SECURE COMMUNICATION PROTOCOLS, SECRET SHARING AND AUTHENTICATION BASED ON GOLDBACH PARTITIONS

By

ADNAN AHMED MEMON

Bachelor of Engineering in Telecommunication

Sukkur Institute of Business Administration

Sukkur, Sindh, Pakistan

2012

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, 2017

SECURE COMMUNICATION PROTOCOLS, SECRET SHARING AND AUTHENTICATION BASED ON GOLDBACH PARTITIONS

Thesis Approved:

Dr. Subhash C. Kak

Thesis Adviser

Dr. Qi Cheng

Dr. Yanmin (Emily) Gong

ACKNOWLEDGEMENTS

Foremost, I would like to thank **Almighty Allah** for bestowing upon me His countless blessings, giving me strength and good health to finish this research.

It is a great pleasure to acknowledge my deepest thanks to **Prof. Subhash Kak**, *Regents Professor, School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK, USA*, whose consistent supervision and motivation enabled me to complete this research successfully.

I would like to thank my family, especially my Mom for encouraging and supporting me during this whole thesis period. Thank you friends for being with me in difficult times.

I would like to thank my sponsors i.e. **Fulbright, IIE and USEFP** for providing me fully funded scholarship so that I could focus on my research. This would not have been possible without their support.

I would like to dedicate this research to my (late) father **Mr. Ghulam Sarwar Memon** who has always been a role model for me.

Name: ADNAN AHMED MEMON

Date of Degree: JULY, 2017

Title of Study: SECURE COMMUNICATION PROTOCOLS, SECRET SHARING AND AUTHENTICATION BASED ON GOLDBACH PARTITIONS

Major Field: ELECTRICAL ENGINEERING

Abstract: This thesis investigates the use of Goldbach partitions for secure communication protocols and for finding large prime numbers that are fundamental to these protocols. It is proposed that multiple third parties be employed in TLS/SSL and secure communication protocols to distribute the trust and eliminate dependency on a single third party, which decreases the probability of forging a digital certificate and enhances the overall security of the system. Two methods are presented in which the secret key is not compromised until all third parties involved in the process are compromised. A new scheme to distribute secret shares using two third parties in the piggy bank cryptographic paradigm is proposed. Conditions under which Goldbach partitions are efficient in finding large prime numbers are presented. A method is also devised to sieve prime numbers which uses less number of operations as compared to the Sieve of Eratosthenes.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

Prime Numbers

Prime numbers play critical role in cryptography and they are widely used in various encryption algorithms like RSA [1] and Diffie-Hellman (D-H) key exchange [2]. The reason for using prime numbers is in the difficulty of factorization of a product of two large prime numbers and the solution of the discrete logarithm problem with respect to a large prime that sets up the asymmetry of computation that is fundamental to cryptography.

Since factorizing a given number would be computationally expensive and time consuming, primality tests such as Miller-Rabin [3], Fermat [4], AKS [5] are used to determine whether a random number generated is prime or not. Primality tests such as Miller-Rabin (M-R) and Fermat, determine the primality of a number in probable terms [6]. They are also called compositeness tests. AKS is deterministic primality test and can determine with certainty the primality of a number but requires considerably more computation resources. Miller-Rabin is the mostly used probabilistic primality test in practice [4].

Apart from random prime number generation algorithms and techniques, there are prime number sieves used to find all primes up to any number. Some of the most common sieves are the Sieve of Eratosthenes [7], the Sieve of Atkin [8], the Sieve of Sundaram [9], and several wheel sieves. In this thesis, we have also devised a way to sieve prime numbers that uses less number of operations as compared to the Sieve of Eratosthenes.

Goldbach Partitions

Goldbach partitions, which are two primes that add up to a given even number, have the potential to be used in cryptography [10]. These partitions can be used to find large prime numbers in an efficient manner. We investigated Goldbach partitions and found that they require less number of attempts to find prime numbers as compared to the incremental search method [6]. Goldbach partitions can also be used in secure communication protocols [11, 12].

Protocols for Security

Asymmetric encryption algorithms like RSA and D-H are widely used in public key cryptography for authentication, encryption, key exchange and digital signatures. However, symmetric encryption algorithms like AES [13], 3DES [14] are used for actual message encryption whereas public key algorithms are used for managing and distributing session keys. In RSA encryption, there is a pair of public key and the private key and the latter is kept secret. The sender encrypts the data using receiver's public key and the receiver decrypts the data using its own private key. D-H is used for key exchange over an insecure public channel. Elliptic Curve Cryptography (ECC) is gaining importance because its 160-bit key provides the equivalent security as 1024-bit key in RSA [15]. Because of its increasing popularity and small key advantage, the elliptic curve variants of algorithms like D-H, Digital Signature Algorithm (DSA) [16], etc. named as ECDH and ECDSA [17] are standardized. Hashing algorithms are one-way functions and are used to assure message integrity. SHA-2 [18] is used in TLS/SSL protocol for creating digital signatures.

Multiple Third Parties for Distributing Trust and Secret Sharing

Imagine a scenario in which two parties wish to communicate securely and rely on a third party for the key exchange. The third party knows all the secrets and if it is compromised or becomes malicious; this will completely break the security of the system. Relying on a single third party is risky and there is dire need of distributing the trust [19]. There can be two solutions to this problem. One is not to share the secret completely with the third party. Second is to distribute the secret among multiple third parties. Keeping in view such scenario, in this thesis, we propose a secure communication protocol which uses a single third party for the key exchange using Goldbach partitions. We propose another secure communication protocol using multiple third parties and Goldbach partitions in which the secret key is not compromised until all or minimum number of third parties are compromised.

In chapter VII, we present a new scheme to distribute secret shares with two third parties using the piggy bank cryptographic paradigm. We also present a protocol to give law enforcing agencies access to sensitive information present on a cell phone or a device using secret sharing scheme. These ideas for classical systems may also be applied to quantum schemes.

Certification Authority (CA)

The purpose of certificate authority (CA) is to issue digital certificates to authenticate and validate the identity of organizations as well as individuals on the Internet. Due to their central figure in TLS/SSL protocol, they are vulnerable to attacks and if digital certificates are stolen or forged, they compromise the trust and security of the system. In this thesis, we propose the use of multiple certificate authorities to issue digital certificates and validate the identity of organizations and individuals on the Internet. The use of multiple certificate authorities raises the difficulty in stealing or forging digital certificates, thereby enhancing the trust and security of the system.

CHAPTER II

LITERATURE REVIEW

Goldbach Partitions

In 1742, Christian Goldbach gave the 'strong' version of the conjecture now named after him which states that every even integer can be expressed as a sum of two primes. There is also the 'weak' Goldbach conjecture, which states that every odd number greater than 7 is a sum of three distinct odd primes (Table 2.1). These conjectures have been verified for values $\leq 4\times 10^{18}$ through computer experiments [10].

Table 2.1. Example of Goldbach partitions

Even Numbers	**Odd Numbers**
16 = 11 + 5	15 = 7 + 5 + 3
240 = 229 + 11	503 = 179 + 281 + 43
3186 = 2803 + 383	31019 = 503 + 6907 + 23609
33330 = 11027 + 22303	493011 = 430543 + 62011 + 457

Here, we focus only on partitions of even numbers. Most even numbers can be represented as a sum of two or more distinct pairs of primes and every distinct pair is called a Goldbach partition.

For *n*=30: there are three Goldbach partitions: (23, 7), (19, 11) and (17, 13).

We can calculate the percentage of Goldbach partitions in the range [n/2, n-2] by writing down all the primes in given range, subtract every prime from *n* and check if the difference is also a prime [20]. If it is a prime, it is called a Goldbach partition. We calculate the percentage of Goldbach partitions by dividing the number of Goldbach partitions from total number of partitions. Figure 2.1 shows the percentage of Goldbach partitions in the range [n/2, n-2] until 10^5 using MATLAB. We illustrate finding Goldbach partitions and its percentage through an example, where *n*=28.

Range = [n/2, n-2]	= [14, 26]
Primes in the range	= 17, 19, and 23
(17, 11), (23, 5)	= Goldbach partition(s)
(19, 9)	≠ Goldbach partition(s)
% of Goldbach partitions	= 2/3 = **66.6%**

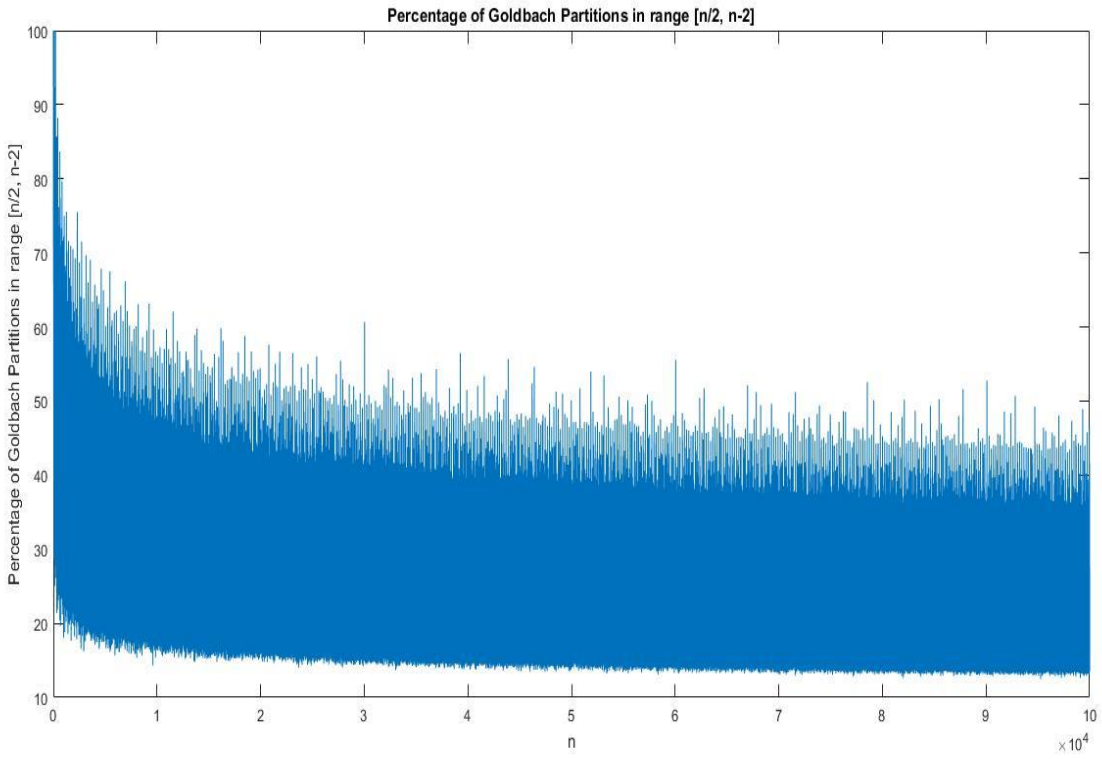

Figure 2.1. Percentage of Goldbach partitions for even numbers in range [n/2, n-2] until 10^5.

Certification Authority (CA)

The purpose of certificate authority (CA) is to issue digital certificates to authenticate and validate the identity of organizations as well as individuals on the Internet. There have been many incidents when CAs have been compromised and some of them were top market shareholders at that time. In 2001, Verisign issued certificates to someone impersonating Microsoft. In 2008, Thawte mistakenly issued non-Microsoft employee a certificate for Live.com and Comodo issued mozilla.org certificate to Startcom. In 2011, Comodo was compromised and nine fraudulent certificates were issued in the names for Google, Yahoo, Live, etc. DigiNotar was also compromised and 531 counterfeit certificates were issued which created major issues for Dutch government. In 2013, code signing certificate was issued by DigiCert to a bogus company [21].

Keeping in view such incidents, some enhancements in the existing framework and some alternatives were suggested and are discussed in detail in the following paragraphs.

Certificate Authority Authorization (CAA) DNS Resource Record as mentioned in IETF draft RFC 6844 allows requesting party to specify which CAs can issue certificates for their domain [22]. Its implementation will avoid unintended mis-issuance of certificates and help in detecting possible mis-issued certificates. It will provide increased security to existing CA infrastructure by limiting the number of CAs allowed to issue certificates for a specific domain. The CAA check at the time of issuance of certificates is made mandatory by CA/Browser forum starting September 2017 [23]. But, attackers can still manipulate DNS records since the use of Domain Name System Security Extensions (DNSSEC) is recommended but not mandatory.

