KEY DISTRIBUTION USING

PRIMITIVE PYTHAGOREAN TRIPLES

By

MONISHA PRABHU

Bachelor of Science in Computer Science

Jawaharlal Nehru Technology University

Hyderabad, Andhra Pradesh, India

2011

Submitted to the Faculty of the
GraduateCollege of the
OklahomaStateUniversity
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 2013

KEY DISTRIBUTION USING

PRIMITIVE PYTHAGOREAN TRIPLES

Thesis Approved:

Dr.SubhashKak

Thesis Adviser

Dr.Tingting Chen

Dr. David Cline

Name: MONISHA PRABHU

Date of Degree: MAY, 2013

Title of Study: KEY DISTRIBUTION USING PRIMITIVE PYTHAGOREAN TRIPLES

Major Field: COMPUTER SCIENCE

Abstract: This thesisstudies the randomness properties ofprimitive Pythagorean triples (PPTs)for different equivalence groups. The autocorrelation and cross-correlation functions of six classes based on divisibility by 3, 4, and 5 are derived from the gaps between each class type. It is shown that two of the classes are different from the other four classes in their randomness properties if they are ordered by the largest term. In the other two orderings each of the six random sequences has excellent randomness properties.The PPTs have also been divided into classes based on residues with respect to prime numbers and their randomness properties have been studied. Lastly, this thesis describes a method for the use of PPTs in key distribution for secure communication.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

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.

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.

In this thesis we propose a key self-establishment scheme using Primitive Pythagorean triples (PPTs) 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 and 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.

CHAPTER II

PYTHAGOREAN TRIPLES

Pythagorean triple is a triple of positive integers a, b, and c such that a right triangle exists with legs a, b and hypotenuse c by the Pythagorean Theorem. This is equivalent to finding positive integers a, b and c satisfying

$$a^2 + b^2 = c^2$$

which acts as the basis for trigonometry. We can generate infinity 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.

Note: For a PPT, a, b, c cannot all be even. Also, a, b cannot both be odd and c even, because then $a^2 + b^2$ is divisible by 2, whereas c^2 is divisible by 4. One of a and b must, therefore, be odd, and we will use the convention that b is even. Also note that the factors (c - b) and (c + b) of ($c^2 – b^2$) must both be squares because they cannot have common factors other than 1 for otherwise they would not be primitive. We can write a, b, c as

$$a = st$$

$$b = (s^2 – t^2) / 2$$

$$c = (s^2 + t^2) / 2$$

where s, t are odd and co-primes to each other and $c + b = s^2$ and also $c-b=t^2$. There exist an infinity of PPTs. The coordinate *(a/c, b/c)* may be seen as a point on the unit circle, implying that a countably infinity of these points are rational. A sequence that generates a subset of PPTs is $(2n+1, 2n^2+2n, 2n^2+2n+1)$ for $n = 1, 2, 3\ldots$

For example for the smallest and perhaps best known triple is (3, 4, 5), where s=3 and t=1. The Pythagorean triples a, b, c are divisible by either 3 or 4 or 5 separately or jointly so as the Primitive Pythagorean triples. This property is used and divided all the PPTs into six classes by Kak [4], who also presents their historical background. For additional background to Greek geometry, see [12],[13]. The Pythagorean result was known in India before the Greeks, and for a background on this information, see [14]-[17]. Random PPTs may also be used in lieu of other randomizing functions such as [18],[19].

Indexing Using s and t Numbers

For a convenient indexing one may use relatively prime *s* and *t* numbers in an array where $s > t$. This may be seen in the diagram shown below:

	s					
	3	5	7	9	11	13…….
1	(1, 3)	(1, 5)	(1, 7)	(1, 9)	(1, 11)	(1, 13)…
3		(3, 5)	(3, 7)	(3, 9)	(3, 11)	(3, 13)…
t 5			(5, 7)	(5, 9)	(5, 11)	(5, 13)…
7				(7, 9)	(7, 11)	(7, 13)…
9					(9, 11)	(9, 13)…
11						(11, 13)…

Table 1. Array of PPTs ordered by s and t numbers

If the indexing were done according to columns of Table 1, we have the following PPTs in the array across the first seven generations (where the duplicate values have been removed):

(3, 4, 5)					
(5, 12, 13)	(15, 8, 17)				
(7, 24, 25)	(21, 20, 29)	(35, 12, 37)			
(9, 40, 41)	(45, 28, 53)	(63, 16, 65)			
(11, 60, 61)	(33, 56, 65)	(55, 48, 73)	(77, 36, 85)	(99, 20, 101)	
(13, 84, 85)	(39, 80, 89)	(65, 72, 97)	(91, 60, 109)	(117, 44, 125)	(143, 24, 145)
(15, 112, 113)	(105, 88, 137)	(165, 52, 173)	(195, 28, 197)		
.	.	.	.	.	.	.	.

Table 2. Example generations of PPTs ordered by *s* and *t* numbers

The fourth row has only three entries as (2, 9) of Table 1, which corresponds to the triple (27, 36, 45) can be reduced to the PPT (3, 4, 5).

Indexing of the PPTs in Table 2 may be done according to increasing *a, b,* or *c*.

CHAPTER III

DIFFERENT WAYS OF REPRESENTING THE TRIPLE

A. Euclidean Pythagorean primitive triples:

The Euclidean Pythagorean primitive triples [12], [13] may be obtained using the formula

$a = m^2 – n^2$

$b = m^2 + n^2$

$c = 2mn$

Where m, n are relatively prime to each other and only of them is even and the other is odd.

B. Representing Pythagorean Triples as Gopal-Hemachandra Numbers:

Consider the Gopala-Hemachandra (GH) quadruple (g, e, f, h). The GH sequence, named after two mathematicians who lived before Fibonacci [4], is the sequence

$$g, e, g + e, g + 2e, 2g + 3e, 3g + 5e, \ldots$$

for any pair g, e. When g=1, e=1, we obtain the Fibonacci sequence.

In the GH quadruple (g, e, f, h), if

$a = g\,h$

$b = (c-a)\,f / e$

$c = e\,h + f\,g$

then (a, b, c) is a Pythagorean triple. If the quadruple has no common factors and g is odd, then

(a, b, c) is a PPT. The values b, and c may also be written as b=2ef, and c= $e^2 + f^2$.

C. A Variant of the Classical Formula

The Classical Greek formula: $(r^2 - s^2, 2rs, r^2 + s^2)$ is a Pythagorean Triple whenever $0 < s < r$; r, s Ɛ Z^+. The generating pair of integers in this case is denoted by [r, s] where, [r, s] = $(r^2-s^2, 2rs, r^2+s^2)$. The triple is primitive iff (r, s) =1 (are relatively prime) and (r – s) is odd. While (r, s) = 1 is a necessary condition for producing primitives, we must also require that (r - s) be odd. We shall refer to the collection of triples generated by [r, s] as the classical triples or classically generated triples.

The classical Pythagorean Triples can be generated by the relationships

$[p + q, q] = ((p + q)^2 – q^2, 2(p + q) q, (p + q)^2 + q^2)$.

$= (p^2 + 2pq, 2q^2 + 2pq, p^2 + 2q^2 + 2pq)$ ---------- (1)

where, p . q Ɛ Z^+.

D. Families of Triples

For each p Ɛ Z, we can define three continuous functions in the variable t as follows:

$A_p(t) = 2pt + t^2$; $B_p(t) = 2t^2 + 2pt$; $C_p(t) = 2t^2 + 2pt + p^2$---------- (2)

These three functions are a variant of the three functions in (1) replacing q with the more commonly used parameter t.

For fixed values of p, $A_p(t)$ is a linear function in while both $B_p(t)$ and $C_p(t)$ are quadratic functions in t.

E. Representing Pythagorean Triples using *[K M t]* System:

If (a, b, c) is a Pythagorean Triple, then c - b is called its hypotenuse-leg difference and is denoted by K. The triple can be expressed as

$(a, (a^2 - K^2)/2K, ((a^2 - K^2)/2K) + K)$

where

$K = D. E^2. L$, where

D = {product of the distinct odd factors of K}; each odd factor occurs exactly once,

E^2 = {product of even powers of the remaining factors including 2} expressed as $[\ ]^2$; each factor has power 2N (N an integer, N > 0), and

L = {product of the factors still left}; each factor occurs at most once.

Using this factorization of K, we define a new integer value M = 2 D E. M is called the co-value of K. We get the triple,

$(Mt + K, (M^2 t^2 + 2KMt)/2K, ((M^2 t^2 + 2KMt)/2K) + K)$ ---------- (3)

F. Comparing the Variant System [p + q, q] with the [K M t] System:

Like the formulas (2) for the Variant system, the formulas (4) for the [K M t] system can be parameterized so that they produce families of triples for each positive integer K:

$A_p(t) = MK + t$; $B_p(t) = (M^2 t^2 + 2KMt)/2K$; $Cp(t) = ((M^2 t^2 + 2KMt)/2K) + K$ ------- (4)

The generation of Pythagorean Triples using (2) can lend itself to the area of cryptology very nicely because two distinct related generating sets, such as { 1, 2,1} and { 2, 2, 1} will yield the 'same' triple (allowing transposition), in this case (3, 4, 5) . The various families of (4) (indexed by K) can be thought of as an infinite number of code wheels with each wheel (family) having infinite ways to code each letter of the alphabet – 'e' could be coded with some value t = 5mod 26 for instance. Together, a value chosen for K (wheel) plus the value selected for t will produce a triple (a ,b, c) ; remember that K will determine its necessary co-value M. If 'e' is the plaintext message, the representative cipher sent will be the first two terms of the triple, namely (a_e, b_e). The third value of the triple can easily be calculated from the other two terms and then Ke and t_e can be calculated to decode the cipher.

Example:

Preliminaries

- Create a table of M-values (one time only)
- Assign numeric values to the letters of the alphabet and single digits – t-values (one time only)
- Create a plain text code

Encryption

- Translate the plain text into its numeric equivalent (t-values)
- Corresponding to the first number(letter/t-value), select a random value for K
- Using (K, t, M), generate a corresponding PT triple (a, b, c)
- The encrypted text for the first letter is (a, b) or singly as (hex(a) hex(b))

Encrypt the remaining letters (numbers) of the plain text

Plain Text: H I

Assign t-values:

Plaintext	H	I
t – value	3	5

H → randomly choose K=3 (M=6)

then [K M t] = (3, 6, 3) →(21, 72, 75)

I → randomly choose K=2 (M=2)

then [K M t] = (2, 2, 5) →(12, 35, 37)

Cipher Text: 21 72 12 35 (a1, b1, a2, b2)

CHAPTER IV

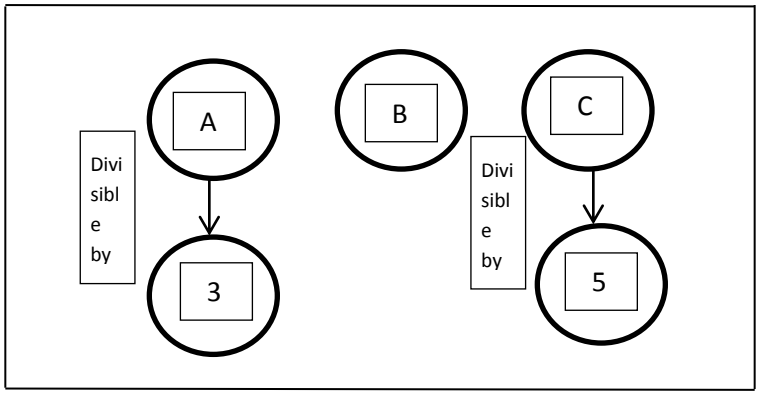
RANDOMNESS PROPERTIES OF SEQUENCES FROM SIX CLASSES

