Cryptography Using Chinese Remainder Theorem

A Dissertation for

MAT-651 Discipline Specific Dissertation

Credits: 16

Submitted in partial fulfilment of Masters Degree

M.Sc. in Mathematics

by

Ms. SHRUSHTEE DEVIDAS BETODKAR

22P0410033

ABC ID : 351-975-342-792

201905578

Under the Supervisor of

Dr. M. TAMBA

School of Physical & Applied Sciences

Mathematics Discipline

GOA UNIVERSITY

APRIL 2024

Examined by: Seal of the School

DECLARATION BY STUDENT

I hereby declare that the data presented in this Dissertation report entitled, "Cryptography Using Chinese Remainder Theorem", is based on the results of investigations carried out by me in the Mathematics Discipline at the School of Physical & Applied Sciences, Goa University under the Supervision of Dr. M. Tamba and the same has not been submitted elsewhere for the award of a degree or diploma by me. Further, I understand that Goa University will not be responsible for the correctness of observations / experimental or other findings given the dissertation.

I hereby authorize the University authorities to upload this dissertation on the dissertation repository or anywhere else as the UGC regulations demand and make it available to any one as needed.

Signature: Betodkar

Student Name: Shrushtee Devidas Betodkar

Seat no: 22P0410033

Date: 08/05/2024

Place: GOA UNIVERSITY

COMPLETION CERTIFICATE

This is to certify that the dissertation report "Cryptography Using Chinese Remainder Theorem" is a bonafide work carried out by Ms. Shrushtee Devidas Betodkar under my supervision in partial fulfilment of the requirements for the award of the degree of Master of Science in Mathematics in the Discipline Mathematics at the School of Physical & Applied Sciences , Goa University.

Signature : [illegible]

Supervisor : Dr. M. Tamba

Date: 08/05/2024

School of Physical and Applied Sciences ★ Goa University

School Stamp

[illegible]

Signature of HoD of the Dept

Date: 10/05/2024

Place: Goa University

PREFACE

This Project Report has been prepared in partial fulfilment of the requirement for the Subject: MAT - 651 Discipline Specific Dissertation of the programme M.Sc. in Mathematics in the academic year 2023-2024.

The topic assigned for the research report is: "Cryptography Using Chinese Remainder Theorem." This survey is divided into seven chapters. Each chapter has its own relevance and importance. The chapters are divided and defined in a logical, systematic and scientific manner to cover every nook and corner of the topic.

FIRST CHAPTER :

The Introductory stage of this Project report is based on overview of the Chinese Remainder Theorem and Cryptography.

SECOND CHAPTER:

This chapter deals with Chinese Remainder Theorem [3]. In this topic we have discussed origins of CRT, application of CRT in ancient China during 3^{rd} century. Then we have the properties of linear congruences, the Chinese Remainder Theorem and it's proof with uniqueness. We have also solved some examples of congruence using Chinese Remainder Theorem.

THIRD CHAPTER:

The third chapter is about Cryptography. In this topic we have discussed about applications of cryptography, how cryptography underpins various aspects of digital communication and information security, types of cryptography and their application, definitions and security goals of cryptography.

FOURTH CHAPTER:

This chapter deal with Random Number Generating [8]. Here we have discussed how a random number is generated using an example.

FIFTH CHAPTER.

This chapter is about RSA Cryptosystem [1] [4] and [9]. Here we have discussed about the original RSA cryptography, the lemmas which are used. Then we have RSA algorithm, in which we have key generation, encryption and decryption. Later we have solved some problems using the RSA algorithm. Then we have double encryption, multiple encryption and its examples.

SIXTH CHAPTER.

In this chapter we deal with Digital Signature Standard [5] and [9]. Here we have discussed about DSS which is also known as the Digital Signature Algorithm. Next we have the process to generate keys, signature and verification and definition of generator,hash function and hash value. Finally we have solve an example using the set up.

SEVENTH CHAPTER.

This chapter is all about different Threshold Cryptosystem or as it called Threshold Cryptography [2], [4], [6] and [7] . First we have original Asmuth-Bloom SSS with an example. Next we have discussed about the modified version of Asmuth-Bloom SSS. After that we have RSA function which is different from the original RSA and threshold RSA, which is a combination of RSA function, Asmuth-Bloom SSS and DSA. Then we

have ElGamal function and threshold ElGamal, which is similar to Threshold RSA, and finally we have discussed about Paillier and threshold Paillier, which is also similar to other threshold functions.

ACKNOWLEDGEMENTS

First and foremost, I would like thank Mr. Brandon Fernandes for introducing me to Cryptography on which my project is based on. Next I would like to express my gratitude to my Mentor, Dr. M. Tamba, who was a continual source of inspiration. He pushed me to think imaginatively and urged me to do this homework without hesitation. His vast knowledge, extensive experience, and professional competence in Number Theory enabled me to successfully accomplish this project. This endeavour would not have been possible without his help and supervision.