Public Key Pinning Extension for HTTPS can help prevent man-in-the-middle (MIM) attack and impersonation if one or more CAs are compromised [24]. The web server communicates the details of the certificate and the public key to be used during first communication. Clients can make assertions about which certificates will be used by specific web server and will not establish secure

connection if the certificate issued is not the one expected but it does not provide protection when used for the first time.

The idea of Certificate Transparency is to make the SSL certificates a public record so that they can be audited and suspicious certificates may be detected but it still requires someone to keep an eye on public records to get benefit from this idea [25].

The Sieve of Eratosthenes

The Sieve of Eratosthenes is an ancient but efficient prime number sieve to find all the prime numbers up to any number *n.* The process is as follows:

1. Make a list of consecutive positive integers starting from 2 till *n*.
2. Set p as first unmarked number in the list and mark it as prime.
3. Mark all the multiples of p.
4. Set p as the next unmarked number to the last p in the list and mark it as prime and repeat steps 3 and 4 until $p > \sqrt{n}$.
5. The unmarked numbers are all prime numbers below *n.*

Example, *n*=14

1. $L = \{2,3,4,5,6,7,8,9,10,11,12,13,14\}$
2. p =2;
3. Multiples of 2 = {4,6,8,10,12,14} L = {2,3,5,7,9,11,13}

   p =3;

   Multiples of 3 = {6,9,12} L = {2,3,5,7,11,13}

   Since, $5 > \sqrt{14}$.

   Therefore L = {2,3,5,7,11,13} = Prime Numbers till 14.

CHAPTER III

PROPERTIES OF GOLDBACH PARTITIONS

Average Number of Attempts to Find Primes using Goldbach Partitions

Prime number theorem describes the distribution of prime numbers. It also gives an approximation of the average gap among primes below x and can be calculated using the formula $\frac{x}{\pi(x)}$, where $\pi(x)$ is the number of prime numbers below x.

Suppose, we randomly pick a large odd number and apply primality tests like Miller-Rabin and AKS to determine its primality. If it is not prime, we add two to that number and repeat the same process until we find the prime number. This process is known as incremental search method to find prime numbers [6]. Since, we will only be using odd numbers for primality tests; we can safely assume that the average number of attempts to find a prime number will be approximate half of the average prime gap calculated using prime number theorem.

We propose a method to calculate average number of attempts to find a specific number of primes using Goldbach partitions and then compare it with incremental search method. The primality can be determined using standard primality tests. The method is given as under:

1. Calculate the value of x = (No. of primes required × 2) + 2.
2. Generate a sequence of primes and replace the first prime, which is two with one. The number of primes in the sequence may be equal to the average prime gap calculated using prime number theorem.
3. Start from 4 and denote it as *n*.
4. Subtract 1st number of sequence generated in step 2 from *n* and determine its primality. If it is not a prime, subtract 2nd number of sequence from *n* and repeat the same process until a prime number is found.
5. Count the number of attempts for finding the prime number in step 4.
6. Add two to *n*.
7. Repeat step 4 through 6 until the value of *n* is equal to *x*.
8. Calculate average number of attempts using this formula.

$$\frac{\Sigma\ (No.of\ attempts * No.of\ values)}{No.of\ prime\ numbers}$$

Example:

We wish to find average number of attempts for finding first four primes using Goldbach partitions for which we have to repeat the process until x=10. The process is given as under and Table 3.1 summarizes the results.

Step 1. $x = 4 \times 2 + 2 = 10$

Step 2. Sequence generated = {1, 3, 5, 7}

Step 4-6. *n* =4; 4-1 = 3 (Prime) Number of attempt(s) =1

Step 4-6. *n* =6; 6-1=5 (Prime); Number of attempt(s) =1

Step 4-6. *n* =8; 8-1=7 (Prime); Number of attempt(s) =1

Step 4-6. n =10; 10-1=9 (Not prime), 10-3=7 (Prime) Number of attempt(s) =2

Step 7. Average = Σ (No. of attempts × No. of values) / No. of prime numbers

Average= {(1×3) + (2×1)} / 4 = 5/4

Average number of attempts= **1.25**

Table 3.1. Average number of attempts using Goldbach partitions and incremental search method

Number of primes	**Average number of attempts using Goldbach partitions**	**Average number of attempts using incremental search**	**Difference in number of attempts**
25	1.560	2.000	0.441
168	2.262	2.976	0.714
1229	3.069	4.068	0.999
9592	4.078	5.212	1.134
78498	5.163	6.370	1.207

If we use Goldbach partitions to find specific number of primes, the average number of attempts are found to be less as compared to incremental search method.

As every prime number is of the form $6n \pm 1$, we can use this property along with Goldbach partitions to reduce the number of attempts in finding prime numbers. Then the proposed modified procedure to calculate average number of attempts to find prime numbers using Goldbach partitions in a given range is given as under:

1. Select a range and denote it as *range*. Input a range in such a way that the lower limit is of the form $6k + 4$ where $k = 0, 1, 2, 3, ...$

2. Generate a sequence of primes replace the first prime, which is two with one. The number of primes in the sequences may be equal to the average prime gap calculated using prime number theorem.
3. Start one loop from (lower limit + 6) and denote it as *n1*.
4. Subtract 2nd number of sequence generated in step 2 from *n1* and determine its primality. If it is not a prime, subtract 3rd number of sequence from *n1* and repeat the same process until a prime number is found.
5. Count the number of attempts for finding the prime number in step 4.
6. Add six to *n1*.
7. Repeat step 4 through 6 until the value of *n1* is equal or less than the upper limit.
8. Start second loop from (lower limit + 2) and denote it as *n2*.
9. Subtract 1st number of sequence generated in step 2 from *n2* and determine its primality. If it is not a prime, subtract 2nd number of sequence from *n2* and repeat the same process until a prime number is found.
10. Count the number of attempts for finding the prime number in step 9.
11. Add six to *n2*.
12. Repeat step 9 through 11 until the value of *n2* is equal or less than the upper limit.
13. Calculate average number of attempts using this formula.

$$\frac{\Sigma\,(No.of\ attempts\ \times\ No.of\ values)}{\frac{range}{2}}$$

Example:

We wish to find average number of attempts in range from 10 to 30 using Goldbach partitions and $6n \pm 1$ property of primes. The process is given as under and Table 3.2 summarizes the results.

Step 1. *Range=10* to *30*

Step 2. Sequence generated = {1, 3, 5, 7, 11}

Step 3-5. *n1* =16; 16-3=13 (Prime); Number of attempt(s) =1

Step 4-7. *n1* =22; 22-3=19 (Prime); Number of attempt(s) =1

Step 4-7. *n1*=28; 28-3=25 (Not prime), 28-5=23(Prime); Number of attempt(s) =2

Step 8-10. *n2* =12; 12-1 = 11 (Prime) Number of attempt(s) =1

Step 9-12. *n2* =18; 18-1=17 (Prime); Number of attempt(s) =1

Step 9-12. *n2*=24; 24-1=23 (prime); Number of attempt(s) =1

Step 9-12. *n2* =30; 30-1=29 (Prime) Number of attempt(s) =1

Step 13. Average = $\frac{\Sigma(No.\ of\ attempts \times No.\ of\ values)}{\frac{range}{2}}$

Average = {(1×6) + (2×1)} / 10 = 8/10= **0.80**

Table 3.2. Average number of attempts using Goldbach partitions and Incremental search method

Range	**Average no. of attempts using Goldbach partitions**	**Average no. of attempts using Incremental search**	**Difference in number of attempts**
10-100	0.978	2.143	1.165
100-1000	1.578	3.147	1.569
1000-10000	2.328	4.241	1.913
10000-100000	2.996	5.381	2.385
100000-1000000	3.734	6.531	2.797

Here, we also observe that while using Goldbach partitions to find prime numbers in a given range, the average number of attempts are less as compared to Incremental search method. We can further improve the average number of attempts by putting a restriction on maximum number of attempts to find a large prime number in a given range.

Reasons to Choose the Range [n/2 to n-2] Instead of [3 to n/2]

In previous studies [10, 20], the range considered for candidate primes in Goldbach partitions was [*n/2, n-2*], where *n* is the number for which we make partitions. The process of making Goldbach partitions and calculating the percentage of Goldbach partitions is illustrated here through an example where *n=40.* The range is [n/2, n-2] = [20, 38]. The candidate primes in the given range are 23, 29, 31 and 37. The partitions are (23, 17), (29, 11), (31, 9), (37, 3). Out of four, three are Goldbach partitions. The partitions (31, 9) is not Goldbach partition because 9 is not a prime

This raises a question why the range [n/2, n-2] is considered and why not the range [3, n/2]. We extend the previous example and find Goldbach partitions for the Range [3, n/2] = [3, 20]. The partitions are (3, 37), (5, 35), (7, 33), (11, 29), (13, 27), (17, 23), (19, 21). Out of seven, only three (23, 17), (29, 11), (37, 3) are Goldbach partitions. The partitions (5, 35), (7, 33), (13, 27), (19, 21) are not Goldbach partitions.

Interesting to note that Goldbach partitions are the same in both ranges: (23, 17), (29, 11), (37, 3). We performed simulations and compared the results for *n* obtained by the product of prime numbers which are summarized in Table 3.3.

Table 3.3. Comparison of percentages of Goldbach partitions in range [n/2, n-2] and [3, n/2]

Prime numbers	***n***	**No. of candidate primes in range [n/2, n-2]**	**No. of candidate primes in range [3, n/2]**	**% of Goldbach partitions in range [n/2, n-2]**	**% of Goldbach partitions in range [3, n/2]**	**Difference in %**
2,3,5	30	3	5	100.00	60.00	40.00
2,3,7	42	4	7	100.00	57.14	42.86
2,3,11	66	7	10	85.71	60.00	25.71
2,3,13	78	9	11	77.78	63.64	14.14
2,3,17	102	10	14	80.00	57.14	22.86
2,3,19	108	11	15	72.73	53.33	19.40
2,3,5,7	210	19	26	100.00	73.08	26.92
2,3,5,11	330	28	37	85.71	64.86	20.85
2,3,5,13	390	32	43	84.38	62.80	21.58
2,3,5,17	510	42	53	76.20	60.38	15.82
2,3,7,13	546	42	57	71.43	52.63	18.80
2,3,7,17	714	56	70	66.07	52.86	13.21
2,3,7,19	798	60	77	63.33	49.35	13.98
2,3,11,19	1254	90	113	56.67	45.13	11.54
2,3,19,23	2622	166	213	46.39	36.15	10.24
2,3,11,17,23	25806	1306	1534	38.70	32.90	5.70
2,3,5,7,11,13,17	510510	19865	22465	47.79	42.26	5.53

It can be seen that there are more number of candidate primes in the range [3, n/2] but its percentage is less than that in the range [n/2, n-2]. The Goldbach partitions are always the same in both ranges but the number of candidate primes will be greater in the range [3, n/2] as deduced from prime number theorem. This holds true for all cases. For $n >= 24$, we can predict that there will always be more number of partitions but less percentage of Goldbach partitions in the range [3, n/2] than in the range [n/2, n-2].

We did the simulations up to n=50000 and the results are given in Figure 3.1.