Six Classes of PPTs

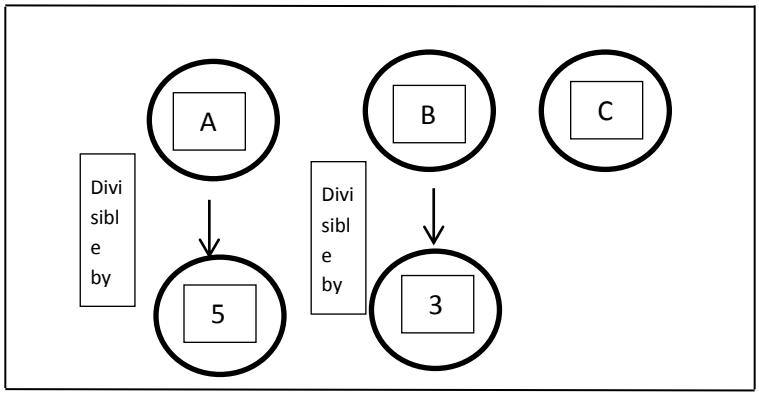
Theorem. *Primitive Pythagorean triples come in 6 classes based on the divisibility of a, b, c by 3, 4, and 5.*

The six classes are defined as follows:

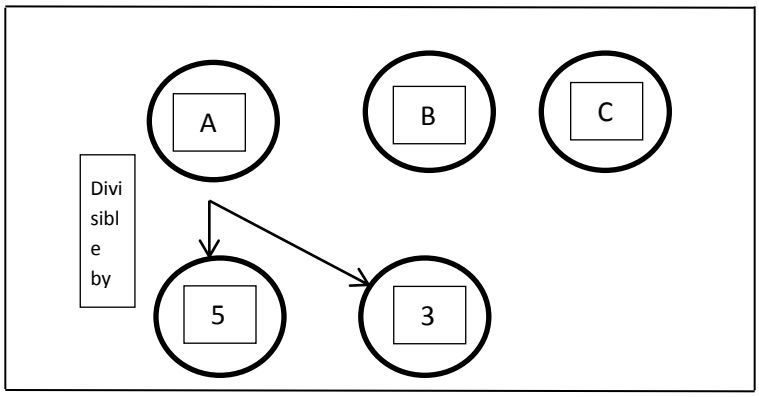
1. Class A in which a is divisible by 3 and c divisible by 5.



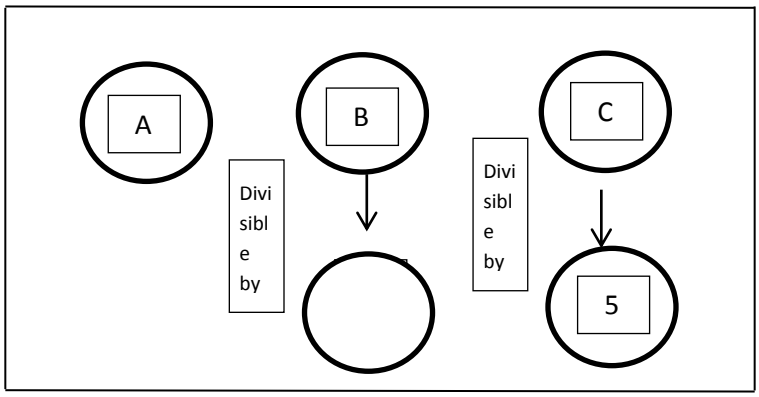
2. Class B in which a is divisible by 5 and b is divisible 3.



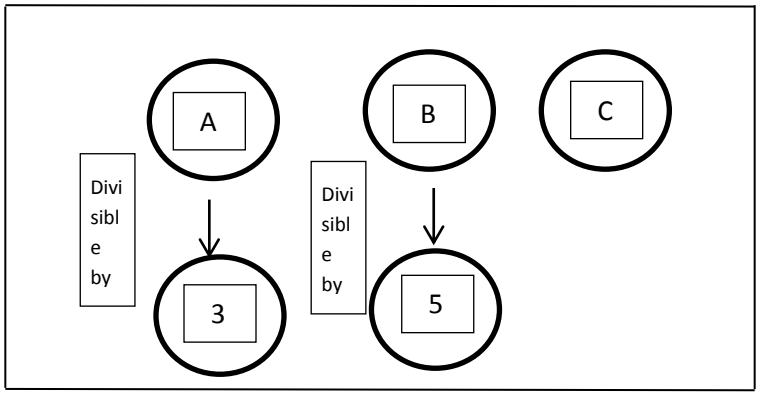
3. Class C in which a is divisible by 3 and 5.



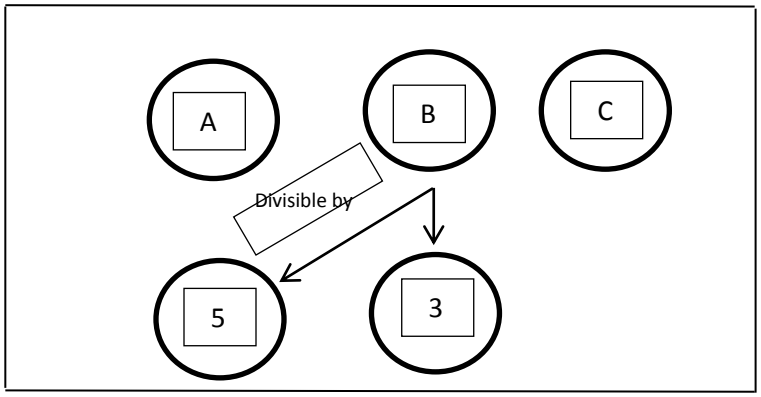
4. Class D in which b is divisible by 3 and c by 5.



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



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



The List of first 34 PPTs and the classes they belong to:

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

These are arranged in increasing order of hypotenuse length (c). If the hypotenuse lengths are equal then they placed according to the increasing order of ‘a’. The corresponding sequence of classes for the above 34 PPTs is ABCDEBECFAABDDEBEFCACDDEBFCFAABCDD.

The six classes may also be shown to be defined as the end nodes of the binary branching tree of figure 1.

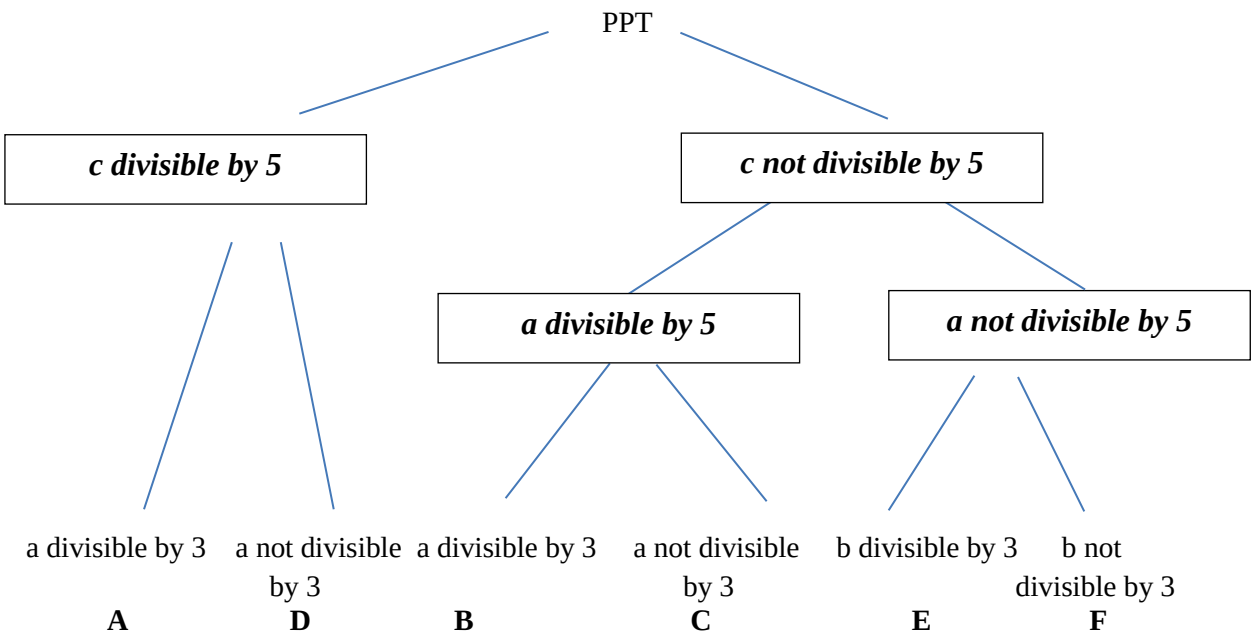


Figure 1.A graphical representation of the 6 classes of PPTs

An experiment was done where 4448 PPTs were generated and indexed by increasing a, *b*, and *c*, respectively.

Indexed by increasing a:

ABDEFDCCDFEEDBAFFAABBDEEFDCCDFEEDBBAAFFAABBDEEFDCCDDFEDBBAA
FFFAABBDEEFDCCCCDFEEDBBAAFFFAABDEEFDDCCDFEEDDBBAAFFAABBDEEFF
DCCCCDFEBBAAFFAABB…

Indexed by increasing b:

ACBBAEEDDCCCDDEEAABBCCAAFFFFACCBBAAEEDDDDCCCCDDEEAABBCCAAF
FFFAACBBBBAAEEEEDDCCCCDDDDEEAABBBBCCAAFFFFAACCBBAAEEDDDDCC
CCDDEEEEAABBBCCAAFF…

Indexed by increasing c:

ABCDEBECFAABDDEBEFCACDDEBFCFAABCDDEEFCFCDDEBEFCAABCDDBFCAAB
CCEBEFFAABDDEBEFFAABBDDEECCFAACDDEBCFBCDEEECFBCDDEEBBEFAABD
DBEFCAABBEFFAABCDD…

Randomness Properties of Sequences from the Six Classes

We obtain separate sequences related to the occurrence of As, Bs, Cs, Ds, Es, and Fs by considering the distance between occurrences of the letters. Thus in the listing by increasing *c*, A occurs, after its first value, at the 10th, 11th, 20th, positions, which corresponds to the numbers 9, 1,9, …. These sequences will be called *Baudhāyana sequences* after the author who used Pythagorean triples several centuries before Pythagoras [14],[15],[16],[17].

Baudhāyana sequences ordered by a:

Sequence of As: 14 3 1 17 1 3 1 17 1 4 1 19 1 4 1 18 1 3 1 20 1 3 1 18 2 10 11 1 1 1 9 3 10 2 9 1 1 1 10 2 11 1 12 2 12 2 1 1 10 2 11 2 13 1 13 1 9 1 11 1 1 10 1 1 21 11 1 3 21 12 8 1 3 1 9 11 1 2 10 1 12 1 1 9 18 2 1 9 …

Sequence of Bs: 12 6 1 13 1 7 1 13 1 8 1 15 1 8 15 1 7 1 16 1 7 1 15 4 8 2 9 5 7 5 8 4 7 5 6 1 1 4 9 3 10 4 1 1 8 6 8 4 9 4 11 3 9 1 1 3 11 8 4 1 1 5 5 8 1 1 2 7 3 8 6 8 1 1 1 8 2 8 4 6 7 7 3 1 1 6 5 8 4 8 1 4 6 1 9 3 1 4 1 26 ...

