$p1 = 1 - \frac{(1-\frac{r}{P})^{2(P-r+0.5)}}{(1-\frac{2r}{P})^{(P-2r+0.5)}} \tag{3.3.5}$$

An example:

Let us suppose that a wireless sensor network contains n=1000 nodes. Assume that the network has to be connected with a probability of P1=0.999 and each node in the network can communicate with n1=40 other nodes. Assume that each node has a memory to hold only 45 keys, which says that the key ring size is r= 45. By using equation 3.3.2, the real constant c is calculated as 6.9. Plugging in this value of c and n using equation 3.3.3, p is calculated as 0.0138. Using p and equation 3.3.1, d is calculated as 14. So, this calculation shows that each node should be able to have at least 14 connected edges with the neighbors to have the whole network to be connected. Since in this example we have been given that a node has 40 neighborhood nodes, using equation 3.3.4, p1 is calculated as 0.36. Using the p1, key ring size and equation 3.3.5 the key pool size is calculated as 4500. If the number of neighboring nodes is increased the probability p1 decreases and either the key ring has to increase or the key pool size has to increase or both. Increasing n1 from 40 to 60, decreases p1 to 0.23. For the same key ring size 45, the key pool size increases to 7800 from 4500.

CHAPTER IV

FINDINGS

Given the probability based on which the graph is to be connected, the number of neighbors a node in a network, and key ring size, one can calculate the number of connected edges that a node should have with its neighbors(d) and key pool size. From the key pool, key rings are selected at random and assigned to the nodes. Since keys are just odd numbers they are arranged in increasing order inside the nodes. We use maximum of k1 odd numbers which is equal to the length of the key. The keys of nodes are compared till maximum of k1 common numbers are obtained. Then an adjacency matrix with numbers of odd numbers in common is generated. Once the adjacency matrix is obtained, the number of hops, which is number of nodes present in the path between source and destination nodes, can be calculated.

An Example:

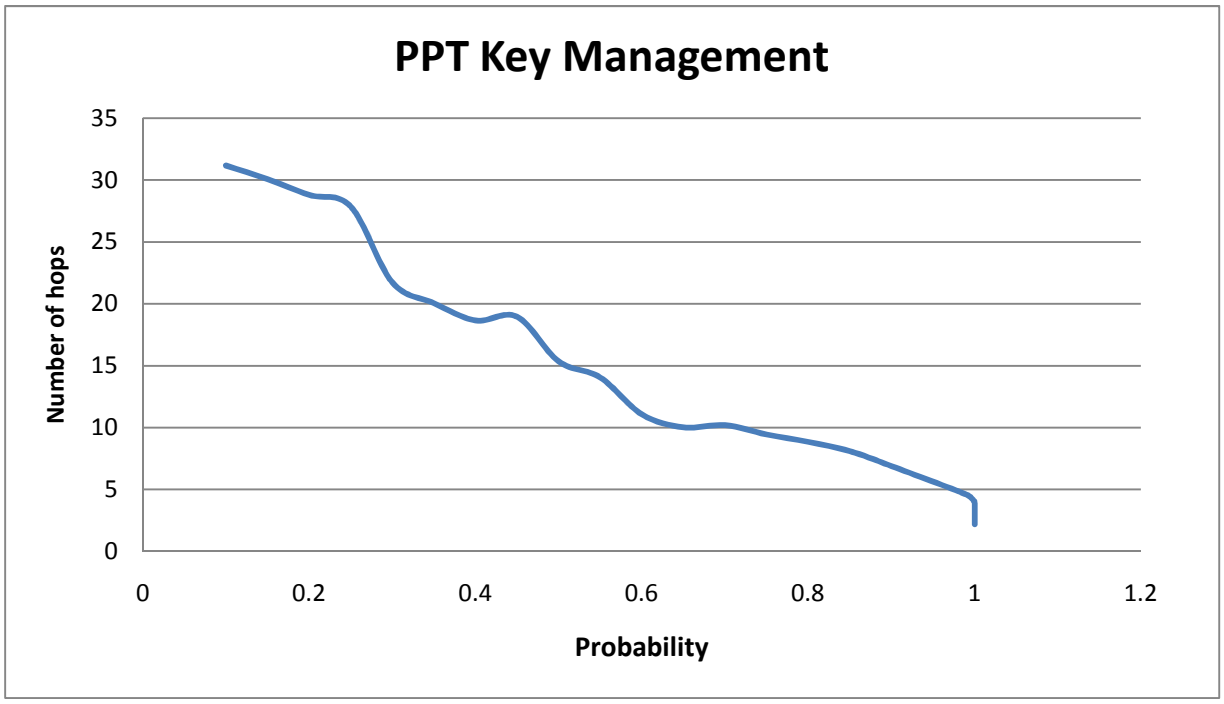
P1=0.9, n=20,n1=5,key ring size=10, key pool size =10,k1=2;