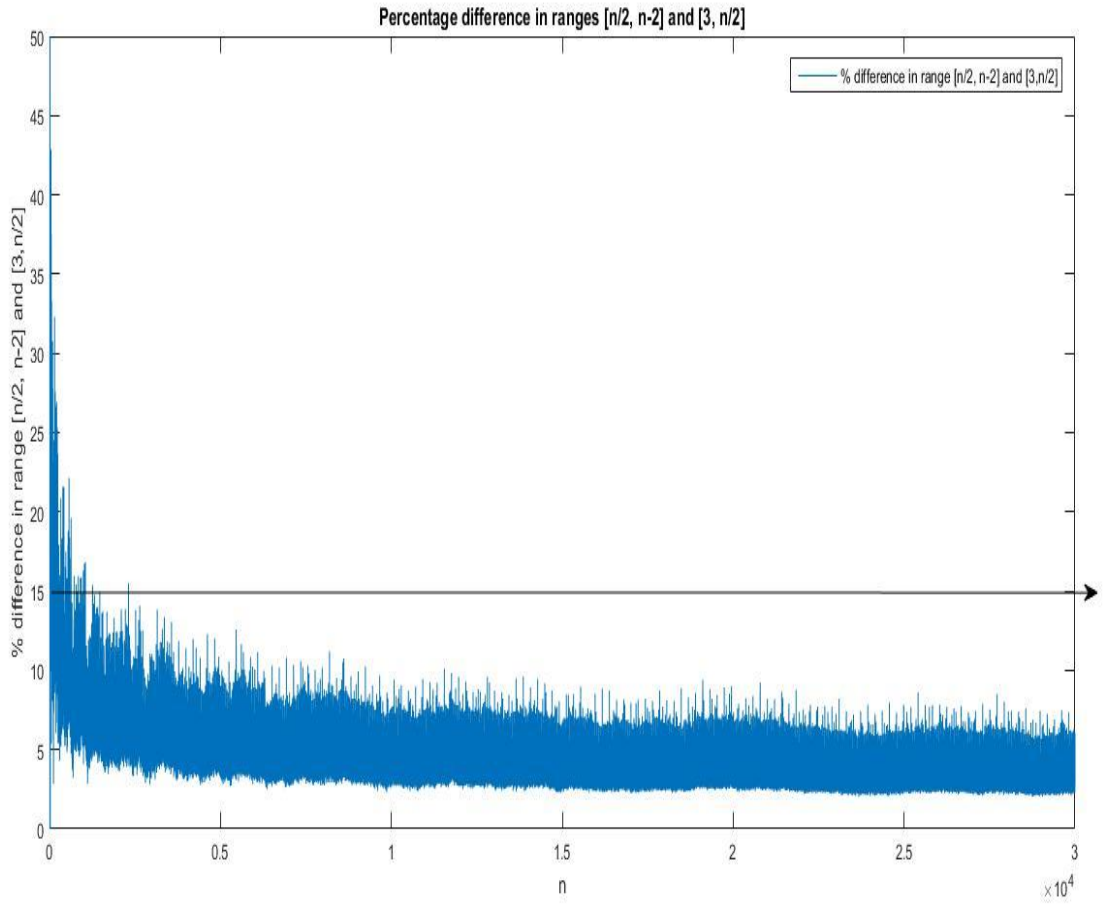

Figure 3.1. Percentage difference in ranges [n/2, n-2] and [3, n/2]

It is very interesting to note that the difference in percentages of Goldbach partitions between two ranges is getting smaller and smaller as the value of *n* increases. The difference is high when the value of *n* is product of prime numbers as compared to other numbers as shown in Figure 3.2.

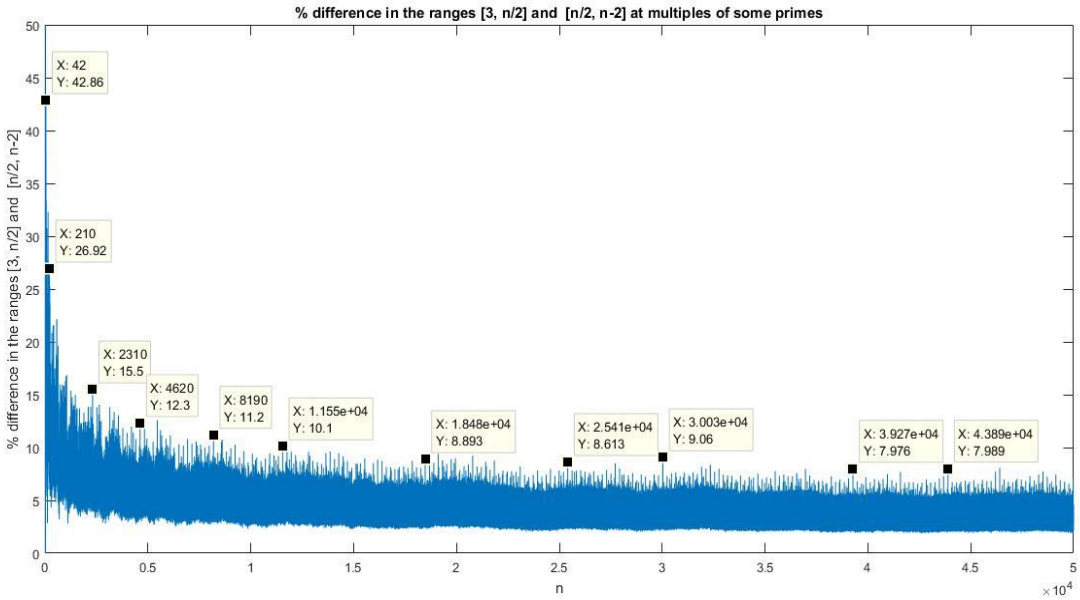

Figure 3.2. Percentage difference in ranges [n/2, n-2] and [3, n/2] at product of primes

Local Peaks at Integer Multiples of Sequential Product of Primes

The percentage of Goldbach partitions is more in both ranges when *n* is product of primes as mentioned in previous section. Sequential product of primes means the multiplication of primes in sequence such as 2×3 = 6; 2×3×5 = 30; 2×3×5×7 = 210; 2×3×5×7×11 = 2310; 2×3×5×7×11×13 = 30030; 2×3×5×7×11×13×17 = 510510, etc.

Another interesting result is that the local peaks are at sequential product of primes and their integer multiples. Figure 3.3. shows peaks at product of primes. i.e. 30030= 2×3×5×7×11×13 and at its integer multiples 60060 and 90090.

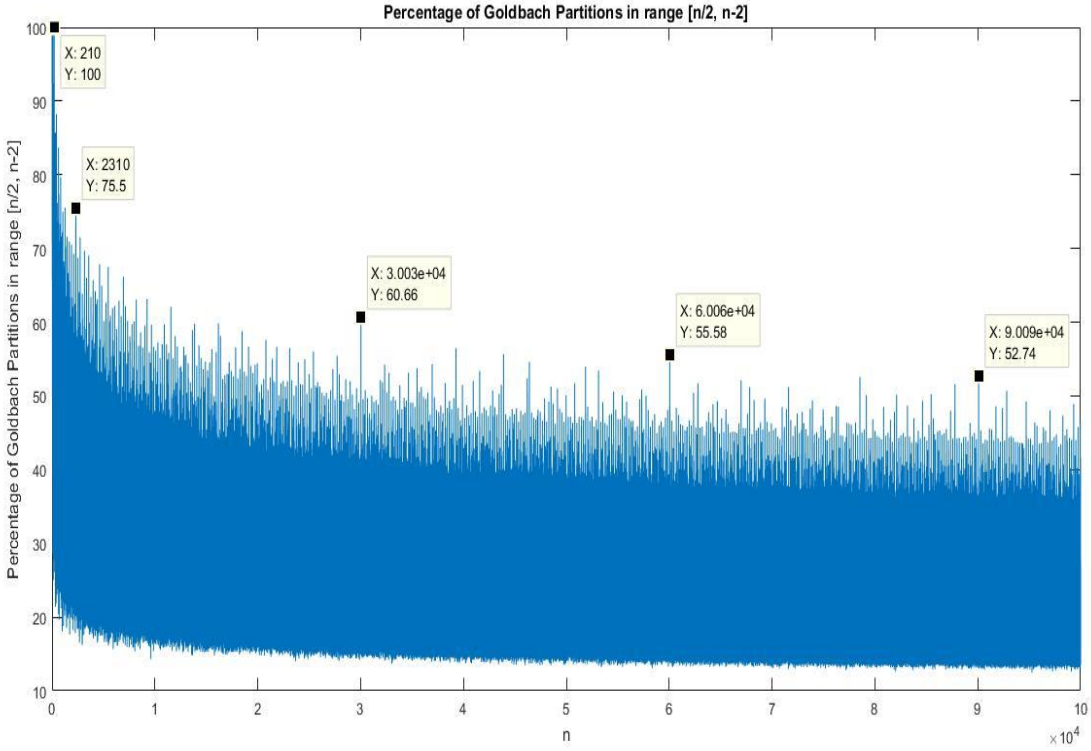

Figure 3.3. Percentage of Goldbach partitions in range [n/2, n-2]

Let us examine it more closely. The figures 3.4 – 3.8. show the local peaks. It was very interesting to find that if the range considered is *n* then first we find the local range. Local range is found in such a way that *n* lies in between two sequential product of primes. Then the local peaks will be at the integer multiples of lower sequential product of primes.

If $n = 21000$, local range will be $2310 < n < 30030$, then the local peaks will be at integer multiples of 2310 i.e. 4620, 6930, 9240, 11550, 13360, 16170, 18480, 20790. In other words, between 2310 and 4620 there will not be a number where percentage of Goldbach partitions will be greater than the percentage at 2310. The same will be true for all ranges such as [4620, 6930], [6930, 9240], etc. It is important to note that if $n = 2310$, the local peaks will be at integer multiples of 210 i.e. 420, 630, 840, 1050, 1260, 1470, 1680, 1890, and 2100.