ABSTRACT

This study looks at how the Chinese Remainder Theorem (CRT) can improve cryptographic protocols by making arithmetic more efficient. It aims to speed up cryptographic processes, especially in situations where resources and efficiency are important. The study examines how CRT can be used in different cryptographic schemes, like public-key cryptography and digital signature algorithms, to cut down on computing work, boost performance, and make cryptographic systems stronger overall.

Keywords: Chinese Remainder Theorem, Cryptography, Encrypting, Decrypting, RSA Cryptosystem, Asmuth-Bloom SSS, ElGamal Function, Paillier Function, Threshold Cryptosystem, Random Number Generator.

Table of contents

List of figures

Notations and Abbreviations

$\mathbb{Z}$	Set of integers
$p!$	p factorial
$\phi(n)$	Euler's totient function
$PubKey$	Public key
$PriKey$	Private key
g	Generator
$H(M)$	Hash Message
h	Hash code or digest
φ	coalition of t users
φ'	coalition of $t-1$ users
λ	lcm of $(p-1)(q-1)$
$\bar{s}$	Incomplete decryptor of s
K_s	Corrector of s

Chapter 1

INTRODUCTION

The Chinese Remainder Theorem (CRT) is a key concept in number theory that has important applications in mathematics and computing. It helps solve sets of modular arithmetic equations and gives us a deeper understanding of how integers and their remainders work together.

Originating from ancient Chinese mathematics and later formalized by Carl Friedrich Gauss in the West, the CRT establishes conditions under which a system of congruences can be simultaneously solved. Its fundamental principle asserts that if we know the remainders when an integer is divided by several pairwise coprime integers, then we can uniquely determine the remainder when that integer is divided by the product of those integers.

So basically, if we know what’s left when we divide a number by different numbers that don’t share factors, we can figure out what’s left when we divide by all those numbers multiplied together.

Formally, given a system of congruences:

$$\begin{aligned}
x &\equiv a_1 \pmod{n_1} \\
x &\equiv a_2 \pmod{n_2} \\
&\vdots \\
x &\equiv a_r \pmod{n_r}
\end{aligned}$$

where $m_1, m_2, \cdots m_n$ are pairwise coprime integers and $a_1, a_2, \cdots a_n$ are arbitrary integers, the CRT guarantees the existence and uniqueness of a solution modulo.

The applications of the CRT are wide-ranging, spanning cryptography, error correction, scheduling algorithms, and beyond. In cryptography, for instance, it forms the basis of various encryption and decryption schemes, enabling secure communication protocols. In error-correcting codes, it aids in the design of efficient algorithms for data transmission and storage. Moreover, the CRT finds applications in diverse areas such as number theory, algebraic geometry, and computer science.

The beauty of the CRT lies not only in its theoretical elegance but also in its practical utility. Its algorithmic formulation makes it a powerful tool for problem-solving and algorithm design, with implications extending to both theoretical and applied research. As such, the Chinese Remainder Theorem continues to inspire exploration and innovation across disciplines, making it a subject of enduring interest and significance in mathematical research and beyond.

Cryptography is the mathematical foundation on which one builds secure systems. It studies ways of securely storing, transmitting, and processing information. Understanding what cryptographic primitives can do, and how they can be composed together, is necessary to build secure systems, but not sufficient. Several additional considerations go into the design of secure systems, and they are covered in various Berkeley graduate courses on security.

In this course we will see a number of rigorous definitions of security, some of them requiring seemingly outlandish safety, even against entirely implausible attacks, and we shall see how if any cryptography at all is possible, then it is also possible to satisfy such extremely strong notions of security. For example, we shall look at a notion of security for encryption in which an adversary should not be able to learn any information about a message given the ciphertext, even if the adversary is allowed to get encodings of any messages of his choice, and decodings of any ciphertexts of his choices, with the only exception of the one he is trying to decode.

We shall also see extremely powerful (but also surprisingly simple and elegant) ways to define security for protocols involving several untrusted participants.

Learning to think rigorously about security, and seeing what kind of strength is possible, at least in principle, is one of the main goals of this course. We will also see a number of constructions, some interesting for the general point they make (that certain weak primitives are sufficient to make very strong constructions), some efficient enough to have made their way in commercial products.

Chapter 2

CHINESE REMAINDER THEOREM

2.1 The Origin of Chinese Remainder Theorem

The Chinese Remainder Theorem also called as CRT was formulated in Ancient China in the 3^{rd} century by Sun Zi. The CRT was developed to assist with practical problems that arose in astronomy, construction and commerce. The Chinese used the algorithm to calculate calendars, compute the number of soldiers when lined up in different configurations, or constructing the wall of a building or the base of a house. It is basically a mathematical tool that helps you solve a system of equations where each equation has a different modulo. A modulo is just a fancy term for the remainder you get when you divide a number by another number.