Sequence of Cs: 1 19 1 21 1 22 1 1 1 22 1 23 1 1 1 22 1 1 1 13 10 1 1 10 1 1 10 1 1 11 1 1 10 14 12 1 1 11 1 1 13 1 1 10 1 1 12 1 1 12 1 1 12 1 9 1 13 10 1 10 1 11 1 1 11 1 12 1 7 1 1 8 1 14 1 10 1 1 11 1 13 9 1 7 12 9 9 …

Sequence of Ds: 3 3 4 9 4 3 4 11 4 3 1 3 12 4 5 4 11 4 1 3 4 1 11 5 5 4 11 1 4 5 3 6 10 2 16 14 21 12 2 2 5 7 15 9 4 7 5 2 8 14 18 5 14 10 1 3 2 9 9 4 9 5 5 6 4 16 10 3 1 7 13 14 2 7 3 1 4 2 4 17 3 4 6 7 2 7 14 2 2 4 3 17 …

Sequence of Es: 7 1 11 1 7 1 13 1 8 14 1 9 1 13 1 8 1 14 1 10 1 14 1 9 8 3 1 1 5 6 7 5 7 5 7 5 7 4 8 1 1 4 7 1 1 5 8 5 9 6 7 5 7 7 1 1 5 7 7 1 4 7 4 1 23 11 1 5 1 18 7 1 1 12 1 9 10 23 1 5 6 24 1 1 7 12 1 13 6 9 1 3 1 5 1 5 6 …

Sequence of Fs: 5 6 1 8 5 8 1 8 6 7 1 1 8 7 8 1 1 7 6 9 1 8 1 7 8 1 9 7 5 5 14 16 1 7 5 4 12 8 26 9 5 14 13 4 5 9 5 10 7 6 11 2 2 15 2 8 12 1 3 4 18 7 1 5 6 1 12 8 4 8 5 3 1 1 26 1 5 4 3 6 1 9 2 8 4 4 1 9 6 5 13 1 1 9 3 1 13 2 7 …

Baudhāyana sequences ordered by b:

Sequence of As: 4 12 1 5 1 5 5 1 15 1 5 1 5 1 6 1 17 1 7 1 5 1 5 1 17 1 7 1 5 1 7 21 1 5 1 5 1 9 1 17 1 7 1 5 1 1 1 7 1 16 5 8 3 15 5 1 1 4 5 12 7 4 5 15 5 4 5 1 1 13 1 1 3 8 5 15 5 4 7 17 3 8 16 4 1 11 12 1 4 3 19 10 17 10 1 12 1 1 14 15 …

Sequence of Bs: 1 15 1 12 1 19 1 12 1 1 1 21 1 1 1 13 1 21 1 1 1 13 1 1 1 24 1 15 1 1 1 21 1 1 1 15 1 1 1 19 1 1 12 17 1 1 10 1 1 14 1 1 10 1 1 17 1 1 8 1 1 19 12 1 1 17 1 1 10 1 1 19 12 1 1 13 1 12 1 15 1 1 5 1 1 17 1 20 1 1 10 1 1...

Sequence of Cs: 8 1 1 9 1 8 1 11 1 1 1 9 1 9 13 1 1 1 13 1 9 1 11 1 1 1 13 1 9 1 14 1 1 1 13 1 9 1 1 1 11 1 1 1 15 1 11 1 13 1 1 1 10 10 9 1 11 8 9 11 1 1 6 11 1 1 1 9 6 13 1 9 10 11 1 11 6 1 1 11 1 1 1 9 10 7 1 1 9 10 7 1 1 27 9 1 1 4 …

Sequence of Ds: 1 4 1 24 1 1 1 5 1 28 1 5 1 1 1 27 1 1 1 5 1 32 1 1 1 5 1 1 1 31 1 5 1 1 1 35 1 5 1 1 1 22 1 1 3 1 1 22 1 1 2 1 1 24 1 1 5 24 1 1 3 1 1 24 1 1 3 1 1 24 1 1 5 1 1 21 1 4 1 1 22 4 1 19 1 1 1 1 2 1 1 16 1 1 4 1 22 1 2 1 1 22 …

Sequence of Es: 1 8 1 20 1 11 1 22 1 1 1 11 1 23 1 11 1 1 1 24 1 1 1 13 1 1 1 25 1 11 1 1 1 27 1 1 1 11 18 1 1 9 1 1 18 8 1 1 18 1 1 9 1 1 18 1 1 9 20 1 1 9 1 1 18 1 1 11 1 1 17 9 1 18 1 7 1 1 13 1 1 11 1 12 1 9 18 1 1 7 1 18 1 7 1 1 17 1 1 …

Sequence of Fs: 1 1 1 30 1 1 1 36 1 1 1 35 1 1 1 38 1 1 1 39 1 1 1 34 1 1 1 1 1 1 27 1 1 26 1 1 27 1 1 27 1 1 1 1 1 1 27 1 1 29 1 1 1 1 1 1 23 1 1 1 1 1 1 22 1 26 1 1 20 1 1 1 1 1 23 1 1 24 1 1 1 1 1 24 1 21 1 1 1 1 1 21 1 1 1 1 1 20 1 1 1 …

Baudhāyana sequences ordered by c:

Sequence of As: 9 1 9 9 1 18 1 8 1 9 1 9 1 10 1 26 1 8 1 8 1 9 1 9 1 8 1 18 1 11 1 19 1 7 1 8 1 19 1 8 1 7 1 9 1 16 1 21 19 1 5 1 11 1 6 1 11 1 17 1 1 1 6 1 1 1 8 1 30 1 17 1 9 1 18 1 7 1 8 31 1 8 1 6 1 10 1 1 1 9 1 17 1 7 …

Sequence of Bs : 4 6 4 9 6 13 6 4 5 4 6 4 6 1 14 3 8 6 1 5 3 6 1 8 4 9 10 6 8 7 4 12 3 10 18 1 5 8 1 5 1 7 10 5 3 1 6 5 12 11 11 4 6 6 6 6 1 1 10 1 6 9 1 2 11 6 9 1 4 5 4 10 7 11 7 4 8 5 5 1 8 14 1 13 5 4 1 8 9 7 6 11 13 4 7 2...

Sequence of Cs: 5 11 2 6 5 6 2 7 4 5 4 1 24 1 4 5 3 5 3 17 5 8 16 3 5 5 1 4 1 2 11 1 6 5 8 6 1 8 10 10 3 1 6 8 4 9 5 16 1 5 7 4 1 6 4 16 1 6 7 4 8 5 6 4 4 5 10 24 1 9 11 1 7 1 4 5 1 3 8 6 1 12 1 9 15 1 2 7 1 17 16 1 5 4 4 4 6 …

Sequence of Ds: 9 1 8 1 10 1 7 1 10 1 17 1 10 1 9 1 7 8 1 10 1 18 1 7 1 10 1 17 1 1 1 9 1 18 1 9 1 7 1 9 1 10 1 7 1 27 1 8 1 8 1 1 1 9 1 9 1 16 1 16 1 12 1 18 1 18 1 8 1 6 1 1 1 11 1 6 1 11 1 16 1 19 1 8 1 1 1 5 1 9 1 20 1 …

Sequence of Es: 2 8 2 7 11 1 7 2 17 2 8 2 9 1 9 7 1 1 7 1 3 8 8 1 10 7 2 18 1 10 10 9 2 1 7 2 8 8 10 1 1 18 7 1 8 3 8 1 2 1 16 1 9 1 1 15 1 9 2 15 1 12 9 11 7 1 10 2 1 6 2 7 1 11 1 6 2 11 6 10 8 1 2 9 11 1 5 2 10 1 7 1 1 9 9 …

Sequence of Fs: 9 8 2 9 2 7 9 10 1 9 1 11 9 8 10 8 9 1 9 1 8 1 8 2 8 11 9 1 2 9 1 1 9 6 2 8 10 10 8 2 7 1 10 11 1 5 12 1 17 1 1 6 12 7 10 2 8 1 2 6 19 1 1 11 1 16 2 1 8 8 18 1 1 1 10 7 1 8 1 11 1 6 13 7 1 1 18 10 2 5 1 2 1 …

We consider the autocorrelation function, *C (k),* of these sequences when written as *a(i):*

$$C(k) = \frac{1}{n}\sum_{i=0}^{n} a(i)a(i+k)$$

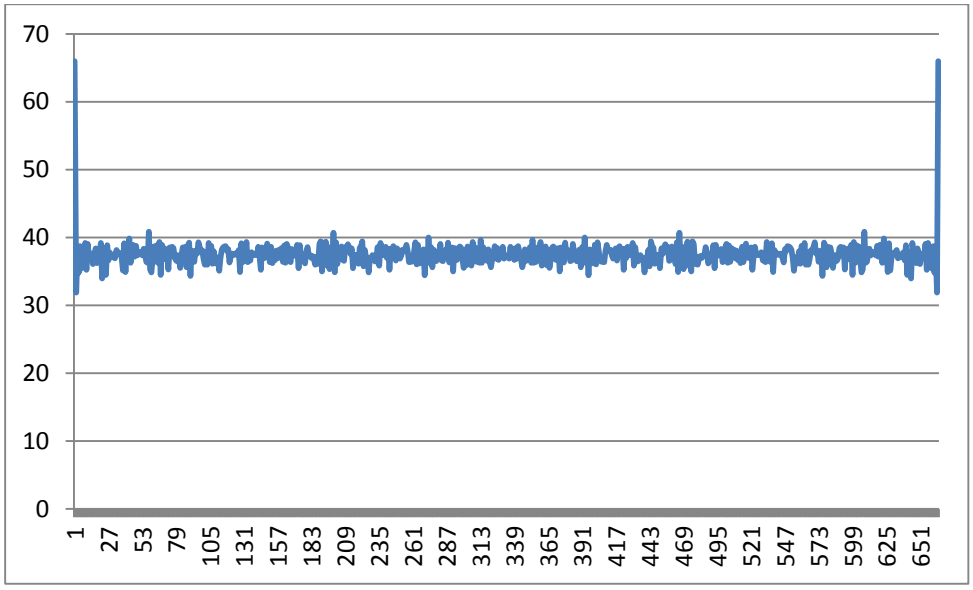


Figure 2. The typical autocorrelation function for Baudhāyana sequences ordered by *a* and *b*

The autocorrelation function for each of the six classes of Baudhāyana sequences ordered by *a* and *b* is qualitatively the same (Figure 2). Since there are 6 classes, the average distance between consecutive points will be 6. This, in turn, makes the function for non-zero values of the argument to be approximately 36 as we find in the plots.

The autocorrelation functions of the six random Baudhāyana sequences for ordering by *c* is shown in Figure 3. Notice that the value of the autocorrelation function for zero lag is not the same for all sequences. The functions for classes B, C, E, and F are similar to the results in Figure 2. However when the Baudhāyana sequences are arranged by order of *c*, As and Ds bunch together as there are more than one solution for values of *c* that are divisible by 5. These are represented by classes A and D. This is the reason why the plots for these two classes are different from the others as shown in Figure 3. This means that the correlation of Baudhāyana sequences ordered by c as shown in Figure 3a and Figure 3d is an artifact of the ordering process and the six sequences have excellent randomness properties if we order them by *a* or *b* or if we consider the classes B,C, E, and F when they are ordered by *c*.