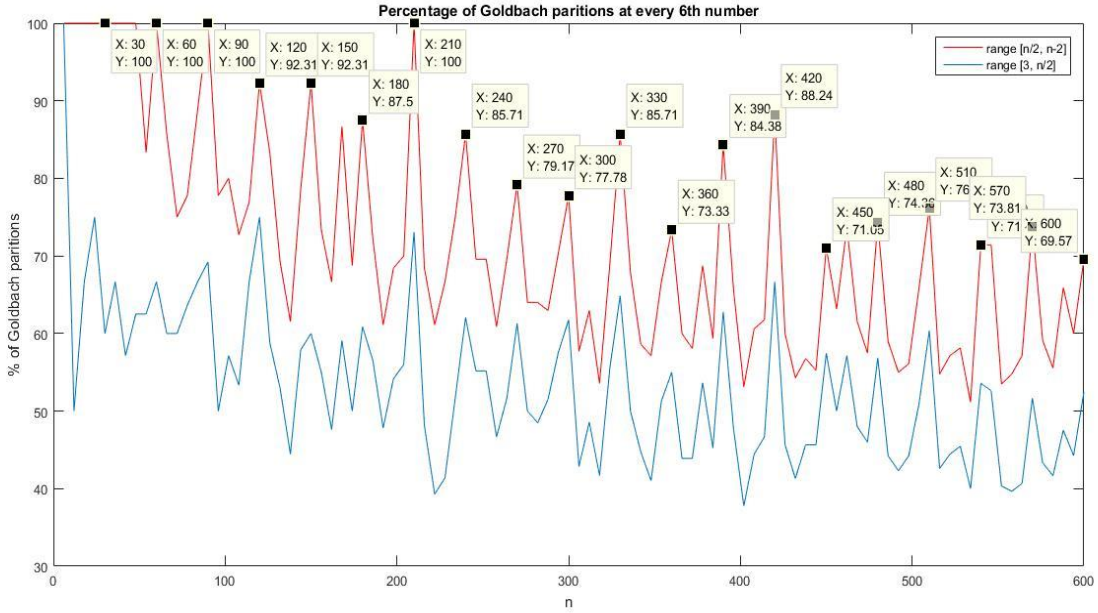

Figure 3.4. Percentage of Goldbach partitions at every 6th number

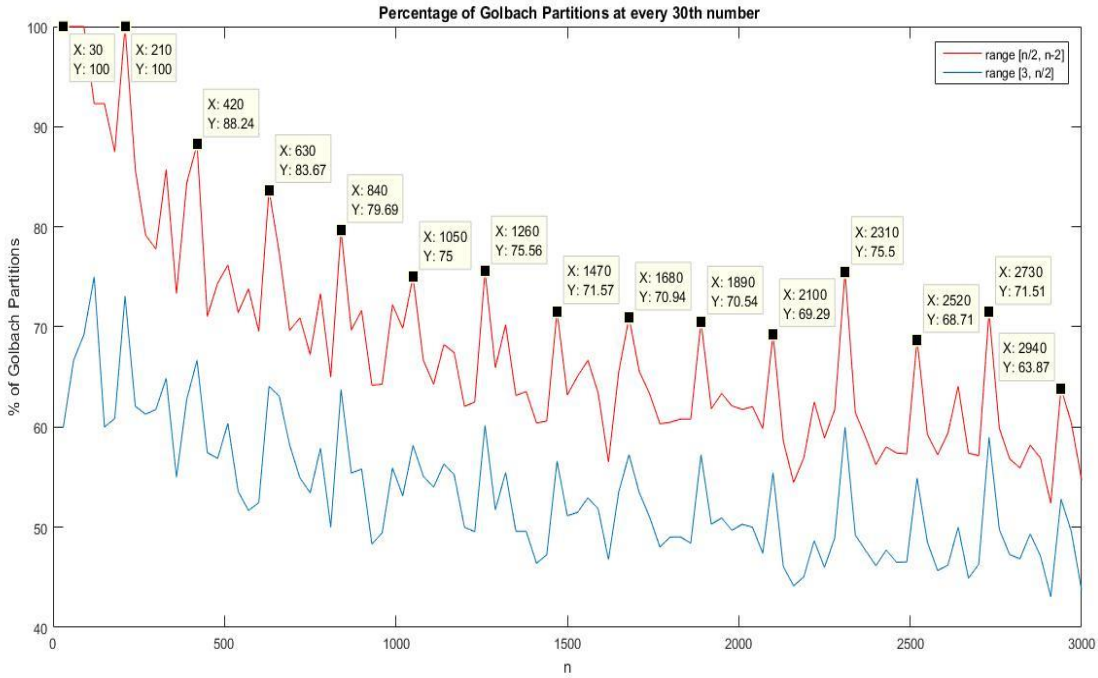

Figure 3.5. Percentage of Goldbach partitions at every 30th number

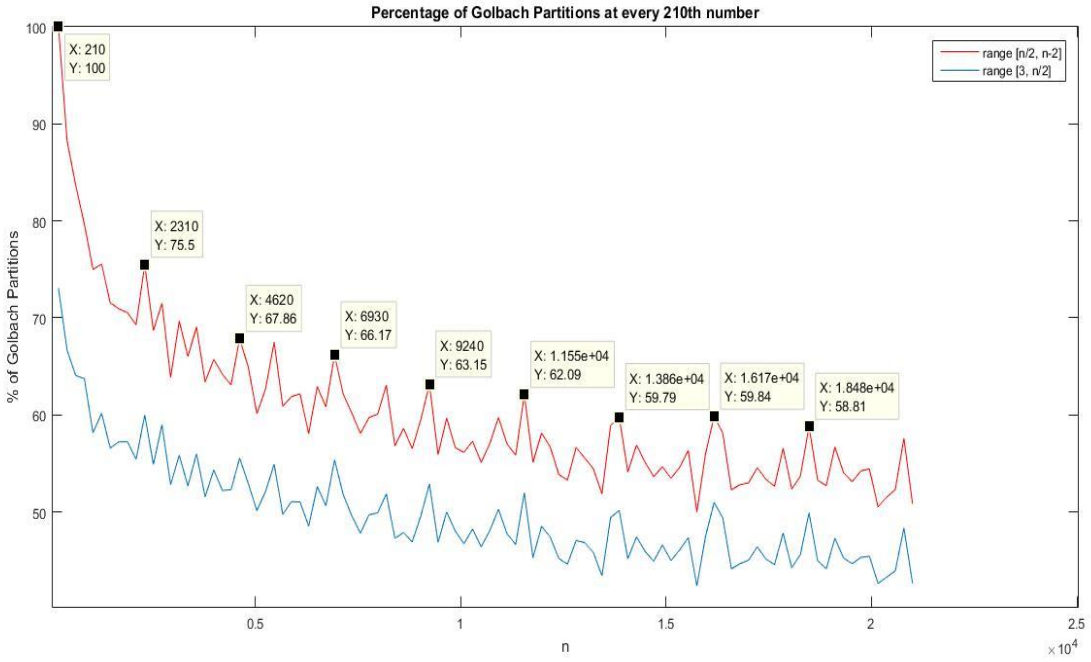

Figure 3.6. Percentage of Goldbach partitions at every 210th number

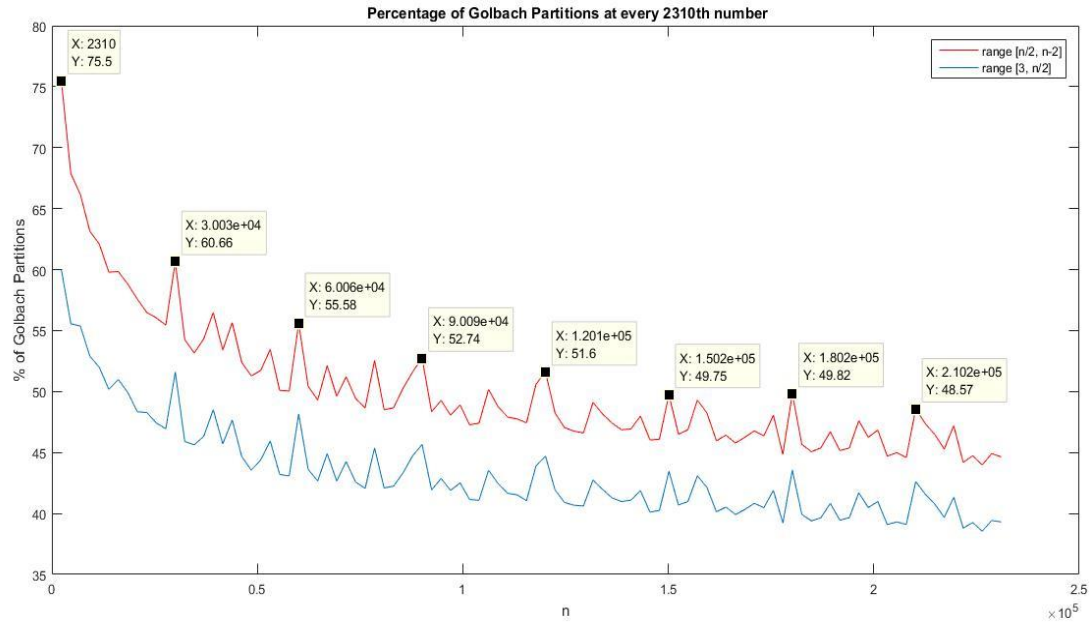

Figure 3.7. Percentage of Goldbach partitions at every 2310th number

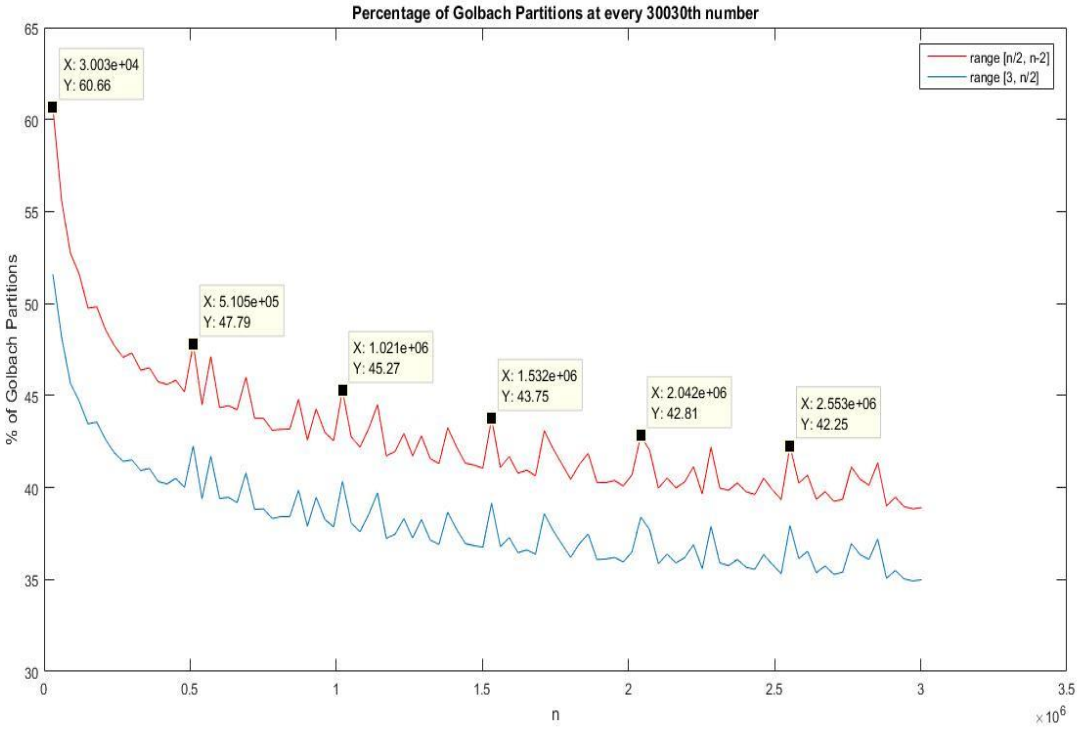

Figure 3.8. Percentage of Goldbach partitions at every 30030th number

It is also interesting to note that the percentage of Goldbach partitions in both ranges [n/2, n-2] and [3, n/2] varies in somehow almost same fashion and is the main reason that the minimum and maximum value of difference in both ranges remain very small as the value of *n* increases.

CHAPTER IV

THE PROBLEM OF COMPUTING PRIMES

The sieve of Eratosthenes is an ancient but efficient prime number sieve to find all the prime numbers up to any number *n* and is described in Chapter II in detail. Since every prime is odd except two; we consider a list of positive odd integers up to *n* instead of all positive integers. We declare the first number in the sequence as prime p, and subtract all the odd multiples of p starting from p^2 from the original sequence S to create a new sequence S_n. Then we set the next number to the last p in the new sequence as new p and continue this process until the new p in the new sequence is greater than $\sqrt{n}$ [26].

We illustrate the process through an example where *n=53.*

S = {3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,51,53}

The first number in the sequence is 3, which will be considered as a prime number p. We create a sequence using all the odd multiples of 3 starting from $p^2 = 9$ and subtract it from the original sequence S.

$S_3 =$ {9,15,21,27,33,39,45,51}

$S_n = S - S_3 =$ {3,5,7,11,13,17,19,23,25,29,31,35,37,41,43,47,49,53}

The next number to the last p in the new sequence is 5, which will be considered as a prime number p now. We create a sequence using all the odd multiples of 5 starting from $p^2 = 25$ and subtract it from the new sequence S_n.

$S_5 = \{25,35,45\}$

$S_n = S - S_5 = \{3,5,7,11,13,17,19,23,29,31,37,41,43,47,49,53\}$

The next number to the last p in the new sequence is 7, which will be considered as a prime number p now. We create a sequence using all the odd multiples of 7 starting from $p^2 = 49$ and subtract it from the new sequence S_n.

$S_7 = \{49\}$

$S_n = S - S_7 = \{3,5,7,11,13,17,19,23,29,31,37,41,43,47,53\}$

Since $\sqrt{53} = 7.28$, and the next number to the last p in the new sequence is 11, which is greater than $\sqrt{n}$, we will stop the process here. The new sequence is now the list of all prime number up to number *n. i.e.* $S_n = \{3,5,7,11,13,17,19,23,29,31,37,41,43,47,53\}$

The problem with the sieve of Eratosthenes is that it crosses off the same number multiple number of times. In the above example, the number 45 was crossed off two times as it is the odd multiple of 3 and 5 though we have already considered few optimizations to the actual sieve i.e. starting from p^2 instead of p, considering only odd positive integers and odd multiples. The number of crossings increase as *n* increases. To reduce the number of crossings, we have devised a new way of crossing off the numbers and is explained step-by-step here.

1. Generate a sequence of odd positive integers until *n*.
2. Subtract all the odd multiples of 3 and 5 starting from 9 and 25 respectively from the original sequence to generate a new sequence S_n.

3. Multiply each number in S_n starting from 7 with itself and the numbers before it and we call this number as x. The number x is multiplied in the increasing order starting from the first number in the sequence S_n. Subtract all the products from the sequence S_n.
4. Stop multiplying and move to the next number in the sequence S_n if the product of two numbers is greater than *n.*
5. End this process when $\frac{n}{x} < 7$.

We illustrate this through an example where *n* = 169.