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1		1	2	2	1	0	0	2	2	1	2	0	0	0	1	0	1	2	1	0
2			0	1	1	0	2	2	2	0	2	2	1	0	2	1	1	1	1	0
3				2	0	1	1	2	2	2	1	0	0	1	0	0	1	0	0	2
4					1	0	1	2	2	2	2	1	1	0	2	2	0	1	0	0
5						0	0	2	0	1	2	1	1	1	1	1	0	2	0	1
6							1	2	0	0	1	2	2	2	1	1	2	1	2	2
7								0	2	0	0	1	2	1	0	2	2	1	2	1
8									2	1	2	1	1	1	2	1	1	0	0	2
9										0	2	2	0	1	2	0	2	1	0	2
10											1	0	0	1	1	1	0	0	0	1
11												2	1	2	1	0	2	1	0	0
12													2	1	0	2	2	0	1	0
13														1	0	1	2	0	2	1
14															0	1	2	1	1	1
15																1	1	1	0	0
16																	0	0	1	1
17																		2	1	0
18																			1	0
19																				1
20																				

Table 4. Adjacency matrix for 20 nodes

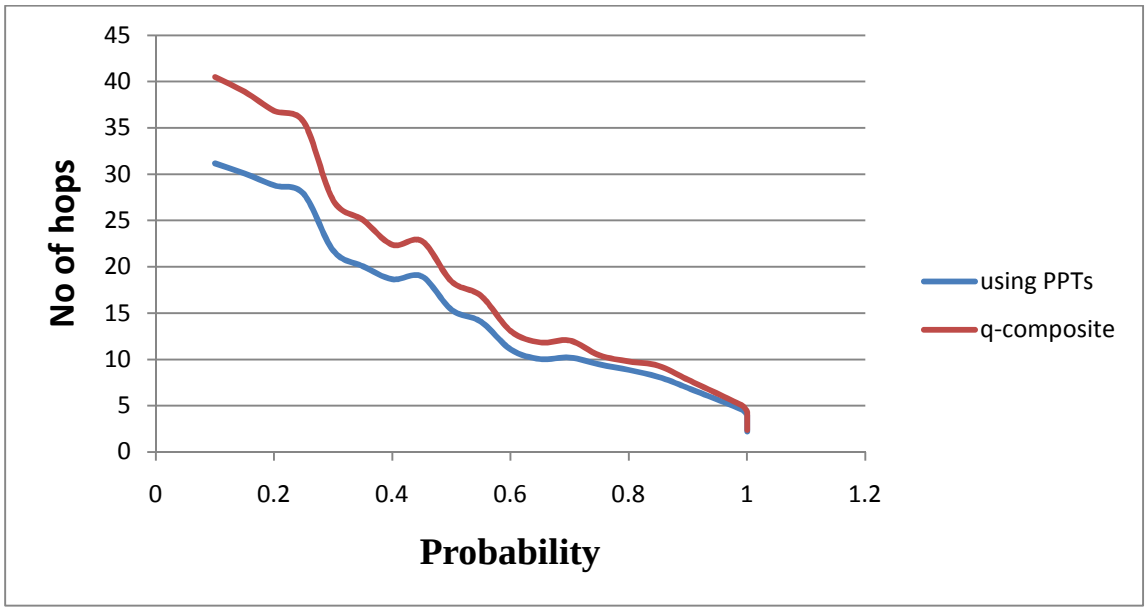
We assume that the source node is 5 and destination node is 12 in the above example. The results show that the number of hops is 1 for the key management using PPTs, which indicates that there is a direct path between source and destination nodes. When the same example is considered for q-composite scheme with q= 2, the results show that node 11 acts as intermediate node between source and destination nodes and number of hops is 2.

The environments are simulated considering the network size of 1000, average number of neighbor nodes as 20 and key ring size(r) as 30, with different probabilities ranging from 0.1 to 0.9999. The number of hops on an average in the considered network is shown in graph 2. The key pool size varies from 2500 to 550, with maximum at probability 0.1, and minimum at 0.9999. The average number of hops is 31.2 hops at probability 0.1 and is 2.4 at 0.9999.



Graph 2. Number of hops at different probabilities in PPT Key Management Scheme

These results are compared with q-composite scheme with q =3in Graph 3.



Graph 3. Presents number of hops of two schemes at different probabilities.

Graph 3 presents the number of hops for Key Management using PPT scheme and q-composite scheme for q=3. When the probability is 0.1, the number of hops for q-composite scheme is 40.508 and is 3.16 at probability 0.9999. When the probabilities are low, the two graphs are far apart but when the probabilities get higher, they approach each other.