The CRT was developed to assist with practical problems that arose in astronomy, construction and commerce. The Chinese used the algorithm to calculate calendars, compute the number of soldiers when lined up in different configurations, or constructing the wall of a building or the base of a house. According to Shen Kangsheng, a translated problem from Master Sun's Mathematical Manual (problem 26, Volume 3) reads as follows:

"There are certain things whose number is unknown.
A number is repeatedly divided by 3, the remainder is 2;
divided by 5, the remainder is 3, and divided by 7, the remainder is 2.
What will the number be?"

From the given information we can construct three equations:

$$x = 3n + 2, \ \ x = 5m + 3, \ \ x = 7l + 2$$

The equations form the following congruences:

$$x \equiv 2 \pmod 3, \ \ x \equiv 3 \pmod 5, \ \ x \equiv 2 \pmod 7$$

For a system of equations such as these, the Chinese Remainder Theorem provides a unique solution up to a certain modulus. The Chinese Remainder Theorem (CRT) can determine an integer from its residues modulo by a set of pairwise relatively prime moduli. One of the earliest applications of the CRT arose from the computation of calendars in ancient China.

In ancient time, Chinese used 60 days (or years) as a period to record days (or years) starting from Jiazi to the end of Guihai. Sixty days are a period (or a week), and the recording was repeated. "If Winter Solstice of a certain year occurred r_1 days after shangyuan and r_2 days after the new moon, then that year was N years after shangyuan; hence arose the system of congruences: $N \equiv r_1 \pmod{60}, \ aN \equiv r_2 \pmod{b}$ where a is the number of days in a tropical year and b the number of days in a lunar month".

The CRT offered a simple method of determining the solution to this type of simultaneous congruences. While the CRT appeared in ancient texts as early as the 3rd century, a complete solution for solving linear congruences was not given until much later. A generalized solution for the CRT appeared in the mid thirteenth century.

The generalized method for solving systems of linear congruences was written by Qin Jiushao in 1247 and published in a comprehensive book of Chinese mathematics titled, Mathematical Book in Nine Chapters. In the first section he presents a remarkable generalization of the Chinese Remainder Theorem, which allowed for congruences to be solved even if the moduli were not relatively prime.

The CRT algorithms worked on a counting board with rods that represented numbers, to find the modular multiplicative inverse of a given congruence. For cases where the moduli were not relatively prime, his method looked for common factors to reduce the moduli. Qin Jiushao then applied a method of continued division to solve the resulting system of congruences.

The Euclidian algorithm is a computational process that computes the greatest common divisor of two positive integers. It's a very old method that first appeared in Euclid's Elements, which was written approximately 300 BC. In this method, long division is repeated, using the quotient and remainder. The method is exhausted when the remainder is zero and the greatest common divisor is the last non-zero remainder.

2.2 Properties of Linear Congruence

Lemma 2.2.0.1. *A congruence modulo m forms an equivalence relation.*
That is, $\forall, a, b, c \in \mathbb{Z}$*, the following holds:*
Reflexive*:* $a \equiv a \pmod{m}$
Symmetric*:* $a \equiv b \pmod{m}$*, then* $b \equiv a \pmod{m}$
Transitive*:* $a \equiv b \pmod{m}$ *and* $b \equiv c \pmod{m}$*, then* $a \equiv c \pmod{m}$

Proof. **Reflexive:** Let a and m be any positive integers.

We have $a = a \implies a - a = 0$.

Let's take 0 as $0m$ since any number that is multiplied by 0 is 0 so we have,

$a - a = 0m \implies m|(a - a)$ and 0 is divisible by any nonzero integer.

Therefore, $a \equiv a \pmod{m}$

Symmetric: Let a, b and m be any integers.

Suppose $a \equiv b \pmod{m}$ and $\exists\ n \in \mathbb{Z}$, we can write the equation as

$a = mn + b \implies a - b = mn$

$\implies m|(a - b)$

Therefore, if we multiply $m|(a - b)$ by -1 we get,

$m|(-1)(a - b)$

$\implies m|(b - a)$

$\implies b \equiv a \pmod{m}$

Transitive: Let a, b, c and m be any integers.

Suppose for $a \equiv b \pmod{m}$ and $b \equiv c \pmod{m}$, $\exists\ n, p \in \mathbb{Z}$ such that

$a = mn + b \implies a - b = mn$ and

$b = mp + c \implies b - c = mp$

$\implies m|(a - b)$ and $m|(b - c)$

Now, using linear combination theorem, we have

$m|(a - b + b - c)$

$\implies m|(a - c)$

$\implies a \equiv c \pmod{m}$ □

Lemma 2.2.0.2. ***Congruence Modulo*** m ***Addition:***

If $a \equiv b \pmod{m}$ *and* $c \equiv d \pmod{m}$*, then* $a + c \equiv b + d \pmod{m}$*.*

Proof. Suppose $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$ then by definition,

$\exists\ k_1, k_2 \in \mathbb{Z}$ such that $a - b = k_1 m$ and $c - d = k_2 m$.