1. Generate a sequence of odd positive integers until *n*.

Table 4.1. Sequence of odd positive integers until *n*

	3	5	7	9	11	13	15	17	19
21	23	25	27	29	31	33	35	37	39
41	43	45	47	49	51	53	55	57	59
61	63	65	67	69	71	73	75	77	79
81	83	85	87	89	91	93	95	97	99
101	103	105	107	109	111	113	115	117	119
121	123	125	127	129	131	133	135	137	139
141	143	145	147	149	151	153	155	157	159
161	163	165	167	169					

2. Subtract all the odd multiples of 3 and 5 starting from 9 and 25 respectively from the original sequence to generate a new sequence S_n.

Table 4.2. Multiples of 3 and 5 starting from 9 and 25 respectively.

9	15	21	27	33	39	45	51	57	63
69	75	81	87	93	99	105	111	117	123
129	135	141	147	153	159	165	25	35	45
55	65	75	85	95	105	115	125	135	145
155	165								

Table 4.3. New sequence S_n by subtracting all the odd multiples of 3 and 5.

	3	5	7	11	13	17	19	23	29
31	37	41	43	47	49	53	59	61	67
71	73	77	79	83	89	91	97	101	103
107	109	113	119	121	127	131	133	137	139
143	149	151	157	161	163	167	169		

3. Multiply each number in S_n starting from 7 with itself and the numbers before it and we call this number as x. The number x is multiplied in the increasing order starting from the first number in the sequence S_n. Subtract all the products from the sequence S_n.

$7 \times 7 = 49,$

$11 \times 7 = 77,\ 11 \times 11 = 121,$

$13 \times 7 = 91,\ 13 \times 11 = 143,\ 13 \times 13 = 169,$

$17 \times 7 = 119,\quad 17 \times 11 = 187,$ (Stop because product is greater than *n*)

$19 \times 7 = 133,\quad 19 \times 11 = 209,$ (Stop because product is greater than *n*)

$23 \times 7 = 161,\quad 23 \times 11 = 253,$ (Stop because product is greater than *n*)

4. End the process because 169/29 < 7

Table 4.4. List of all prime numbers until 169

	3	5	7	11	13	17	19	23	29
31	37	41	43	47	53	59	61	67	71
73	79	83	89	97	101	103	107	109	113
127	131	137	139	149	151	157	163	167	

Figure 4.1. shows the comparison between the number of operations of the Sieve of Eratosthenes and the new sieve against the value of *n* which changes from 1 to 10^8. The new sieve uses less number of operations as the value of *n* increases.

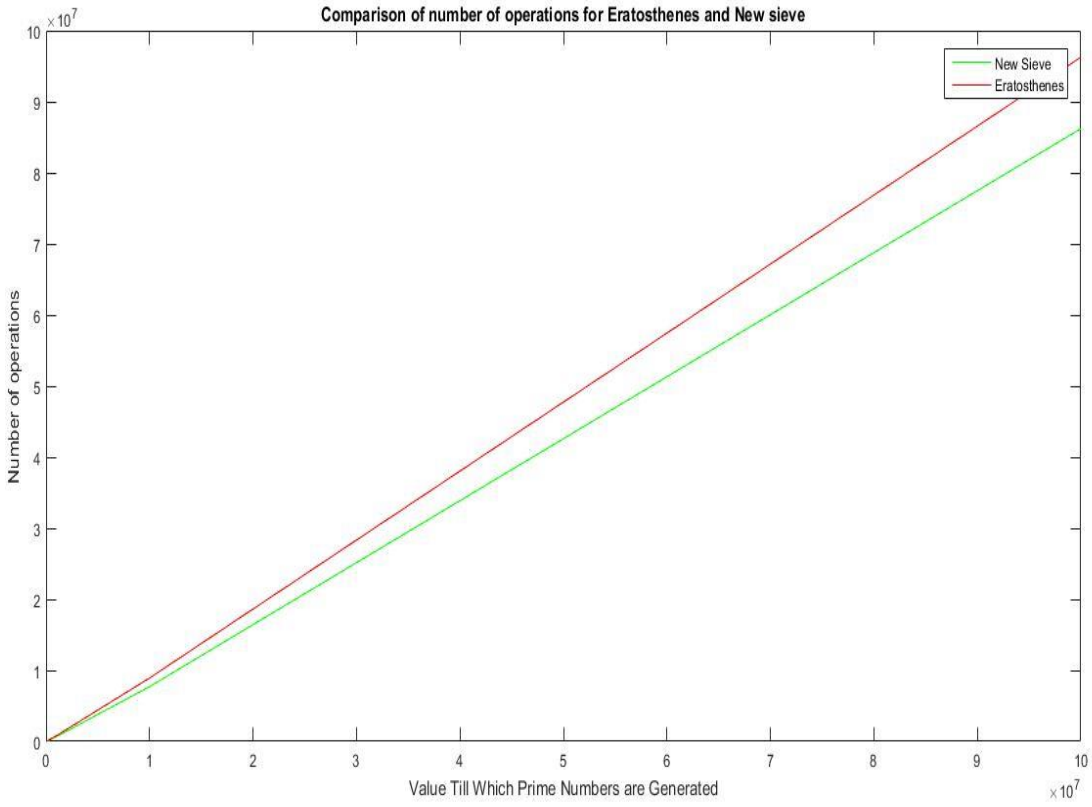

Figure 4.1. Comparison of number of operations for Eratosthenes and its improved version

CHAPTER V

USE OF MULTIPLE CERTIFICATE AUTHORITIES IN TLS/SSL PROTOCOL

Certification Authority

The purpose of certificate authority (CA) is to issue digital certificates to authenticate and validate the identity of organizations as well as individuals on the Internet. It is also the responsibility of CAs to maintain updated information and validity of issued certificates and they do it using Online Certificate Status Protocol (OSCP) and/or Certificate Revocation Lists (CRLs). There is a strict hierarchical chain of trust in a tree-like structure and the trust flows downward only. Root CAs are the topmost secure CAs and serve as trust anchors for digital certificates. Intermediate CAs are below the root CAs and issue most of the certificates to clients or end users.

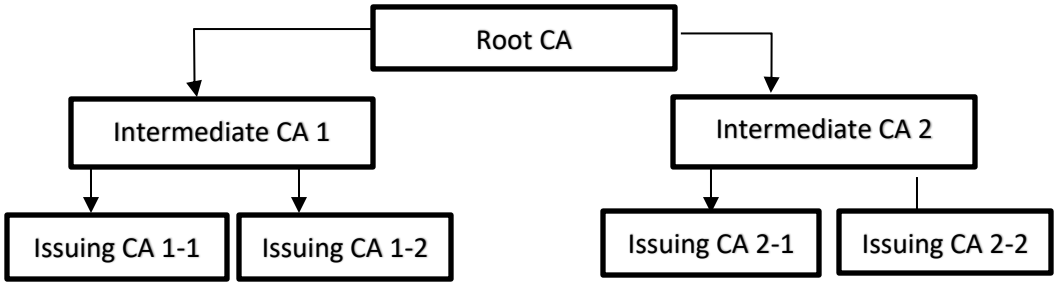

Figure 5.1. Certification Authority Hierarchy

In cryptography, X.509 standard specifies the format of digital certificates. Digital certificates are also known as public key certificates or SSL certificates. Individuals or organizations apply for a digital certificate using CSR (Certificate Signing Request), and then a CA verifies the applicant's credentials and issues digital certificate as shown in Figure 5.2. The applicant usually generates the key pair of public and private key and the private key is always kept secret. Digital certificates contain the version number of X.509 standard, serial number, issuer name, digital signature of CA, validity period, identity of public key owner, public key, and relevant information about the digital certificate owner.

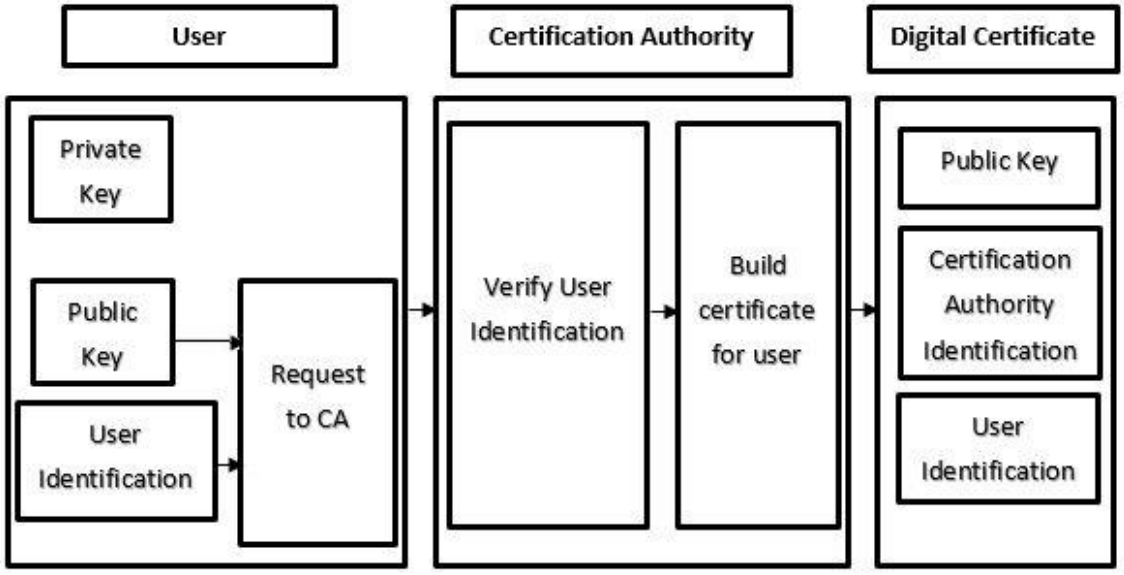

Figure 5.2. Process of issuing Digital certificate using single CA

Digital certificates play critical role in providing secure communication between servers and clients. These certificates are used in protocols like SSL/TLS, VPN, IPSec etc. These certificates ensure the authenticity of a web server and a client feels secure interacting and sharing its information on such websites. When a user visits a website secured by SSL/digital certificate, the

user uses the public key contained in the digital certificate for encrypting data. The user also receives digital signature of the public key signed by CA's private key and can be verified using CA's public key and if the user trusts the CA, it will trust the public key contained in the certificate. A fake website does not have the private key of CA and hence cannot create a fake digital signature to impersonate the actual website. The browsers and operating systems maintain lists of trusted CAs and their public keys.

If the hacker manages to fake a certificate for a particular website and could redirect the traffic to his rogue website, then the hacker can read all the information and the browser would not ring any alarm because it trusts the CA. The hackers may spoof someone by using the forged certificates and they may claim to be one, which they are not. The hackers may also setup a fake root CA using OpenSSL and might succeed to add this fake certificate in the collection of root certificates present in the computers. Then, they may setup a fake but lucrative e-business website to trap the customers and they will be able to steal their credit card information [27].

CAs have been compromised in the past as mentioned in Chapter II in detail. When a CA is compromised, one may think of removing it from the trusted CAs. It will create problems when a large CA, who has issued millions of certificates to different organizations, is removed from trusted ones because all the websites who were issued certificates by that CA will be seen as insecure even if their certificates are not compromised or forged. It implies that we cannot remove those large CAs from trusted ones and are bound to trust them forever.

Keeping in view such incidents, some enhancements in the existing framework and some alternatives were suggested and are discussed in detail in Chapter II.

We also suggest an enhancement to the existing CA ecosystem, which can be used in combination with one or more complements and enhancements proposed and are mentioned in Chapter II. We

want to utilize the idea of distributing the trust by using multiple CAs to verify the identity of an organization or individual on the Internet as shown in Figure 5.3. It will have the following benefits:

1. It will make it difficult and harder for the hacker to compromise multiple CAs at the same time to create a forged certificate and impersonate any website.
2. It will also make it easy to remove compromised CA(s) from trusted ones.
3. It will force every CA to be more vigilant and careful about their security infrastructure.