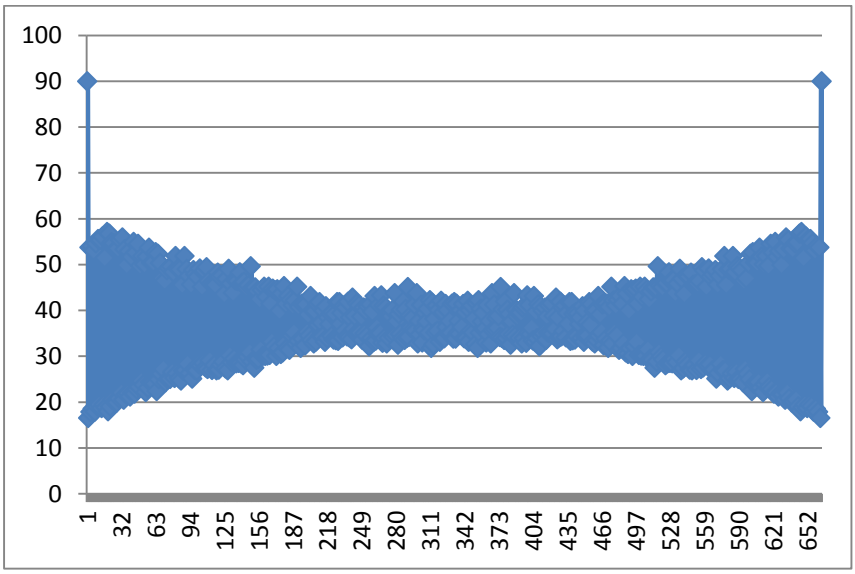


Figure 3a. Autocorrelation of Class A Baudhāyana sequences ordered by c

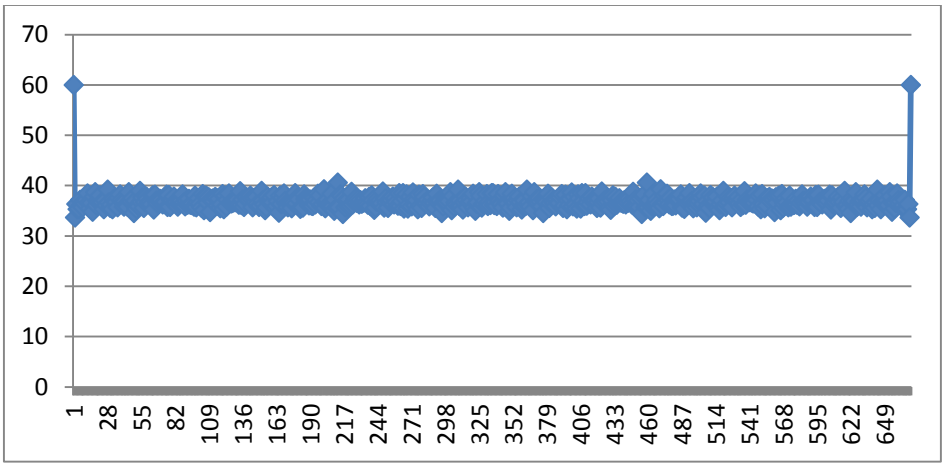


Figure 3b.Autocorrelation of Class B Baudhāyana sequences ordered by *c*

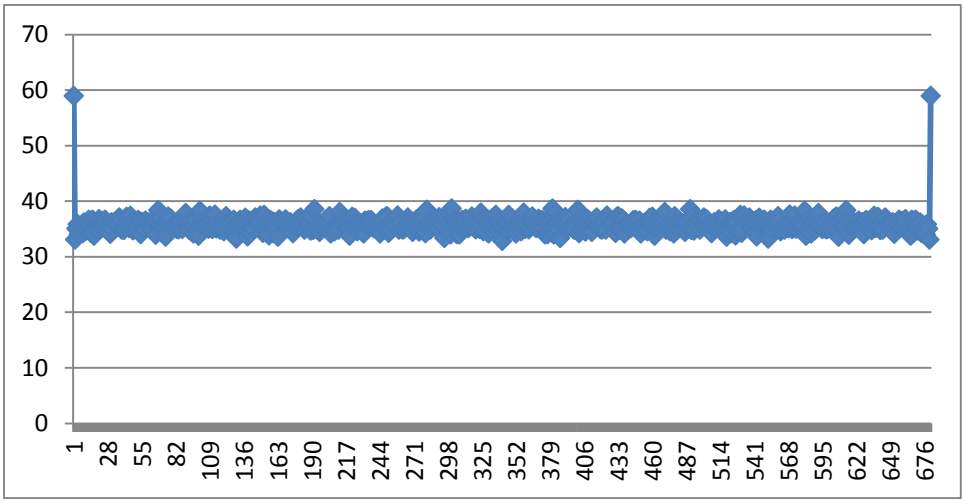


Figure 3c.Autocorrelation of Class C Baudhāyana sequences ordered by *c*

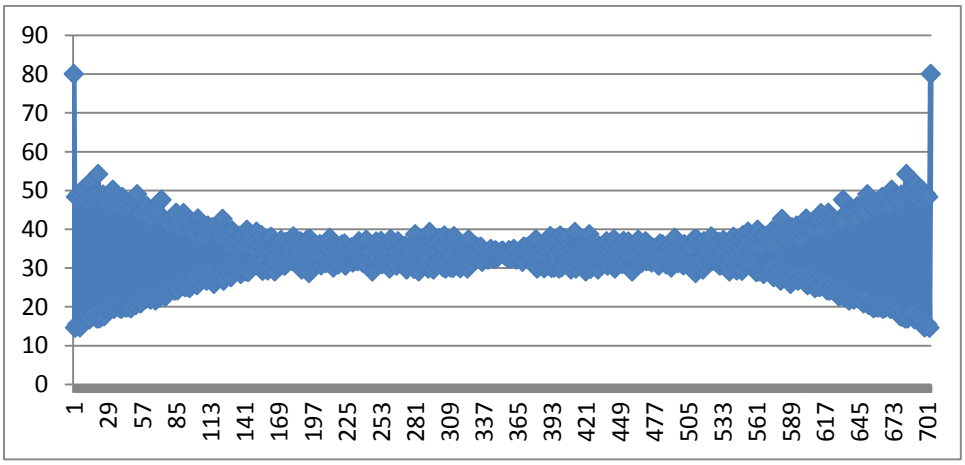


Figure 3d.Autocorrelation of Class D Baudhāyana sequences ordered by *c*

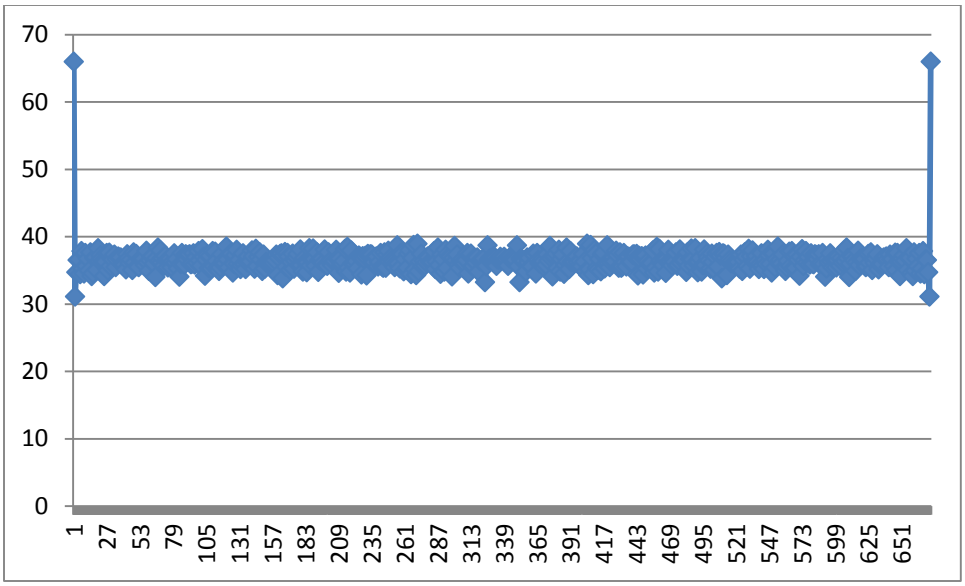


Figure 3e.Autocorrelation of Class E Baudhāyana sequences ordered by *c*

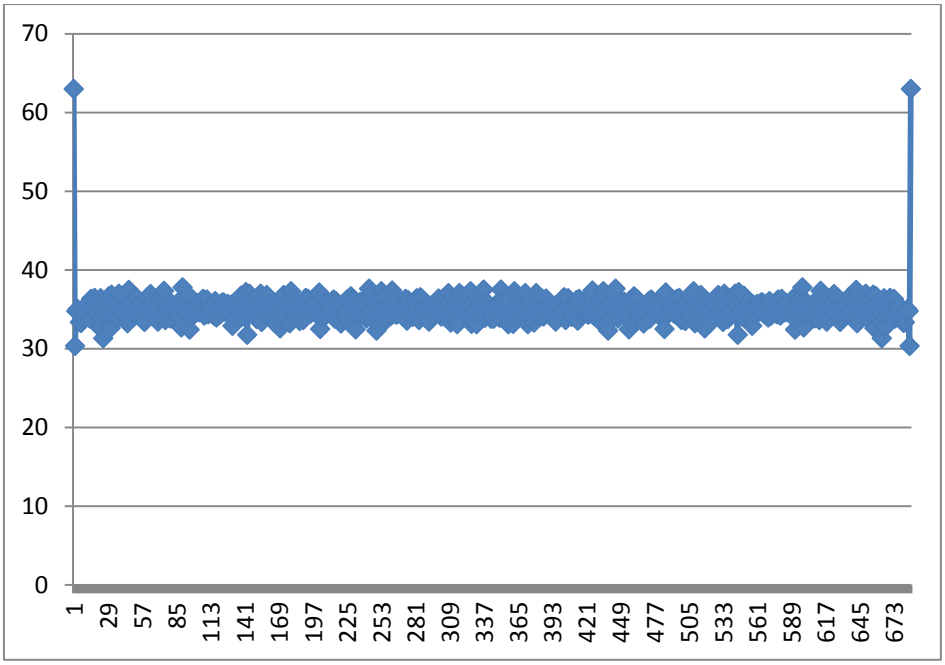


Figure 3f. Autocorrelation of Class F Baudhāyana sequences ordered by *c*

The A to F sequences may alternatively also be mapped into binary sequences and their properties remain qualitatively similar to the results of Figures 2 and 3. For example the autocorrelation of the binary sequence for Class E is shown in Figure 4 below.

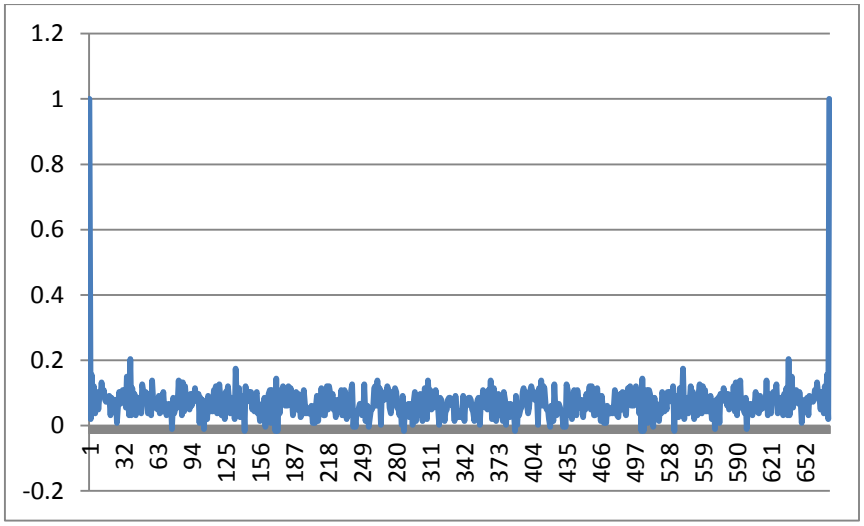


Figure 4. Autocorrelation of Class E for binary sequence

Cross Correlation Properties of the Baudhāyana sequences

We consider the autocorrelation function, *H (k)*, of these sequences when written as *x (i) and y (i):*

$$H_{x,y} = \frac{1}{N}\sum_{i=0}^{N} x(i)y(i+k)$$

Where,

$$x, y \in Six\ Classes$$

Computation of the cross correlation functions reveals that the cross correlation is relatively high between A and D (Figure 5) when the Baudhāyana sequences are ordered according to *c* as is to be expected. The cross correlation values between other pairs of Baudhāyana sequences are low (typically like that of Figure 6). This validates our assessment that the sequences possess excellent randomness properties.