Now, adding both of these equations we get,

$(a-b)+(c-d) = k_1m + k_2m$

$\implies (a+c)-(b+d) = m(k_1+k_2)$.

Now, let $k = k_1 + k_2 \implies (a+c)-(b+d) = km$

$\implies a+c \equiv b+d \pmod{m}$

□

Lemma 2.2.0.3. ***Congruence Modulo** m **Subtraction:***
If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then $a-c \equiv b-d \pmod{m}$.

Proof. Suppose $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$ then by definition,

$\exists\ k_1, k_2 \in \mathbb{Z}$ such that $a-b = k_1m$ and $c-d = k_2m$.

Now, subtract both of these equations we get,

$(a-b)-(c-d) = k_1m - k_2m$

$\implies (a-c)-(b-d) = m(k_1-k_2)$

Now, let $k = k_1 - k_2 \implies (a-c)-(b-d) = km$

$\implies a-c \equiv b-d \pmod{m}$

□

Lemma 2.2.0.4. ***Congruence Modulo** m **Multiplication:***
If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$, then $ac \equiv bd \pmod{m}$.

Proof. Suppose $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$ then by definition,

$\exists\ k_1, k_2 \in \mathbb{Z}$ such that $a-b = k_1m$ and $c-d = k_2m$.

Now, consider

$$\begin{aligned} ac-bd &= ac-cb+cb-bd \\ &= c(a-b)+b(c-d) \\ &= ck_1m + ck_2m, \\ &= m(ck_1 = ck_2) \end{aligned}$$

Let $k = (ck_1 = ck_2)$ for the above $c, k_1, k_2 \in \mathbb{Z}$

$\implies ac - bd = mk \implies ac \equiv bd \pmod{m}$

□

Lemma 2.2.0.5. ***Congruence Modulo m to the k^{th} Power:***

If $a \equiv b \pmod{m}$ and $k \in \mathbb{N}$, then $a^k \equiv b^k \pmod{m}$.

Proof. Suppose $a \equiv b \pmod{m}$, where $a, b \in \mathbb{Z}$, then $\exists\ k_1 \in \mathbb{Z}$ such that

$$a - b = k_1 m$$

Now consider $(a^k - b^k) = (a-b)(a^{k-1} + a^{k-2}b + a^{k-3}b^2 + \cdots + b^{k-1}) \cdots\cdots (1)$

Substitute $a - b = k_1 m$ into equation (2)

$\implies (a^k - b^k) = k_1 m(a^{k-1} + a^{k-2}b + a^{k-3}b^2 + \cdots + b^{k-1})$

Let $r = k_1(a^{k-1} + a^{k-2}b + a^{k-3}b^2 + \cdots + b^{k-1})$ where $r \in \mathbb{Z}$

$\implies a^k - b^k = rm$, therefore $m|(a^k - b^k)$

$\implies a^k \equiv b^k \pmod{m}$

□

Lemma 2.2.0.6. ***Modular Multiplicative Inverse:***

Let a, m be positive integers such that $gcd(a,m) = 1$. Then a has a multiplicative inverse modulo m, and it is unique modulo m. In other words, $a \in \mathbb{Z}$ is invertable mod m, if and only if the $gcd(a,m) = 1$.

Proof. Suppose a is invertible mod m and $m > 1$, then $\exists\ x, y \in \mathbb{Z}$ such that

$ax \equiv 1 \pmod{m}$

$\implies ax = my + 1$

$\implies ax - 1 = my$ or $ax - my = 1$

$\implies m|(ax - 1)$

Let $d = gcd(a,m)$, then $d|a$ and $d|m$ so,

$d|(ax - my)$

$\implies d|1$

$\implies gcd(a,m)|1$

$\implies gcd(a,m) = 1$

$\impliedby$ Suppose $gcd(a,m) = 1$, then by Bezout's identity $\exists\ x, y \in \mathbb{Z}$ such that

$ax + my = 1$

$\implies ax = 1 - my$

$\implies ax - 1 = (-y)m$

$\implies m|(ax-1)$

$\implies ax \equiv 1 \pmod{m}$

□

Lemma 2.2.0.7. ***Bezout's Identity states***:

If the greatest common divisor of a and b is d, then $ax + by = d$ for x and $y \in \mathbb{Z}$.