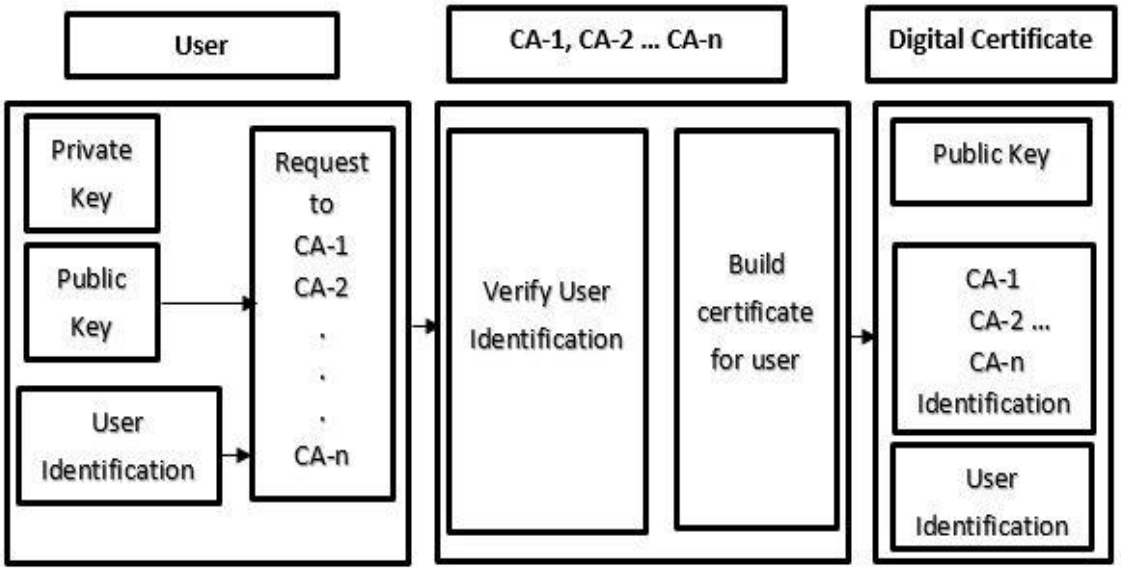

Figure 5.3. Process of issuing Digital certificate using multiple CAs

The current format of X.509 v3 certificates only allows only one issuer and single signature on a certificate. If we want to use multiple signatures by distinct CAs on the same certificate, some updates will be needed for existing X.509 format to pave the way for multiple signatures on the same certificate.

Another way to verify the certificate is to use the same public key while requesting for a certificate from different CAs. While establishing SSL session, the user requests multiple certificates instead of just one for verification to establish secure connection. Since the public key is same in every certificate, it can be used for encrypting data from the client to the server. If same public key is not used in every certificate, then the client may choose which public key it has chosen for encryption during TLS/SSL hand shake and the corresponding private key is used at the server side.

The optimal number of certificates used for verification may be standardized taking into account the factors of introduced redundancy, increased security, etc. The provision may be made that if *k* out of *n* provided certificates are verified, then a secure connection may be established. In case, $(k - n)$ CA(s) are compromised, the secure connection can be established with certain security rating. A new term may be introduced to indicate the security rating of a certain certificate. The certificate which can be verified by all CAs will have the highest rating and the one with minimum number of CAs will have the lowest rating.

Analysis

Since we are using more than one CA, it will make it hard for an attacker to forge or steal digital certificates of the same website provided by distinct CAs. It is easier to steal or forge more than one digital certificate of different websites when one or more CA(s) are compromised, but it is hard to steal or forge the digital certificates of the same website and compromise many CAs and that too the exact CAs at the same time.

We assume that an attacker can compromise any CA with the same probability. Let p be the probability that a digital certificate can be stolen or forged if a CA is compromised, then p^n is the probability of *n* CAs compromised since every CA and digital certificate is independent of each

other. The rate of successful attack is decreasing exponentially and the difficulty raises exponentially even in the worst-case scenario under given assumptions as shown in figure 5.4.

In reality, some CAs are much more secure and cannot be compromised easily. Compromising exactly the same CAs for which the website has digital certificates in a limited time makes it even more difficult. Figure 5.4. shows the probability of success vs number of CAs when the probability is 0.5. In actual, the probability is much low.

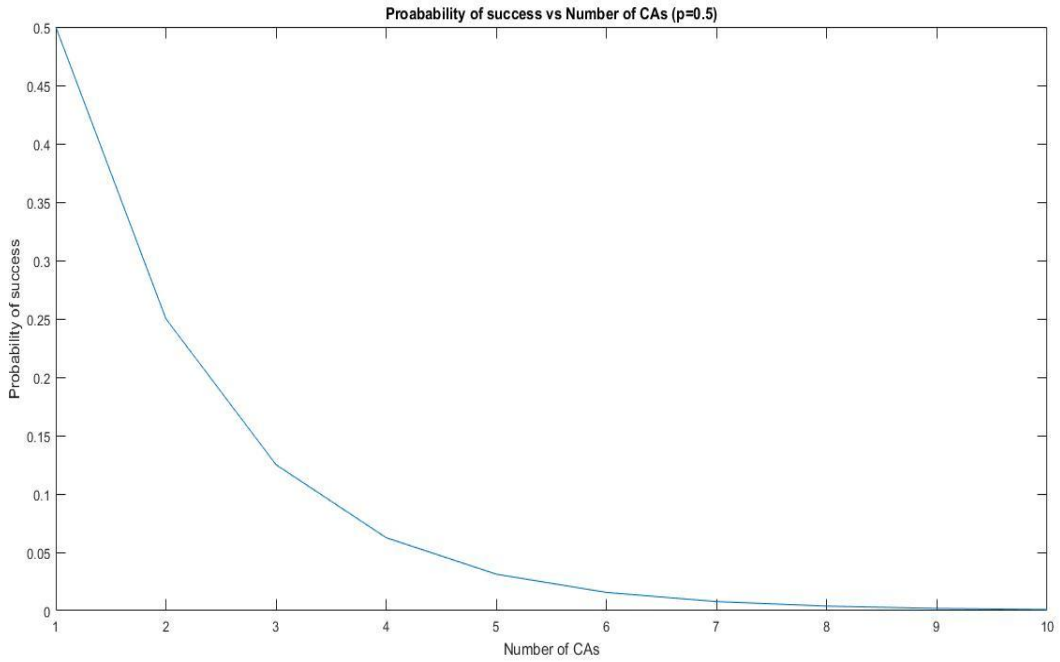

Figure 5.4. Probability of success vs Number of CAs (p=0.5)

The use of multiple CAs is compatible with the enhancements proposed. It is not the replacement of any of the enhancements but complements to the current CA ecosystem.

CHAPTER VI

SECURE COMMUNICATION PROTOCOLS USING GOLDBACH PARTITIONS

Secure Communication Protocol using Single Third Party

Imagine a scenario in which two parties want to communicate securely and rely on a third party for the key exchange. The third party knows all the secrets and if it is compromised or becomes malicious; this will completely break the security of the system and hence relying on a single third party is risky [19]. There can be two solutions to this problem:

1. Do not share the secret completely with the third party.
2. The secret is distributed among multiple third parties.

Keeping in view the above scenario, first we present a secure communication protocol, which uses a single third party for the key exchange using Goldbach partitions. We exchange the key in such a way that even if third party reads all the messages sent to it for the key exchange, it is unable to determine the key. We assume that third party will not try to look at the communication between two parties. We assume that Alice (A) and Bob (B) choose private keys and shares hashed value of their private keys $h(k_a)$ and $h(k_b)$, respectively with the third party (T) during the registration period.

The protocol is as follows:

1. The third party chooses a random even number *n* and send it to both the parties Alice and Bob using their private keys. The random number is sum of two primes.

$$T: n = p_1 + p_2$$

$$T \longrightarrow A: n \oplus h(k_a)$$

$$T \longrightarrow B: n \oplus h(k_b)$$

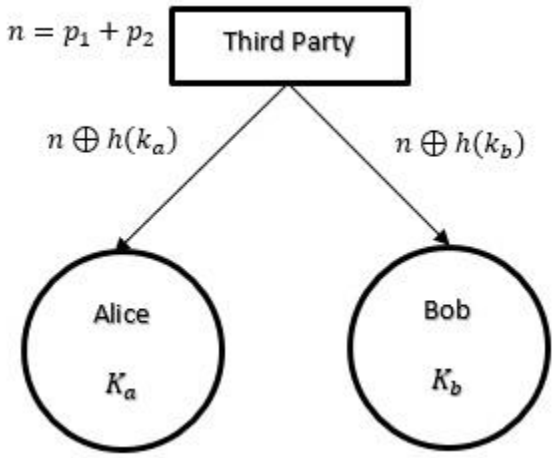

Figure 6.1. Third party sends random number *n* to both Alice and Bob

2. Alice chooses a random number *r* and sends it to the Bob using the hashed value of *n.* As Bob already knows the value of *n*, it can decrypt the value of *r.* Alice and Bob adds *r* and *n* using Bit XOR.

$$A \longrightarrow B: r \oplus h(n)$$

$$A, B: q = r \oplus n$$

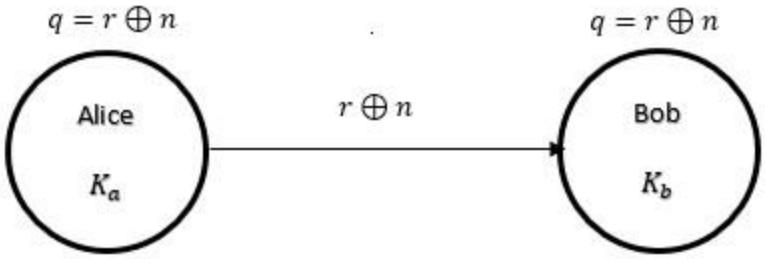

Figure 6.2. Alice sends random number *r* to Bob

3. Alice sends a number *m* to T using the hashed value of *r*, which is the number of partition of *q*. Third party (T) forwards the same to Bob. As Bob already knows the value of *r* and hence can decrypt the value of *m.* The partition *m* will be used as a secret key for the communication.

$$A \longrightarrow T: m \oplus h(r)$$

$$T \longrightarrow B: m \oplus h(r)$$

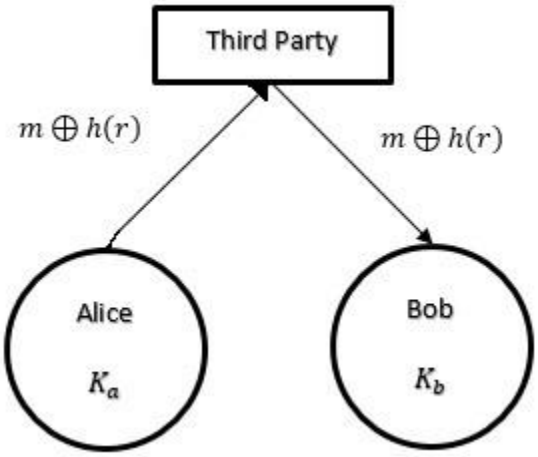

Figure 6.3. Alice sends *m* to third party and secure connection is established

We illustrate this through an example:

1. T chooses 42 as *n,* which has four Goldbach partitions: 5+37, 11+37, 13+29, and 19+23 and sends it to Alice and Bob using hashes of their private keys.

2. Alice chooses 18 as *r*, hashes it using 42, and sends it to Bob who then performs XOR on *r* and *n*: 42 $\oplus$ 18= 56. The Goldbach partitions of 56 are 53+3, 43+13, and 37+19.

3. Alice chooses 3 as *m*, hashes it using 18, and sends it to Bob via Third party. Hence, the pair (37, 19) is used as a secret key.

Security Analysis

Since, the third party only knows about *n* and *m*, it cannot determine the key and even if it is compromised, the key remains safe. To derive the key, it needs to know *n*, *r*, and *m*. If the third party is looking at the communication between Alice and Bob, it can determine the key and is not consistent with our assumption. The Eve needs to intercept the communication between Alice and Bob and also needs to compromise the third party in order to determine the key. If Eve wants to impersonate as a legitimate third party and reads all the messages sent to it for the key exchange, it will still be unable to determine the key. The hashing function used in the protocol is assumed to be cryptographically strong.