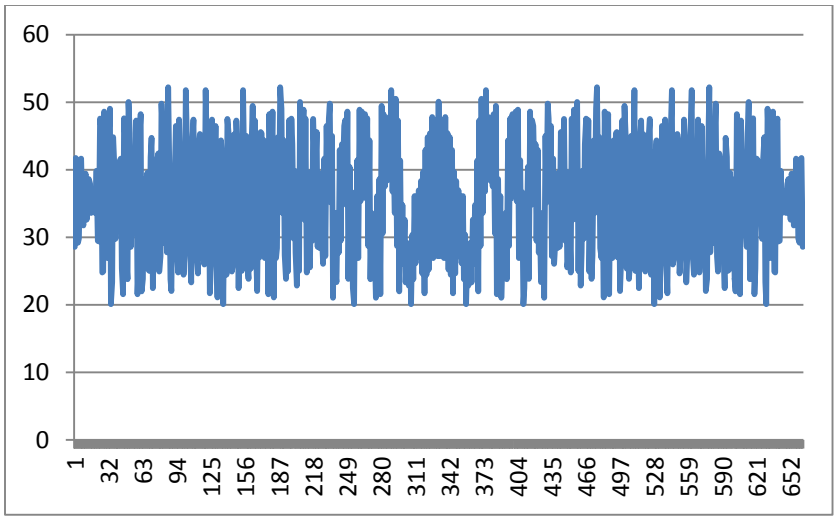


Figure 5. Cross correlation between A and D (the values range between 20 and 53)

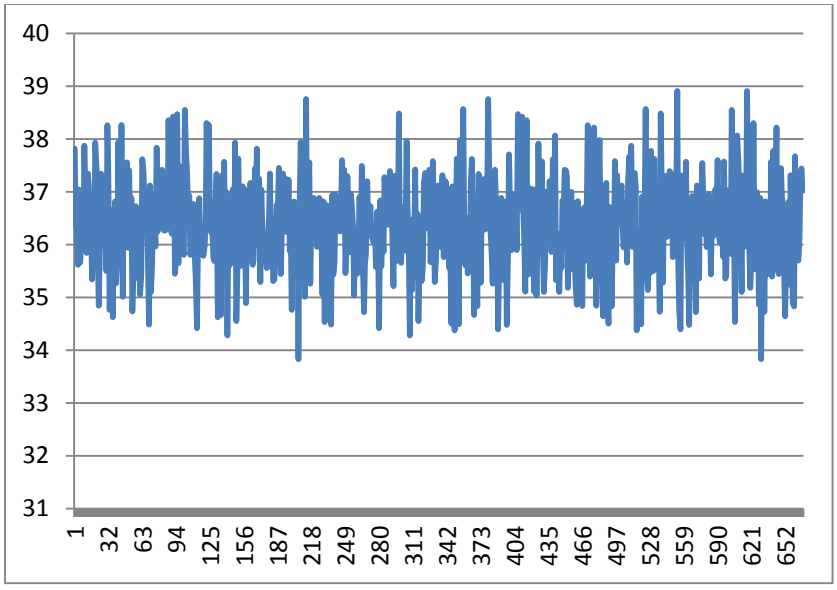


Figure 6. Cross correlation between B and C (the values range between 34 and 39)

CHAPTER V

CLASSIFICATION USING RESIDUES

The PPTs may be put into different equivalence classes using residues with respect to different primes.We have determined the autocorrelation function of the residue classes and as the graphs below show these sequences have excellent randomness properties.

Example 1. Consideringresidues of *a* with respect to 3 in the increasing order of *c*

If a mod 3 == 0; Class A,

If a mod 3 == 1; Class B,

If a mod 3 == 2; Class C

The sequence obtained when indexed by increasing c:

ACAACAABAACACABAAACCACAAACAAABACACAABAAAACCAAACAAAABCAA

CCACABAACCAAAABAAAACABCACAAAAAAACCAAACCCABAAACAAACBAAAC

ABCAACACABBAAAACACAAAAA……

Autocorrelation for A:

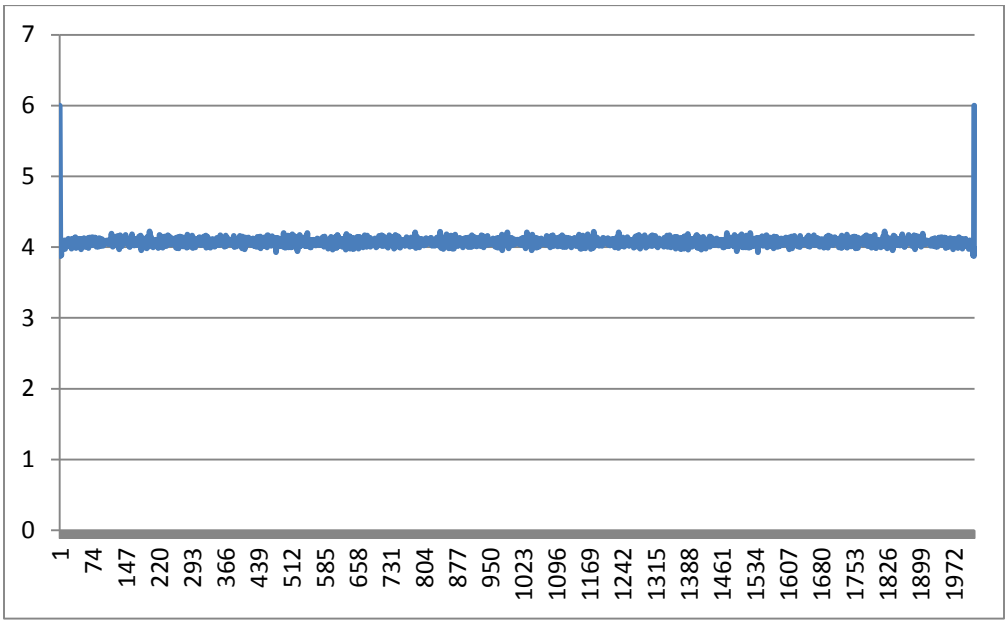


Figure 7. Autocorrelation of Class A for residue with respect to 3

Autocorrelation for B:

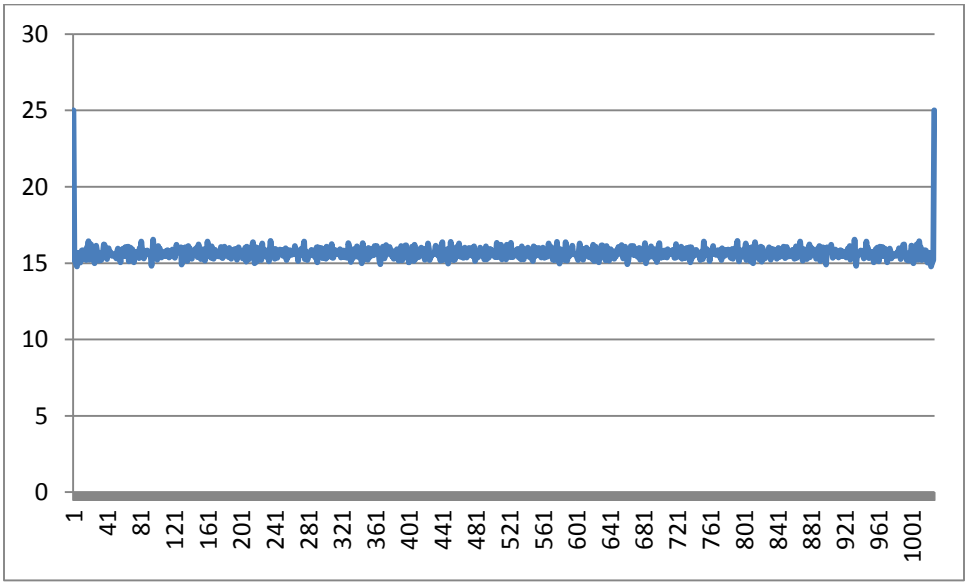


Figure 8. Autocorrelation of Class B for residue with respect to 3

Autocorrelation for C:

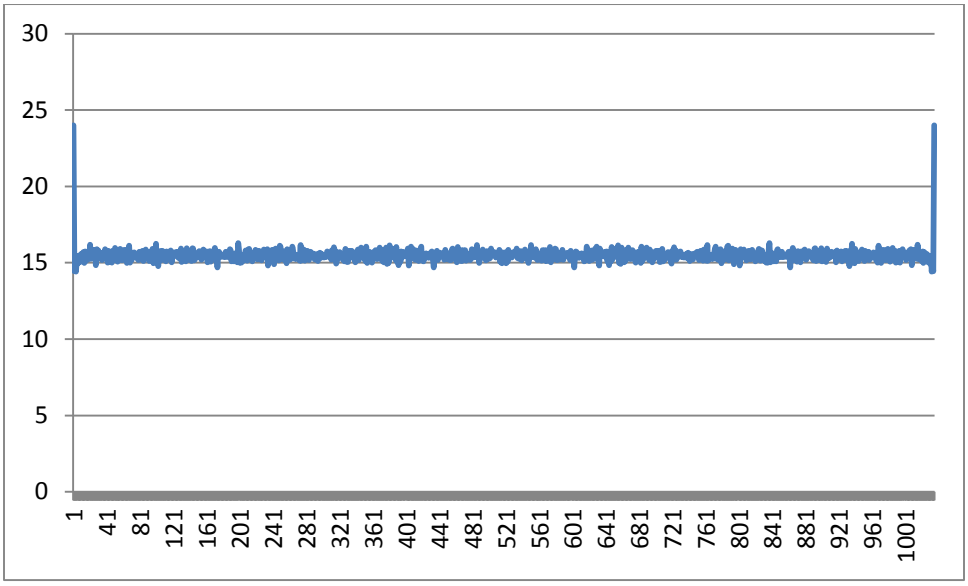


Figure 9. Autocorrelation of Class C for residue with respect to 3

Example 2. Consider residues of *a* with respect to 5 in the increasing order of *c*

If a mod 5 ==0; Class A,

If a mod 5 ==1; Class B,

If a mod 5 ==2; Class C,

If a mod 5 ==3; Class D,

If a mod 5 ==4; Class E,

The sequence obtained when indexed by increasing c:

DAACBAEABDDADCEAEBACACDBAEAECDAACDBBBAEACDEABBACDAADDAEA

CCAAAEAEBECCADCBABBEDDAACCEEAABDDADDBAABAACEBEAEAACCBBAAB

EDCACDAEBADDAAAEEEBDCAADCAEBECDDDEABBBACC……..