Proof. Consider the set S of all positive linear combination of a and b.

$$S = \{au + bv \mid au + bv > 0,\ \ u, v \in \mathbb{Z}\}$$

For $a > 0$, taking $u = 1, v = 0$

$au + bv = -a > 0 \implies S \neq \phi$

By definition we know that S contains positive integers and by applying well ordering principle $\exists\ x, y \in \mathbb{Z}$ such that $d = ax + by \cdots\cdots (1)$

Claim: d is gcd of a and b

$d \in \mathbb{Z} \implies d > 0$

Applying division algorithm to a and b

$$a = qd + r, \quad 0 \leq r \leq d \cdots\cdots (2)$$

$$\implies r = a - qd$$

$$\implies r = a - q(ax + by) \text{ by equation (1)}$$

$$\implies r = a - qax - qby$$

$$\implies r = a(1 - qx) + b(-qy)$$

If $r > 0$, then $r \in S$ because $1 - qx$ and qy.

But $r < d$ and d is least element of S.

Hence this is a contradiction.

$a = qd \implies q|a$

Now if c is any other posiyive common divisor of a and b i.e. $c|a,\ c|b$.

Assume $c|a\ ,c|b$

$\implies c|(ax+by)$

$\implies c|d$

$\therefore$ By definition of gcd, $d = gcd(a,b)$

$\square$

2.3 The Proof of Chinese Remainder Theorem

Theorem 2.3.0.1. ***Chinese Remainder Theorem:***

Let $n_1, n_2, \ldots, n_r$ be positive integers such that gcd(n_i, n_j) = 1 for $i \neq j$. Then the system of linear congruences

$$\begin{aligned} x &\equiv a_1 \pmod{n_1} \\ x &\equiv a_2 \pmod{n_2} \\ &\vdots \\ x &\equiv a_r \pmod{n_r} \end{aligned}$$

has a simultaneous solution, which is unique modulo $N = n_1 n_2 \ldots n_r$.

Proof. Let $n_1, n_2, \ldots, n_r$ be positive integers such that gcd(n_i, n_j) = 1 for $i \neq j$.

Suppose N be the product of all positive integers that is

$$N = \prod_{i=1}^{r} n_i$$

then our N_k will be

$$N_k = \frac{N}{n_k} = n_1, n_2, \dots n_{k-1}, n_{k+1}, \dots, n_r$$

where N_k is the product of all integers of n_i except for $n_k \ \forall \ i = 1, 2, \dots, r$.

Since n_k is not a factor of the product N_k, we have

$$gcd(N_k, n_k) = 1.$$

Since $gcd(N_k, n_k) = 1$, we know that N_k has an inverse modulo n_k.

$\therefore$ It is possible to solve the following congruence

$$N_k x \equiv 1 \pmod{n_k} \implies x \equiv N_k^{-1} \pmod{n_k} \tag{2.1}$$

and we call this unique solution x_k. So, by construction we have

$$N_k \equiv 1 \pmod{n_k}.$$

But $N_k x_k \equiv 0 \pmod{n_j}$ for $j \neq k. \cdots\cdots (1)$

This is because, N_k is a product of all integers except n_k, which means n_j is present in N_k and therefore congruent to 0 modulo $n_j \ \forall \ i \neq j$.

Let $x = x_1 N_1 a_1 + x_2 N_2 a_2 + \dots, x_r N_r a_r$.

If we compute x modulo n_k, then every term which is not k will be 0 by (1).

$\therefore \ x = 0 + 0 + \dots + x_k N_k a_k + 0 \cdots + 0 \equiv x_k N_k a_k \pmod{n_k}$.

However, x_k was chosen to satisfy the congruence $N_K x \equiv 1 \pmod{n_k}$.

Therefore, $x \equiv a_k \pmod{n_k}, \ \forall \ 1 \leq k \leq r$.

Hence $x = x_1 N_1 a_1 + x_2 N_2 a_2 + \dots, x_r N_r a_r$ is a solution of the system.

Uniqueness: Suppose x and y are two solutions of the proposed system of congruence.

$\implies x \equiv a_k \pmod{n_k}$ and $y \equiv a_k \pmod{n_k}, \ \forall \ 1 \leq k \leq r$

Using properties of subtraction congruence, we have

$x - y \equiv 0 \pmod{n_k}, \ \forall \ 1 \leq k \leq r$

$\implies n_k | x - y$

$\implies (x - y)$ is a multiple of $n_k, \ \forall \ 1 \leq k \leq r$

And since N_k is the product of all integers where n_i are pairwise relatively prime.

$\implies (x-y)$ is a multiple of N, which means N divides $(x-y)$.

$\implies x \equiv y \pmod{N}$.

Therefore, the initial pair of congruences are the same modulo N.

□

Example: 2.3.0.2. *Solve the following congruence*

$$x \equiv 5 \pmod{11}$$
$$x \equiv 4 \pmod{23}$$
$$x \equiv 15 \pmod{37}$$

Solution: Given that the moduli are pairwise relatively prime, a unique solution is given by the Chinese Remainder Theorem

We have $n_1 = 11,\ n_2 = 23$ and $n_3 = 37$.