We can also calculate the value of *q*, which is sum of *r* and *n,* as a sum of all the values of *r* sent in the previous communications. It will require Eve to know all the previous values of *r* in order to determine the key.

$$\mathrm{q} = r_1 \oplus r_2 \oplus r_3 \ldots \oplus r_n \oplus n$$

Secure Communication Protocol using Multiple Third Parties

We can use Goldbach partitions to exchange the key using multiple third parties. This will require Eve to compromise all the third parties taking the part in key exchange to determine the key. Any third party cannot determine the key unless they compromise all other third parties involved in the process. The minimum number of third parties is three.

We assume that Alice (A) and Bob (B) choose their private keys and shares hashed value of their private keys $h(k_a)$ and $h(k_b)$, respectively with third parties (T) during the registration period. The private keys chosen are different for each third party. The symbols used in this protocol have nothing to do with the symbols used in the previous protocol.

The protocol is as follows:

1. Consider *n* third parties taking place in the key exchange. They all generate an even random number *r* and send it to both Alice and Bob using hashes of their private keys.

$$T_1, T_2, T_3, \dots T_n : r_1, r_2, r_3, \dots r_n$$

$$T_1, T_2, T_3, \dots T_n \longrightarrow A : r_1 \oplus h(k_a), r_2 \oplus h(k_a), r_3 \oplus h(k_a) \dots r_n \oplus h(k_a)$$

$$T_1, T_2, T_3, \dots T_n \longrightarrow B : r_1 \oplus h(k_b), r_2 \oplus h(k_b), r_3 \oplus h(k_b) \dots r_n \oplus h(k_b)$$

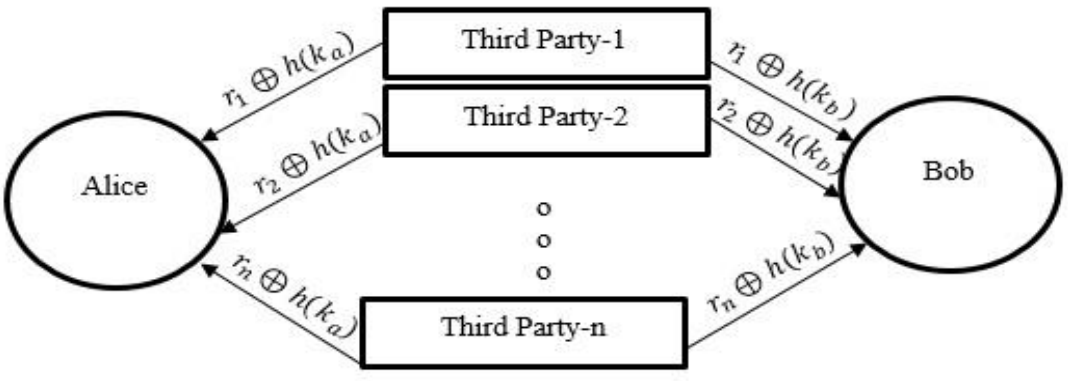

Figure 6.4. All third parties send even random number *r* to both Alice and Bob

2. Both Alice and Bob perform XOR on all the values received from all the third parties.

$$q = r_1 \oplus r_2 \oplus r_3 \ldots \oplus r_n$$

3. Alice chooses a random number which is less than or equal to the value of number of partitions of *q* and splits it into the *d* partitions. Then, it sends one partition to each third party using the hash value of specific *r*, which is then forwarded to Bob using the hash of its private key.

$$q = d_1 \oplus d_2 \oplus d_3 \ldots \oplus d_n$$

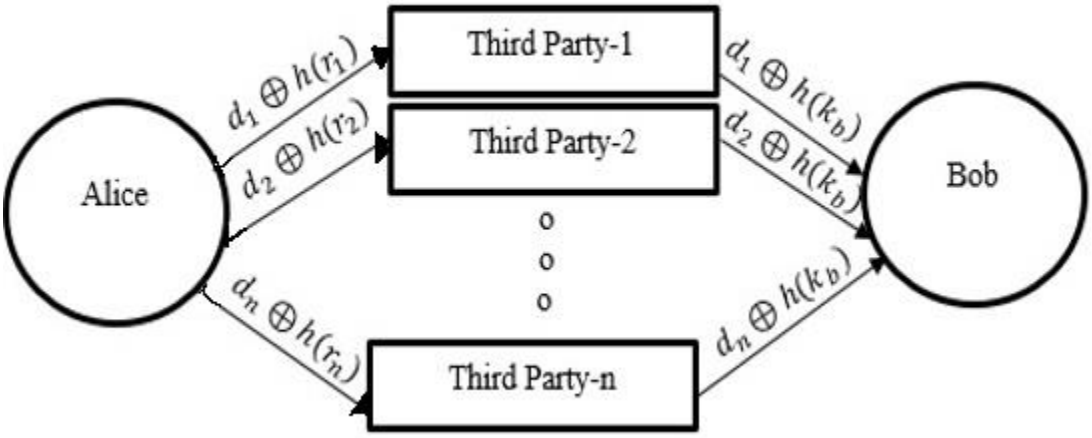

Figure 6.5. Alice sends partitions of a random number to Bob via each third party

4. Bob performs XOR on all the partitions received from all the third parties. Hence, the key is exchanged and can be used for the communication.

The protocol presented can be modified to XOR the values of only specific *m* out of *n* parties to derive the key. The parties chosen for key exchange can be changed from time to time for every session.

Consider three third parties taking part in key exchange.

1. T_1, T_2, T_3 chooses 24, 30, and 40 as r_1, r_2, r_3 and send to Alice and Bob using hash of their private keys.

   Goldbach partitions of 24: 5+19, 7+17, 11+13

   Goldbach partitions of 30: 7+23, 11+19, 13+17

   Goldbach partitions of 40: 3+37, 11+29, 17+23

2. Alice and Bob compute q = 011000 $\oplus$ 011110 $\oplus$ 101000 = 101110

   q =46 ; Goldbach partitions of 46: 43+3, 41+5, 29+17
3. Alice chooses two as the partition number and sends using hash value of specific r.
4. Bob receives all the partitions and computes

   s = 011100 $\oplus$ 111010 $\oplus$ 100100 = 000010 = 2

   Hence (41,5) partition and is used as secret key.

Security Analysis

Since every third party only knows about its own communication with Alice and Bob, it cannot determine the key on its own. The key can only be compromised if all the third parties taking part in the key exchange are compromised at the same time. Using cryptographically strong hash function will prevent man-in-the-middle attack. The Eve needs to impersonate all the third parties as legitimate third parties taking part in the key exchange, read all the messages and determine the key. The number of third parties taking place in key exchange may be varied for each session. Another variation is to involve *n* parties but compute the output of only *k* parties.

CHAPTER VII

SECRET SHARING WITH THIRD PARTIES USING THE PIGGY BANK PROTOCOL

A third party can be defined as an entity which facilitates communication between two parties. Third parties are widely employed for distributing secrets. This makes the use of third parties critical and make them main target for distributed systems and cryptography research.

Imagine a scenario in which two parties completely trust the third party and use it to exchange secrets, but it is compromised or it becomes dishonest, then the system's security will be completely broken. We can use multiple third parties to reduce the risk and increase security.

Imagine a cell phone or a device with sensitive information which may be a threat to national security is in the custody of law enforcing agency but they cannot access it because it is password protected and the data wipes out automatically if the wrong password is entered few times. The cell phone or device owner company has also refused to provide any backdoor for information to be accessed stating the reason that the backdoor may come in wrong hands and hence the security of phones or devices in someone's physical ownership may be compromised. This in fact was what happened in the case between FBI and Apple last year.

Keeping such scenarios in mind, the use of secret sharing scheme becomes logical. We present a protocol to distribute the secret shares with two third parties using the piggy bank protocol (Figure. 7.1).

The piggy bank protocol [28] can be used between a third party and the user as it can provide authentication, secures double-lock cryptography and counters MIM attack [29]. There have been several secret sharing schemes proposed in which the secret is partitioned and shared to various distributed servers. Shamir's secret sharing scheme [30] is based on polynomial interpolation and maps the data on y-axis whereas another data partitioning scheme [31] is based on the roots of a polynomial on the x-axis.

For background, consider the specific *(k, n)* threshold scheme based on polynomial roots where *k=2 and n=3*. The data is partitioned into three pieces and stored on three servers. Data reconstruction requires access to at least two out of three servers. This scheme may be called as *(2,3)* threshold scheme. The servers are chosen in such a way that only one piece of data goes to each server and this will secure the user's data from being accessed by any of them until at least two of them agree and combine their pieces to recreate the original data

Proposed Protocol

In this protocol, we encrypt the secret share as well as the decryption key used for decrypting the secret share. Hence, we first decrypt the decryption key of the secret share and then the secret share. The terms used in the protocol are explained as:

Decryption key of the encrypted secret share decryption key: This is used to decrypt the decryption key of the secret share in which the latter is encrypted and the former is not.

Decryption key of the encrypted secret share: This is used to decrypt the secret share.

Step by Step Protocol

1. Secret shares are created on user's device using some appropriate process.

2. One unique secret share is encrypted and sent to third party 1 along with the decryption key of the encrypted secret share decryption key. Each secret share is encrypted with a unique key.
3. The decryption key of the encrypted secret share is encrypted and sent to third party 2.
4. Third party 1 sends unique recipient only one encrypted secret share and the decryption key of the (encrypted) secret share decryption key.
5. Third party 2 sends the recipient the decryption key of the (encrypted) secret share and the former is also encrypted.
6. Step 2 to Step 5 are repeated until every recipient receives its secret share.
7. Every recipient can decrypt the secret share by first decrypting the encrypted secret share decryption key using the decryption key received from third party 1 and then decrypting the encrypted secret share using the key received from third party 2.

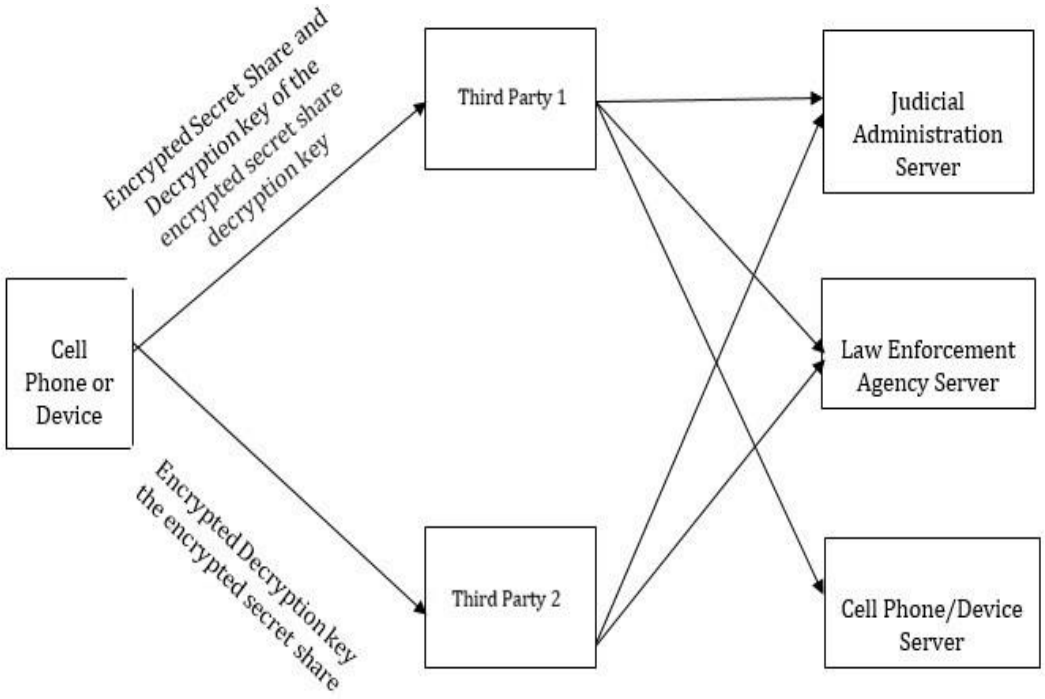

Figure. 7.1. Protocol for distributing secret shares Involving two third parties

The above mentioned protocol can be applied in a variety of applications i.e. Cloud Computing. Now, we present an application protocol in which the cell phone or device data is accessible to law enforcing agencies using secret sharing scheme and two third parties

Application: Accessing a Cell Phone/Device

The data is partitioned into three pieces and stored on the servers of law enforcing agencies, judicial administration and cell phone/device owner company in such a way that only one piece of secret goes to each authority (Figure 7.2). Data reconstruction requires access to at least two out of three servers. Law enforcing agencies cannot access cell phone or device until one of other two combine their secret share with them. It is assumed that cell Phone or device owner companies would never share their secret share until it is the matter of national security because they would never want to lose the trust of people. In case, it is the matter of national security and cell phone or device owner company has denied to share its secret, the judicial administration can give its share to law enforcing agencies and access the cell phone or device. The number of servers can be *n* and any (*k, n*) secret share can be used to implement this. The decision to select authorities whom the secrets are sent is also on the will of the implementation authority.