Autocorrelation for A:

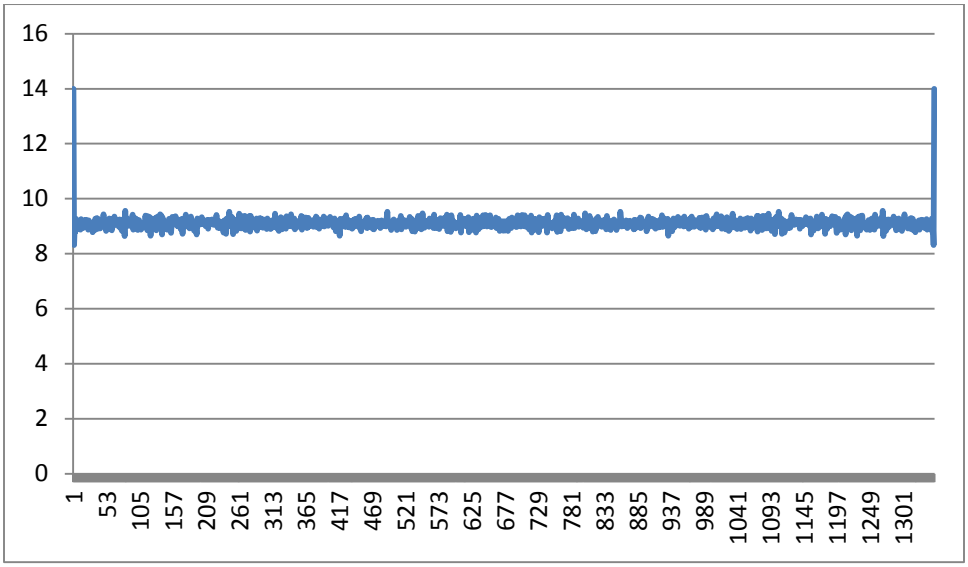


Figure 10. Autocorrelation of Class A for residue with respect to 5

Autocorrelation for B:

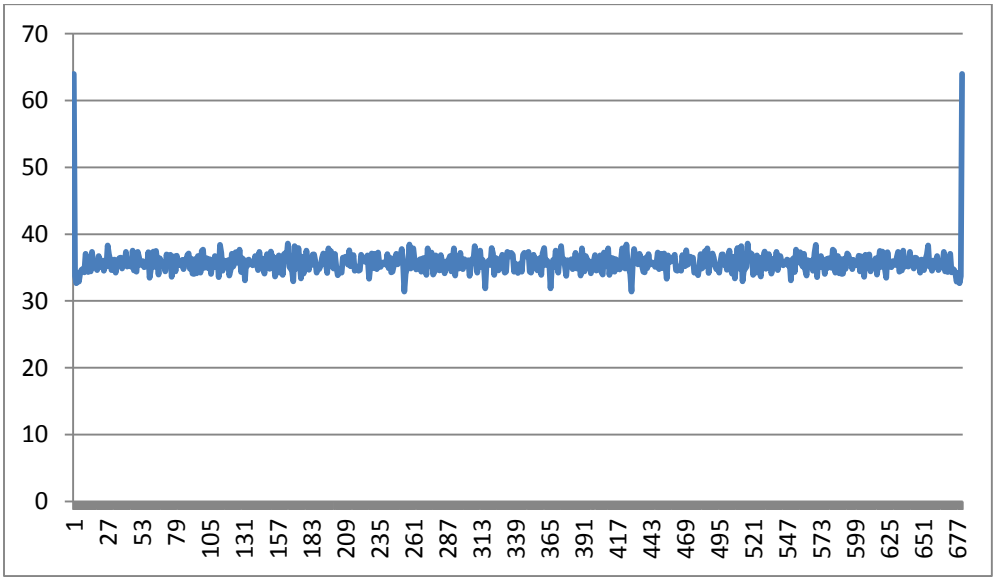


Figure 11. Autocorrelation of Class B for residue with respect to 5

Autocorrelation for C:

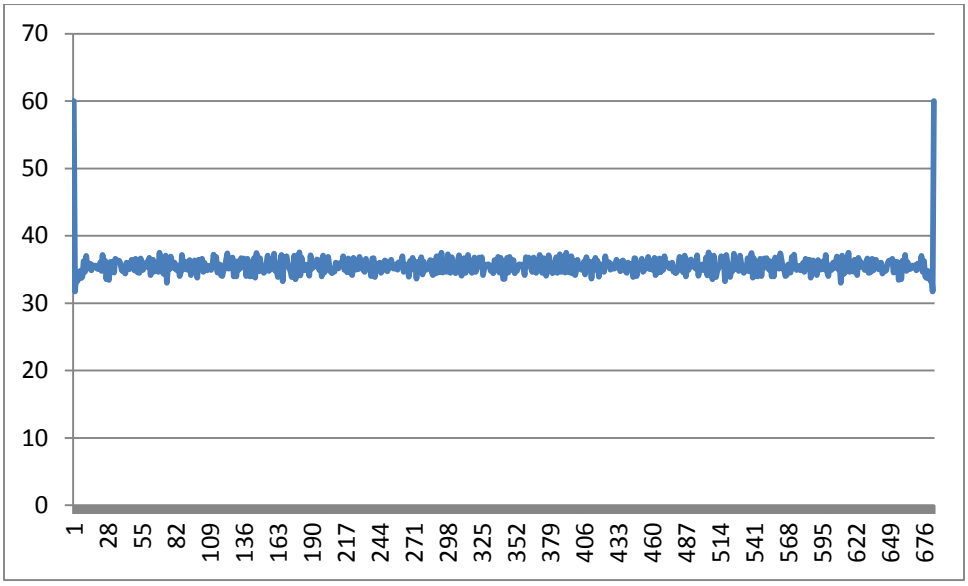


Figure 12. Autocorrelation of Class C for residue with respect to 5

Autocorrelation for D:

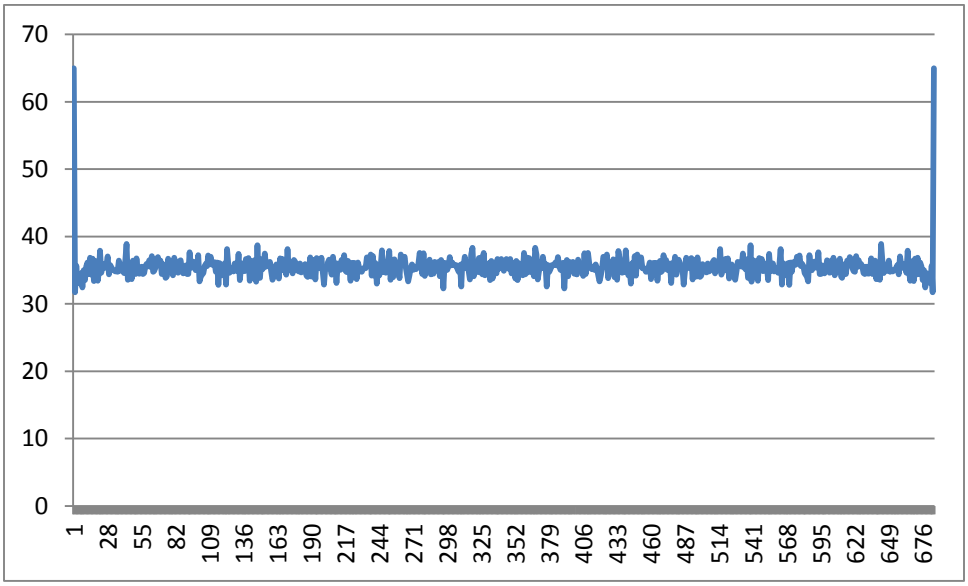


Figure 13. Autocorrelation of Class D for residue with respect to 5

Autocorrelation for E:

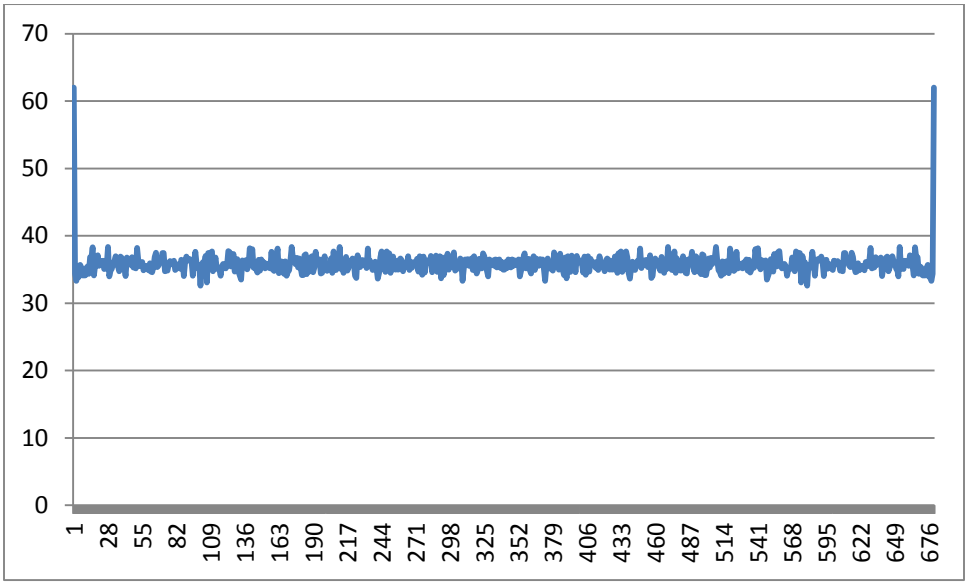


Figure 14. Autocorrelation of Class E for residue with respect to 5

Example 3. Consider residues of *a* with respect to 7 in the increasing order of *c*

If a mod 7 == 0; Class A,

If a mod 7 == 1; Class B,

If a mod 7 == 2; Class C,

If a mod 7 == 3; Class D,

If a mod 7 == 4; Class E,

If a mod 7 == 5; Class F,

If a mod 7 == 6; Class G,

The sequence obtained when indexed by increasing c:

DFBAAACDEFAGGAECBABFADDCBAEFBGEGFAADEAGDCCGDAAFEAEFBBAFBGA

CCCADAGBDDFADBFCDECAGBGAEGBAFFCEEDABDFAACCFAABADCCBGDAAAE

EEBGAEFGAGFACDCGGEECGDAAFDDAFBBDFGEAG.....