$\therefore\ N = n_1 \cdot n_2 \cdot n_3 = 11 \cdot 23 \cdot 37 = 9361$

So $a_1 = 5,\ a_2 = 4$ and $a_3 = 15$

Now we find N_k and $N_k = \frac{N}{n_k}$ where $k = 1,2,3$

$\therefore\ N_1 = \frac{N}{n_1} = \frac{11 \cdot 23 \cdot 37}{11} = 23 \cdot 37 = 851$

Similarly we have $N_2 = 407$ and $N_3 = 253$

By using equation (2.1) we find the inverse modulo of N_k.

$851x_1 \equiv 1 \pmod{11}$

$\implies ((11 \times 77) + 4)x_1 \equiv 1 \pmod{11}$

$\implies 4x_1 \equiv 1 \pmod{11}$

$\implies x_1 \equiv 3 \pmod{11}$

Similarly we have $x_2 \equiv 13 \pmod{23}$ and $x_3 \equiv 6 \pmod{37}$

The solution x is as follows

$$\begin{aligned} x &= N_1 a_1 x_1 \;+\; N_2 a_2 x_2 \;+\; N_3 a_1 x_3 \\ &= 851 \times 5 \times 3 + 407 \times 4 \times 13 + 253 \times 15 \times 6 \\ &= 12765 \;+\; 21164 \;+\; 22770 \\ &= 56699 \end{aligned}$$

$\therefore$ 56699 will be a solution to the system of linear congruence.

However, the solution is unique modulo 9361. $\implies x \equiv 56699 \mod 9361$

$\implies x \equiv 533 \pmod{9361}$.

$\therefore$ 533 is the smallest possible positive solution to the system of linear congruences.

Chapter 3

CRYPTOGRAPHY

Definition 3.0.0.1. Cryptography is technique of securing information and communications through use of codes so that only those person for whom the information is intended can understand it and process it.

The term "cryptography" derives from the Greek words "kryptos," meaning hidden, and "graphein," meaning writing. Essentially, cryptography involves transforming plaintext (readable data) into ciphertext (encrypted data which is unreadable) through the use of algorithms and keys. Only authorized parties possessing the correct key can decrypt the ciphertext and recover the original plaintext. This process ensures that sensitive information remains confidential even if intercepted by unauthorized entities.

Historically, cryptography has been employed for centuries, dating back to ancient civilizations such as the Egyptians and Greeks. However, its modern applications have expanded dramatically with the advent of computers and the internet.

Today, cryptography underpins various aspects of digital communication and information security, including:

1. **Secure Communication:** Cryptographic protocols, such as SSL/TLS (Secure Sockets Layer/Transport Layer Security), enable secure communication over networks like the internet. These protocols encrypt data transmitted between parties, preventing eavesdropping and tampering.

2. **Data Integrity:** Cryptographic hash functions generate fixed-size hash values from input data, which serve as unique digital fingerprints. By comparing hash values before and after transmission, recipients can verify the integrity of received data and detect any alterations.

3. **Authentication:** Cryptographic techniques, such as digital signatures, allow entities to verify the authenticity and origin of messages or documents. Digital signatures provide assurance that data has not been forged or modified by unauthorized parties.

4. **Access Control:** Cryptography is integral to authentication mechanisms like passwords and access tokens, which control access to sensitive resources. Encryption techniques help safeguard authentication credentials and prevent unauthorized access to systems and data.

5. **Privacy Preservation:** Techniques like public-key cryptography enable secure communication between parties without the need for a shared secret key. This facilitates confidential exchanges, such as online transactions or private messaging, while minimizing the risk of interception or impersonation.

The field of cryptography continues to evolve rapidly, driven by advancements in technology and the ever-changing landscape of cyber threats. Researchers and practitioners are constantly developing new cryptographic algorithms, protocols, and applications to address emerging challenges and enhance security measures.

3.1 Types of Cryptography

There are two different kinds of cryptography systems:

1. Private key cryptography also known as symmetric.

2. Public key cryptography also known as asymmetric.

Definition 3.1.0.1. Symmetric Cryptography: Sender and receiver agree on secret key that both use to encrypt and decrypt the message. Here decryption is simply the opposite of encryption. Such a system is called symmetric cipher because encryption and decryption is symmetrical.
For example, Caesar cipher, Shift Cipher, Hill Cipher, Vigenère Cipher, etc.

Common Applications of Symmetric Cryptography

- **Data Encryption:** Symmetric-key algorithms, such as the Data Encryption Standard (DES), Advanced Encryption Standard (AES), and Triple DES (3DES), are widely used for encrypting sensitive data in storage and transmission.

- **File and Disk Encryption:** It is used for encrypting files, folders, and entire disk volumes to protect data stored on computers or portable storage devices.