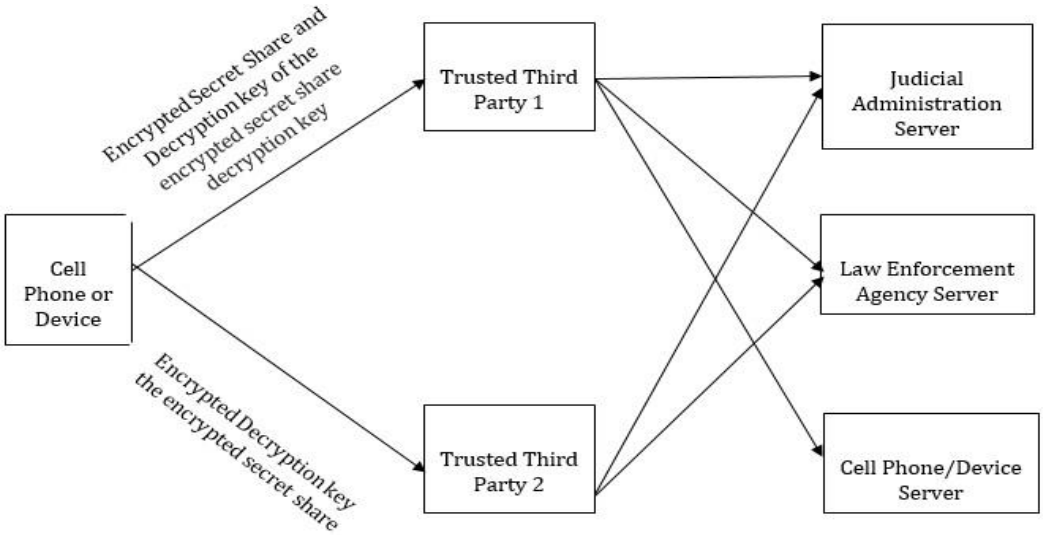

Fig. 7.2. Protocol for accessing cell phone using secret sharing scheme and two third parties

Quantum Secret Sharing

Now we come to quantum cryptography [32]. The concept of quantum secret sharing has been investigated and different schemes have been proposed. The procedure in which a message is split into various parts and all of them are required to read the message is described in [33]. A mechanism has also been shown to implement this procedure using GHZ states. The advantage of quantum case is the detection of eavesdropper because his presence will introduce errors. The concept of *(k, n)* threshold scheme is to divide a secret quantum state into *n* shares and any set of *k* shares are required to reveal the secret, but any set of $k-1$ or less shares reveal absolutely nothing about the secret [34]. Quantum no-cloning theorem puts a restriction on threshold schemes and requires $n < 2k$ and in our case $n=3$, $k=2$ and hence $3 < 2\ (2) = 3 < 4$, thus obeys the no-cloning theorem. The quantum mechanical version of the piggy bank cryptography protocol is presented in [35] and relevant problems of information in [36,37,38].

Security Analysis

In proposed protocol, we are using two Third parties and sending encrypted secret share via one third party and its decryption key which is also encrypted via other third party. This provides the confidentiality to the secret shares even in the case when one of the Third parties is compromised. If third party 2 is compromised, the Eve will only have the encrypted secret share decryption key and not the secret shares. As the secret share decryption key is also encrypted, it provides another level of security to the secret shares and ultimately the secret. If third party server 1 is compromised, the Eve will only have encrypted secret shares but not its decryption key and using better encryption technique and long decryption key will also make the brute force attack impossible for the Eve.

Use of the piggy bank protocol between a third party and the user prevents man-in-the-middle (MIM) attack if this protocol uses cryptographically strong RNG (Random Number Generation) algorithm or complex hash function. Both parties can introduce onetime pad or any other technique

for authentication resource to recognize the misinformation from Eve, if she gets holds of the secret message. The Piggy bank protocol can also provide authentication. Data partitioning scheme is implicitly secure as less than 2 shares cannot deduce any information about the secret. Eve has to access at least two servers to construct the secret.

The protocols presented in this chapter are published in [39]; the passages and figures are reproduced with slight modifications.

CHAPTER VIII

CONCLUSIONS

This thesis presents two secure communication protocols based on Goldbach partitions. The first secure communication protocol uses a single third party for the key exchange, and the key is exchanged in such a way that even if the third party is compromised, the secret key remains safe assuming that the third party does not intercept the communication between the communicating parties. The second secure communication protocol uses multiple third parties to distribute the trust and eliminate the dependency on single third party. The secret key is not compromised until all third parties are compromised. In case we are using k out of n parties to exchange the key, at least k third parties need to be compromised.

Goldbach partitions can be used to find large prime numbers in an efficient manner under certain conditions. We investigated Goldbach partitions and found that they require less number of attempts to find prime numbers as compared to the incremental search method. The local peaks of percentage of Goldbach partitions are found to be at sequential product of primes and their integer multiples.

We also present a new scheme to distribute secret shares with two third parties using the piggy bank cryptographic paradigm. We present an application of the given protocol to give law enforcing agencies access to sensitive information present on a cell phone or a device using secret sharing scheme. These ideas for classical systems may also be applied to quantum schemes.

We propose the use of multiple certificate authorities to issue digital certificates and validate the identity of organizations and individuals on the Internet. The use of multiple certificate authorities raises the difficulty in stealing or forging digital certificates, thereby enhancing the trust and security of the system.

We have also devised a way to sieve prime numbers which uses less number of operations as compared to the Sieve of Eratosthenes.

REFERENCES

[1] Rivest, R., Shamir, A., Adleman, L. A method for obtaining digital signatures and public key cryptosystems, Communications of ACM 21 (2) (1978) 158–164.

[2] Diffie, W., Hellman, M.E. New directions in cryptography. IEEE Transactions on Information Theory 22 (1976) 644–654.

[3] Rabin, M. O. Probabilistic algorithm for testing primality. Journal of Number Theory 12.1 (1980): 128-138.

[4] Conrad, K. Fermat'S Test. HTTPS://www.math.uconn.edu/~kconrad/blurbs/ugradnumthy/fermattest.pdf

[5] Agrawal, M., Kayal, N. and Saxena, N. PRIMES is in P. Annals of Mathematics (2004): 781-793.

[6] Menezes, A. J., Van Oorschot Paul C., and Scott A. Vanstone. Public-Key Parameters. Handbook of Applied Cryptography. Boca Raton: CRC, 1997. 133-68. Print.

[7] Horsley, S. ΚΟΣ ΚΙΝΟΝ ΕΡΑΤΟΣ Θ ΕΝΟΥ Σ. or, The Sieve of Eratosthenes. Being an Account of His Method of Finding All the Prime Numbers, by the Rev. Samuel Horsley, FRS. Philosophical Transactions (1683-1775) 62 (1772): 327-347.

[8] Atkin, A., and Bernstein, D. Prime sieves using binary quadratic forms. Mathematics of Computation 73.246 (2004): 1023-1030.

[9] Ramaswami Aiyar, V. Sundaram's sieve for prime numbers. The Mathematics Student 2.2 (1934): 73.

[10] Kak, S. Goldbach partitions and sequences. Resonance 19.11 (2014): 1028-1037.

[11] Cherlopalle, D. and Kak, S. Goldbach triples and key distribution. arXiv preprint arXiv:1209.0135 (2012).

[12] Kanchu, K.R. Secure key transfer protocol using Goldbach sequences. Oklahoma State University, 2013.

[13] Daemen, Joan, and Vincent Rijmen. AES proposal. First Advanced Encryption Standard Candidate Conference, NIST. 1998.

[14] Karn, Phil, Simpson, W.A. and Metzger, P. The ESP triple DES transform. (1995).

[15] Anoop, M. S. Elliptic Curve Cryptography, An Implementation Guide. online Implementation Tutorial, Tata Elxsi, India 5 (2007).

[16] Kerry, Cameron F., and Patrick D. Gallagher. Digital signature standard (DSS). FIPS PUB (2013): 186-4.

[17] Brown, D. Standards for efficient cryptography, SEC 1: elliptic curve cryptography. Released Standard Version 1 (2009).

[18] Federal Information Processing Standard (FIPS), PUB. 180-4. Secure hash standard (SHS)," March (2012).

[19] Küpçü, A. Distributing trusted third parties. ACM SIGACT News 44.2 (2013): 92-112.

[20] Pittu, G. R. Generating primes using partitions. arXiv preprint arXiv:1505.00253 (2015).

[21] Rea, S. Alternatives to Certification Authorities for a Secure Web. DigiCert, Inc. (2013), hthird parties://www.rsaconference.com/writable/presentations/file_upload/sec-t02_final.pdf

[22] Hallam-Baker, P. and Stradling, R. DNS certification authority authorization (CAA) resource record. (2013).

[23] Wilson, B. Ballot 187 - Make CAA Checking Mandatory. CAB Forum. N.p., 22 Mar. 2017. Web. 05 July 2017.

[24] Evans, C., Palmer, C. and Sleevi, R. Public key pinning extension for HTTPS. No. RFC 7469. 2015.

[25] Laurie, B., Langley, A. and Kasper, E.. Certificate transparency. No. RFC 6962. 2013.

[26] O'Neill, M. E. The genuine sieve of Eratosthenes. Journal of Functional Programming 19.1 (2009): 95-106.

[27] Kak, A. Lecture 13: Certificates, Digital Signatures, and the Diffie-Hellman Key Exchange Algorithm. Purdue University.

[28] Kak, S. The piggy bank cryptographic trope. Infocommunications Journal 6: 22-25 (2014)

[29] Kak, S. Authentication Using Piggy Bank Approach to Secure Double-Lock Cryptography. arXiv preprint arXiv:1411.3645 (2014).

[30] Shamir, A. How to share a secret. Communications of the ACM 22: 612-613 (1979)

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

[32] Kak, S. A three-stage quantum cryptography protocol. Foundations of Physics Letters 19: 293-296 (2006)

[33] Hillery, M., V. Bužek, and A. Berthiaume. Quantum secret sharing. Physical Review A 59: 1829 (1999)

[34] Cleve, R., D. Gottesman, and H.-K. Lo. How to share a quantum secret. Physical Review Letters 83: 648 (1999)

[35] Chodisetti, N. A Piggybank Protocol for Quantum Cryptography. arXiv preprint arXiv:1406.2285 (2014).

[36] Kak, S. The three languages of the brain: quantum, reorganizational, and associative. In Learning as Self-Organization, K. Pribram and J. King (editors). Lawrence Erlbaum Associates, Mahwah, NJ, 185-219 (1996)

[37] Kak, S. Quantum information and entropy. Int. Journal of Theoretical Physics 46, 860-876 (2007)

[38] Kak, S. The initialization problem in quantum computing. Foundations of Physics 29, 267-279 (1999)

[39] Memon, Adnan. Secret Sharing With Trusted Third Parties Using Piggy Bank Protocol. arXiv preprint arXiv:1608.05097 (2016).

VITA

Adnan Ahmed Memon

Candidate for the Degree of

Master of Science

Thesis: SECURE COMMUNICATION PROTOCOLS, SECRET SHARING AND AUTHENTICATION BASED ON GOLDBACH PARTITIONS

Major Field: Electrical Engineering

Biographical:

Education:

Completed the requirements for the Master of Science in Electrical Engineering at Oklahoma State University, Stillwater, Oklahoma in July, 2017.

Completed the requirements for the Bachelor of Engineering in Telecommunication at Sukkur Institute of Business Administration, Sukkur, Sindh, Pakistan in 2012.

Experience: Junior Lecturer/Lab Engineer at Bahria University Karachi Campus, Karachi, Pakistan (02 Years)

Professional Memberships: PEC (Pakistan Engineering Council)