Autocorrelation for A:

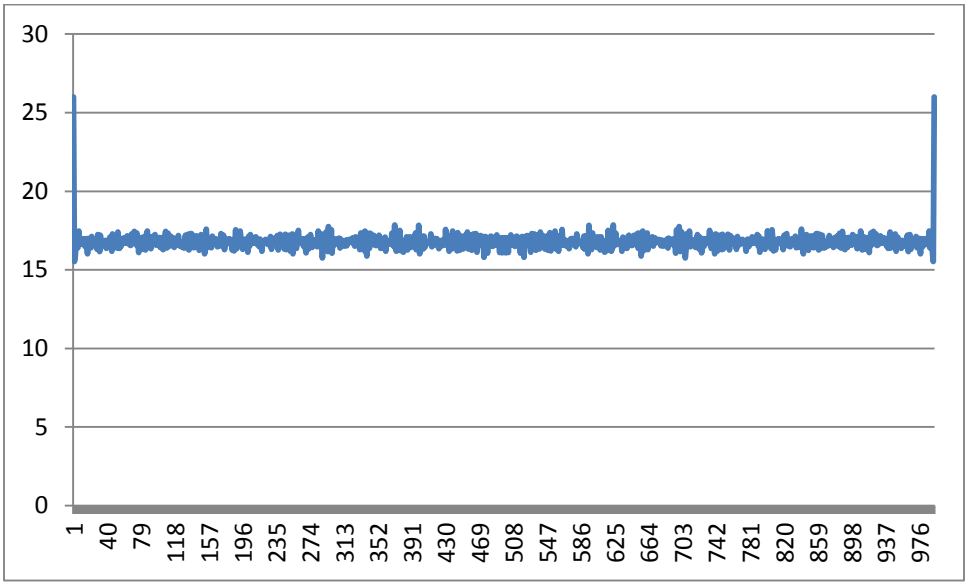


Figure 15. Autocorrelation of Class A for residue with respect to 7

Autocorrelation for B:

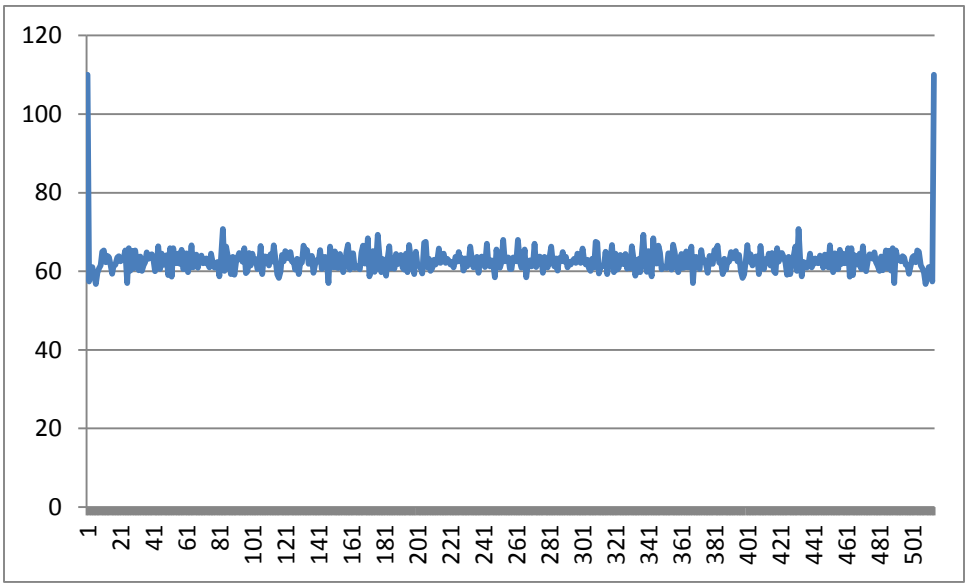


Figure 16. Autocorrelation of Class A for residue with respect to 7

Autocorrelation for C:

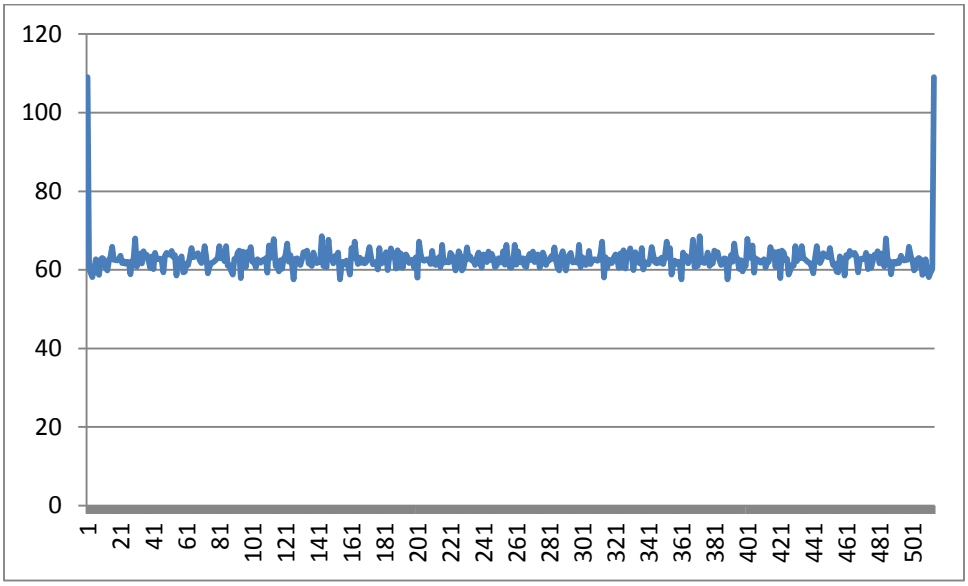


Figure 17. Autocorrelation of Class C for residue with respect to 7

Autocorrelation for D:

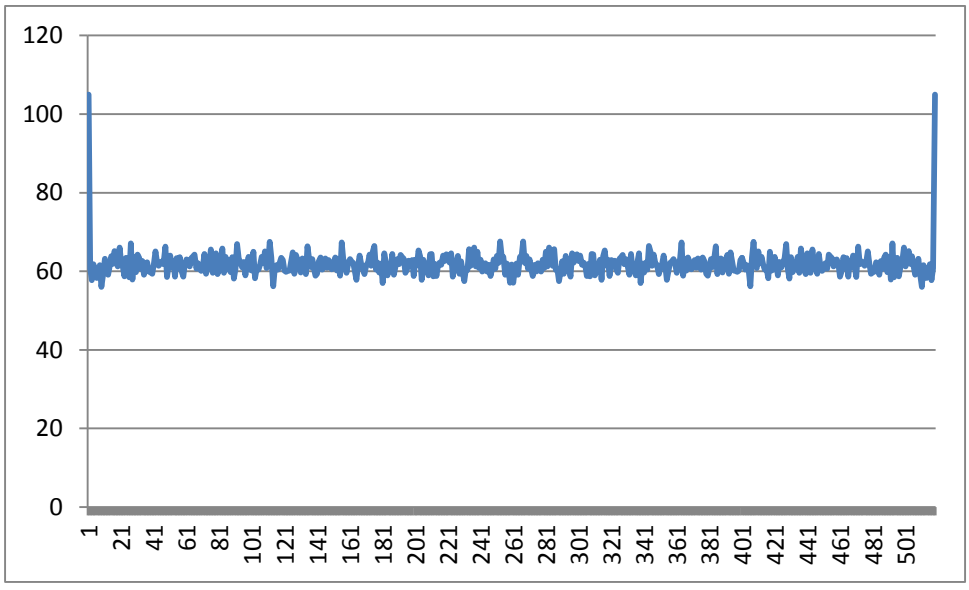


Figure 18. Autocorrelation of Class D for residue with respect to 7

Autocorrelation for E:

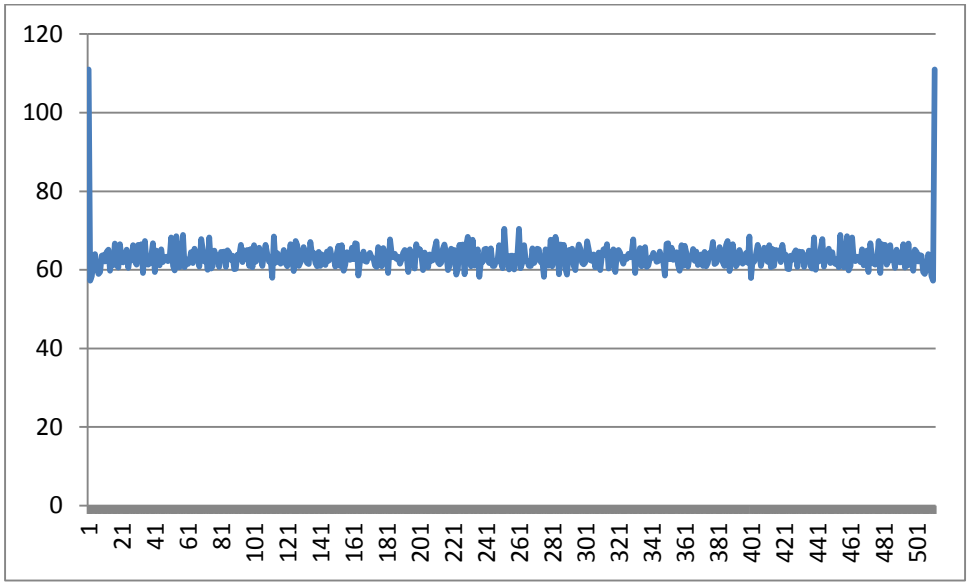


Figure 19. Autocorrelation of Class E for residue with respect to 7

Autocorrelation for F:

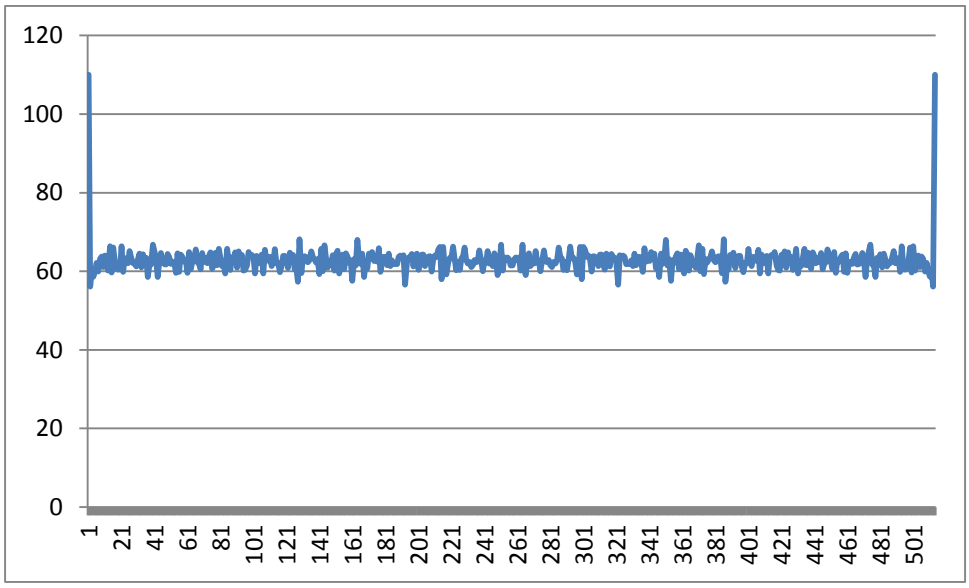


Figure 20. Autocorrelation of Class F for residue with respect to 7

Autocorrelation for G:

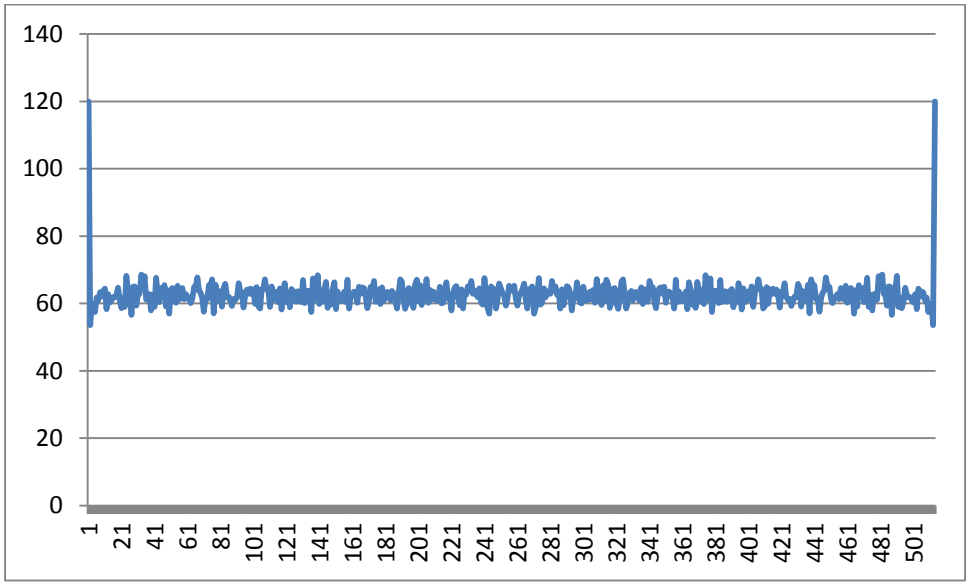


Figure 21. Autocorrelation of Class A for residue with respect to 7

All the class sequences of the PPTs are generated by various methods show excellent randomness properties and therefore can be used in many cryptographic applications.