Definition 3.1.0.2. Asymmetric Cryptography: There are two different but mathematically linked keys the public key and the private key. The publicly disclosed key is accessible by everyone, while the private key is known only to the receiver. Using the public key, messages can be encrypted by anyone but can only be decoded with the use of the private key.
For example, RSA Cryptosystem, DSS, etc

Common Applications of Asymmetric Cryptography

- **Digital Signatures:** Public-key algorithms, such as RSA and ECDSA, are used for creating digital signatures to authenticate the origin and integrity of electronic documents, transactions, and software.
- **Key Exchange:** Public-key cryptography facilitates secure key exchange mechanisms, such as Diffie-Hellman key exchange, enabling parties to establish shared secret keys for symmetric-key encryption without prior communication.
- **Identity Verification:** Public-key certificates, issued by Certificate Authorities (CAs), are used for verifying the identities of individuals, organizations, and websites in digital environments, such as SSL certificates for secure websites.

Definition 3.1.0.3. Public Key:- A public key is a component of asymmetric cryptography used in cryptographic systems. It is shared openly and used for encryption or verification. Messages encrypted with a public key can only be decrypted by the corresponding private key, providing a secure means of communication.

Definition 3.1.0.4. Private Key:- A private key is the secret part of an asymmetric cryptographic key pair. It is kept confidential and is used for decrypting messages encrypted with the corresponding public key. The private key is crucial for ensuring the security of digital communication and transactions.

3.2 Security Goals

In cryptography, various security goals are essential to ensure the confidentiality, integrity, authenticity, and availability of data and communication channels.

Confidentiality ensures that sensitive information remains private and inaccessible to unauthorized parties. Cryptographic techniques such as encryption are used to scramble plaintext data into ciphertext, making it unreadable without the proper decryption key. Common encryption algorithms like AES (Advanced Encryption Standard) and RSA (Rivest-Shamir-Adleman) are employed to achieve confidentiality in various applications, including secure communication, data storage, and online transactions.

Integrity ensures that data remains unaltered and intact during storage, transmission, or processing. Cryptographic integrity mechanisms, such as cryptographic hash functions and digital signatures, are used to detect unauthorized modifications or tampering of data. Hash functions generate fixed-size hashes or message digests from input data, which are then compared to verify data integrity. Digital signatures provide proof of the origin and integrity of electronic documents or messages, allowing recipients to verify the authenticity of the sender and detect any unauthorized changes.

Authenticity verifies the identity of users, devices, or entities involved in communication or transactions. Public-key cryptography plays a crucial role in establishing authenticity through digital signatures and certificates. Digital signatures are created using the sender's private key and can be verified using the corresponding public key, providing assurance of the sender's identity and the integrity of the message. Certificates issued by trusted Certificate Authorities (CAs) bind public keys to the identities of individuals, organizations, or websites, enabling parties to verify authenticity and establish secure communication channels.

Non-repudiation prevents parties from denying their actions or transactions. Digital signatures provide strong non-repudiation guarantees by associating cryptographic proof of identity with electronic documents or transactions. Once a digital signature is applied to a document, the signer cannot later deny their involvement or the validity of the signature, providing evidence in case of disputes or legal proceedings.

Availability ensures that authorized users have timely and uninterrupted access to resources and services. While not directly a cryptographic goal, cryptographic techniques can be used to mitigate denial-of-service (DoS) attacks and ensure the availability of critical systems and communication channels. Cryptographic protocols and algorithms must be designed to withstand attacks and disruptions that may compromise availability, ensuring reliable operation in adverse conditions.

By addressing these security goals, cryptographic systems and protocols provide the foundation for secure communication, data protection, and trust in digital environments. Organizations and individuals rely on cryptographic techniques to safeguard sensitive information, maintain privacy, and mitigate risks associated with cyber threats and attacks.

Chapter 4

RANDOM NUMBER GENERATING

Definition 4.0.0.1. Random Number:- A Random Number refers to a value that is generated unpredictably and uniformly from a defined range of numbers. These random numbers are crucial for various cryptographic operations, such as key generation, digital signatures, and encryption.

Definition 4.0.0.2. Randomness:- Randomness refers to the absence of any discernible pattern or predictability in a sequence of events or values. In other words, when something is random, it occurs without any apparent order or regularity.

A random number generator (RNG) is a computational or physical process that generates a sequence of numbers or symbols that cannot be reasonably predicted better than by a random chance. RNG's are used in various applications such as cryptography, simulations, gaming, and statistical sampling. They can be implemented in software using algorithms or hardware-based methods.

Random number generators (RNG's) are used in various fields for several reasons:

1. **Simulations and Modeling:** RNG's are crucial for creating realistic simulations and models in fields like physics, engineering, and computer science. They allow researchers to mimic complex systems and behaviors where randomness is a factor.

2. **Games and Gambling:** RNG's are extensively used in gaming and gambling applications to ensure fairness and unpredictability. Whether it's a video game, a casino game, or a lottery, RNG's provide the randomness necessary for a fair and entertaining experience.