CHAPTER V

CONCLUSION

In this Thesis, Primitive Pythagorean Triples and their classification into different classes are initially described. Results on generation of PPTs using odd numbers and on indexing PPTs in various ways are presented. Using these properties a new key management scheme is developed which uses odd numbers to generate keys between nodes and to establish a safe communication channel between two parties. This makes it possible to store more keys in each node. A communication path is established between two nodes if two nodes share at least one odd number in this scheme. The connectivity between the nodes in this scheme is greater than the q-composite scheme, which needs at least q common keys between two nodes to establish a communication path. Simulation results shows that the number of hops required to reach the destination node from source node on an average is 11% less than that of q-composite scheme.

REFERENCES

[1] R. Blom, "An Optimal Class of Symmetric Key Generation Systems", *in Advances in Cryptology: EUROCRYPT'84*, LNCS, vol. 209, pp. 335-338, 1985.

[2] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks", Proc. *ACM Conf. on Computer and Commun. Security*, pp. 41–47, Nov. 2002.

[3] H. Chan, A. Perrig and D. Song, "Random key predistribution schemes for sensor networks", *Proc. IEEE Symp. on Security and Privacy*, pp. 197-213, May 2003.

[4] S. Kak, Pythagorean Triples and Cryptographic Coding, 2010. arXiv:1004.3770

[5] A. Bhattacherjee, "Acceptance of e-commerce services: the case of electronic brokerages", *IEEE Trans on Systems, Man, and Cybernetics – Part A: Systems and Humans*, vol. 30, pp. 411-420, 2000.

[6] S. Kak, "A new method for coin flipping by telephone", *Cryptologia*, vol. 13, pp. 73-78, 1989.

[7] S. Kak and A. Chatterjee, " On decimal sequences", *IEEE Transactions on Information Theory*, vol. IT-27, pp. 647 – 652, 1981.

[8] S. Kak, "Encryption and error-correction coding using D sequences", *IEEE Transactions on Computers*, vol. C-34, pp. 803-809, 1985.

[9] S. Kak, "New results on d-sequences," *Electronics Letters*, vol. 23, p. 617, 1987.

[10] S. Kak, "A cubic public-key transformation," *Circuits, Systems and Signal Processing*, vol. 26, pp. 353-359, 2007.

[11] A. Kolmogorov, "Three approaches to the quantitative definition of information", *Problems of Information Transmission. 1*, pp. 1-17, 1965.

[12] T. Heath (ed.), "The Thirteen Books of Euclid's Elements", Dover Publications, vol. 2, pp. 147-169, 1956.

[13] Euclid, Works: http://aleph0.clarku.edu/~djoyce/java/elements/elements.html

[14] W. Du, J. Deng, Y.S. Han, S. Chen, and P.K. Varshney, "A key management scheme for wireless sensor networks using deployment knowledge", *Proc. IEEE Conf. on Computer Commun.* (*INFOCOM*), pp. 586-597, Mar. 2004.

[15] H. Chan and A. Perrig, "PIKE: peer intermediaries for key establishment in sensor networks," *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE* , vol.1, no., pp. 524- 535 vol. 1, 13-17 March 2005

VITA

Kothapalli Yashwanth

Candidate for the Degree of

Master of Science/Arts

Thesis: PRIMITIVE PYTHAGOREAN TRIPLES IN KEY MANAGEMENT OF SENSOR NETWORKS

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Master of Science in Computer Science at Oklahoma State University, Stillwater, Oklahoma in July, 2011.

Completed the requirements for the Bachelor of Engineering in Computer Science and Engineering at Jawaharlal Nehru Technological University, Hyderabad, India in 2009.

Experience:

Research Assistant — Jan 2010 – Dec 2010
Oklahoma State University — *Stillwater, OK*

- Maintained server running on Open Suse os
- Developed software Raid 5 file system

Name: Yashwanth Kothapalli Date of Degree: July, 2011

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: PRIMITIVE PYTHAGOREAN TRIPLES IN KEY MANAGEMENT OF SENSOR NETWORKS

Pages in Study: 33 Candidate for the Degree of Master of Science

Major Field: Computer Science

Scope and Method of Study:

This thesis deals with a new approach to key establishment in wireless sensor networks where the nodes have limited memory and communication capabilities. Primitive Pythagorean triples (PPT's) are used in this method in which we store odd numbers on sensor nodes and generate keys from them. This scheme needs at least one common odd number between two nodes to have a communication link, which increases the connectivity between nodes compared to q-composite scheme that requires at least q keys in common to have a communication.

Findings and Conclusions:

The scheme presented in this thesis provides a memory efficient way of generating keys using odd numbers and properties of PPTs. In this scheme, we established communication link between two nodes containing atleast one common odd number. Through simulations and findings, it is shown that this scheme provides more connectivity between the nodes compared to q-composite scheme without compromising node resiliency.

ADVISER'S APPROVAL: Dr. Subhash Kak