Furthermore, in the above cases the average auto-correlation values for class A sequence is almost one-fourthof the average autocorrelation values of the rest of the classes. In order to find the reason behind it, we first compute the frequency of various residues.

Length of sequence	Number of A's in the sequence for residues of *a*		
	with respect to 3	with respect to 5	with respect to 7
1000	514	341	255
2000	1035	672	505
3000	1537	1004	751
4000	2017	1349	992

Table 3.Number of A's in the sequence for residues of *a* with respect to 3, 5 and 7.

OBSERVATIONS:

Consider residues of a's with respect to 3:

For *n* long sequences, the number of A's in the sequence is almost n/2. Therefore, the average distance between two As is 2.The autocorrelation is given as $C(k) = \frac{1}{n}\sum_{i=0}^{n} a(i)a(i+k)$ which makes the function for non-zero values of the argument to be approximately 4as we find in the plots. The same is the case with the number of Bs and Cs in the sequence where the average distance between two Bs and Cs is 4. Thismakes the function for non-zero values of the argument to be approximately 16as we find in the plots.

Consider residues of a's with respect to 5:

For *n* long sequences, the number of A's in the sequence is almost n/3. Therefore, the average distance between two As is 3. The function for non-zero values of the argument will be approximately 9as we find in the plots.The function for non-zero values of the Bs, Cs, Ds and E's will be approximately 36as we find in the plots.

Consider residues of a's with respect to 7:

For *n* long sequences, the number of A's in the sequence is almost n/4. Therefore, the average distance between two As is 4. The function for non-zero values of the argument will be approximately 16as we find in the plots.The function for non-zero values of the Bs, Cs, Ds, Es, Fs and Gs will be approximately 64 as we find in the plots.

CHAPTER VI

KEY DISTRIBUTION USING PPTs

Consider an odd number which can be represented by a sum of two numbers such that the two numbers are co-primes. i.e., the gcd of these numbers should be 1. These two numbers are (s, t) from which the primitive Pythagorean triple can be calculated as follows:

$$b = 2 s t$$

$$c = s^2 + t^2$$

$$a = s^2 - t^2$$

For a secure communication between two nodes, a secret key is to be agreed between the two nodes. By using the Sequences generated by the PPTs this secret key can be generated as followS

Method:

1. Assuming the central authority knows Alice's and Bob's secret key.

2. The central authority tries to send the secret key to both Alice and Bob using three different parameters namely, the sequence, the mod value generating the sequence and one of the 's' values used to generate the key(odd number).

3. These parameters can be encrypted as follows:

Sequence: the sequence can be encrypted using Transposition.

> Transposition cipher: simple data encryption scheme in which plaintext characters are shifted in some regular pattern to form cipher-text.

Mod value and 's' value: these can be encrypted by performing xor operation of the secret key and the private key.

4. Once these parameters have been encrypted and sent, these values will be decrypted by Alice and Bob.

5. The key is calculated by generating all possible t values such that $t < s$ and $gcd(s, t) = 1$. Sum the (s, t) value to generate odd number. For this odd number generate all possible (s, t) pairs and respective PPTs and their resulting sequence and cross verify this sequence with the sequence sent by the central authority. If it matches then that the sum of (s, t) is the secret key.

Example:

Consider the Secret key selected by the CA is 11.

For 11, the (s, t) values, PPTs and Classes are:

The Class for the PPT is generated by using (a mod 5):

if a mod 5 == 0, then Class A;

if a mod 5 == 1, then Class B;

if a mod 5 == 2, then Class C;

if a mod 5 == 3, then Class D;

if a mod 5 == 4, then Class E;

KEY	(s, t)	PPT	Class
11	(6, 5)	(11, 60, 61)	B
	(7, 4)	(33, 56, 65)	D
	(8, 3)	(55, 48, 73)	A
	(9, 2)	(77, 36, 85)	C
	(10,1)	(99, 20, 101)	E

Selecting the *s* value to be 8, the CA will encrypt the three parameters as follows

Encryption:

Sequence: Transposition Cipher

First select a key.

Write the message letters out over a number of rows then read off cipher column by column. Fill out the empty spaces with *.

Suppose the key is: 3 1 2 (this key is supposed to be secret)

Plain-text is: BDACE

The Arrangement would be:

3	1	2
B	A	E
D	C	*

Cipher Text is: ACE*BD

Modulo and *s* value encryption:

Here we have considered the modulo value to be 5 and the secret key is 11. Therefore the encrypted modulo value passed will be (secret key) ⊕ (Private key)

Secret key: 1 0 1 1

Alice Private Key: ⊕ 1 0 1

1 1 0 1

In the same way the modulo value is sent to Bob. The *s* value selected is also sent in the same manner.

Thus,

Parameters sent to Alice will be: (ACE*BD, 1101, 1101)

Parameters sent to Bob will be: (ACE*BD, 1101, 1110)

Decryption:

Once Alice and Bob receive the ciphered parameters from the CA, they decrypt it as follow:

Sequence:

Using the secret key 3 2 1, the sequence is decrypted by filling out the columns in sequential manner in the following way:

3	1	2
B	A	E
D	C	*

Modula and *s* value:

Modulo/ s = (passed modulo/ s) $\oplus$ (private key)

Now to generate the key:

Both Alice and Bob need to generate (s, t) pairs: (8, 7), (8, 5),(8, 3), (8,1) from the given *s* value i.e., 8. Once they find all the possible pairs, sum each pair to generate the odd number. For this odd number, all possible (s, t) pairs are generated and PPTs are generated. For each *a* value in the PPT,*(a mod (mod value passed by CA))* is calculated to generate a class sequence for the odd number. If this generated class sequence matches the sequence sent by the CA, the odd number generating this sequence is the key.

Analysis:

Transposition Cipher:

A plaintext will be given as **P** = {a_1, a_2 …a_p}. The transposition cipher permutes a block of n consecutive characters of Plaintext according to an n-permutation P, to produce a cipher text block($c_,$, …c_n). Therefore the number of permutations depends on the number of columns chosen to encrypt the plain-text. Besides using a key, multiple transpositions could be performed.This would be an example of "multiple stages" encryption.The result is a more complex permutation that is not easy to determine. To make transposition ciphers a bit more secure, it is usual to remove all punctuation marks from the plaintext first. It is quite often the case that all spaces are also removed.

Time taken to Generate (s, t) pairs:

The secret key used for communication by the nodes is sent using three different parameters by the CA. This increases the difficulty of the eavesdropper as he needs to know all the three parameters in order to retrieve the key. Knowing only one or even two parameters, will be of no use to the eavesdropper as the key can be generated only if all the three parameters will be known. Furthermore the parameters sent to the nodes by the CA are encrypted using different techniques.

Time taken generated the (s, t) pairs to know the key:

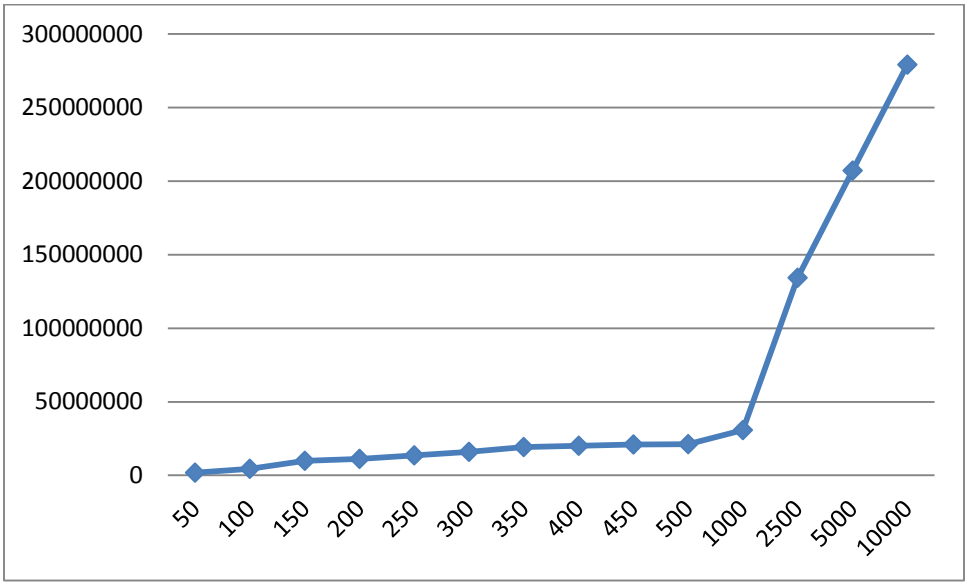


Figure 22.Time taken generated the (s, t) pairs

X - Axis: *s* values ranging from 0-500.

Y - Axis: Time taken to generate (s, t) pairs in nanoseconds.

CHAPTER VII

CONCLUSION

This thesis initially shows how the six classes of PPTs can be put into two larger classes. Specifically, autocorrelation and cross-correlation functions of the Baudhāyana sequences of the six classes have been computed. It is shown that Classes A and D (in which the largest term is divisible by 5) are different from the other four classes in their randomness properties if they are ordered by c. But if the Baudhāyana sequences are ordered by a or b, each of the six classes exhibits excellent randomness properties. This remains true if binary mappings of the Baudhāyana sequences are considered. Further when we divide the PPTs into residue classes they show excellent randomness properties for all the classes. This thesis also explains how this random Baudhāyana sequences can be used in key exchange protocols and how the level of difficulty in knowing the key is enhanced by passing the key as three different parameters. The random sequences obtained using PPT may also be used in other communication systems.

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. 33

[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, 1956.

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

[14] J.J. O’Conner and E.F. Robertson, Baudhayana. History of Mathematics Project. http://www-history.mcs.st-and.ac.uk/~history/Biographies/Baudhayana.html

[15] A. Seidenberg, The origin of mathematics. Archive for History of Exact Sciences 18: 301-42, 1978.

[16] S. Kak, The Astronomical Code of the Rgveda. Oklahoma State University, Stillwater, 2011.

[17] S. Kak, On the chronology of ancient India. Indian Journal ofHistory of Science22: 222-234, 1987.

[18] A. Parakh and S. Kak, Online data storage using implicit security. Information Sciences 179: 3323-3331, 2009.

[19] A. Parakh and S. Kak, Space efficient secret sharing for implicit data security. Information Sciences 181: 335-341, 2011.

VITA

Monisha Prabhu

Candidate for the Degree of

Master of Science

Thesis: KEY DISTRIBUTION USING PRIMITIVE PYTHOGEREN TRIPLES

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Master of Science in Computer Science at Oklahoma State University, Stillwater, Oklahoma in May, 2013.

Completed the requirements for the Bachelor of Science in Computer Science at Jawaharlal Nehru Technology University, Hyderabad, India in 2011.

Experience:

Graduate Research Assistant August 2012 – April 2013
Oklahoma State University Stillwater, OK

- Maintain the department website using Plone and HTML: Creating verbal content for the website, capture and also edit the images for the webpage, updating each faculty's research page
- Maintain poster printer and print posters for the department and students
- Handle Department IT work such as: installing software's, setting up OS, maintaining printers, setting up and maintaining different hardware component, mapping department systems to servers