3. **Cryptography:** RNG's are used in generating cryptographic keys and in encryption algorithms to enhance security. Randomness is fundamental in creating unpredictable keys that are resistant to hacking attempts.

4. **Statistical Sampling:** In statistical analysis and experimentation, RNG's are used for random sampling to ensure representative samples and accurate analysis results.

5. **Security Protocols:** RNG's are utilized in various security protocols, such as generating session keys, authentication tokens, and random challenges to thwart unauthorized access and attacks.

6. **Artificial Intelligence and Machine Learning:** RNG's can be used in training and testing machine learning models to introduce variability and randomness, aiding in robustness and generalization.

Overall, RNG's are essential tools across a wide range of applications where randomness is required or beneficial.

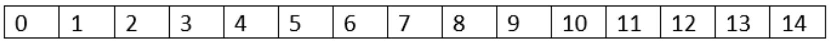
There are some techniques that are used to generating random numbers such as *Pseudorandom Number Generator* and *Linear Congruent Generator* also *Cryptographically Generated Random Numbers*, etc. In this chapter we are using Chinese Remainder Theorem for the purpose of generating random numbers. In essence, CRT says it is possible to reconstruct integers in a certain range from their residue modulo a set of pairwise relatively prime modulo.

4.1 Random Number Generating Using CRT

The CRT is an algorithm with so many applications in mathematics, computing is the main area of its application and moreover, recently it is being used in cryptography also. But in the field of cryptosystem, the algorithm is used for functionality for modular computation.

Example: 4.1.0.1. *Produce random numbers where $p = 5$ and $q = 3$ by using CRT.*

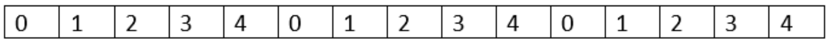
Solution: Let p and q be any large prime numbers for modulo m where m is the product of two primes.



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Figure 4.1: Table for modulo 15

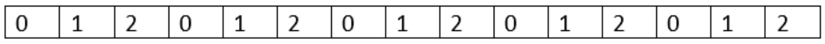
Now for $m = pq$ where we have a function $X : \mathbb{Z}_m \to \mathbb{Z}_p \times \mathbb{Z}_q$
We separate the two primes into two different tables. So we have
$i \pmod 5$



0	1	2	3	4	0	1	2	3	4	0	1	2	3	4

Figure 4.2: Table for modulo 5

and $i \pmod 3$



0	1	2	0	1	2	0	1	2	0	1	2	0	1	2

Figure 4.3: Table for modulo 3

Now we are going to plot the below table in such a way that we obtain the values of $\mathbb{Z}_5$ and $\mathbb{Z}_3$ using the values of $\mathbb{Z}_{15}$.
So we need the table for $i \pmod 5$ as

$i(mod5)$	0	1	2	3	4
0	0	1	2	3	4
1	0	1	2	3	4
2	0	1	2	3	4

Figure 4.4: Modulo 5 looks in tabular form

and $i \pmod 3$ as

$i(mod3)$	0	1	2	3	4
0	0	0	0	0	0
1	1	1	1	1	1
2	2	2	2	2	2

Figure 4.5: Modulo 3 looks in tabular form

Now combining both of them the table for $i \pmod{15}$ would look like
Now we take the value 0 in the first cell since for both modulo 5 and modulo 3 the first value is 0.
Now for to find the value for cell (0,1) so we obtain the numbers 0 for $i \pmod 3$ and 1 for $i \pmod 5$ we take 6. Similarly we find other values for the rest of the cell which give us the following table.
It is clear that on every point a number is being generated for every value.

$i(mod5)$ / $i(mod3)$	0	1	2	3	4
0	(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
1	(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
2	(2,0)	(2,1)	(2,2)	(2,3)	(2,4)

Figure 4.6: Combined tabular form of modulo 5 and 3

$i(mod5)$ / $i(mod3)$	0	1	2	3	4
0	0	6	12	3	9
1	10	1	7	13	4
2	5	11	2	8	14

Figure 4.7: Tabular form of Modulo 15 using above figures

4.1.1 Practical Application and Use Of Random Numbers

Random number generators have application in gambling, statistical sampling, computer simulation, cryptography, and other areas where a random number is useful in producing an unpredictable result. The random numbers also is useful for the prevention of reply attack also for counter measures.

In the area of cryptography and network security random numbers can be used as the Nonce and it can be attached with the message packets on the sender end as the identification of each and every packet. In the above technique the CRT generate unique random numbers for every key value which can be attached with the message.

4.1.2 Conclusion

Chinese reminder theorem, provide benefits in computing, mathematics and also in the field of cryptography, where the algorithm provides relief in case of modular computation and also in case of generating the random numbers. But we had made a complex study of this theorem to develop a new concept of producing random numbers. As we know that random numbers has a wide application in cryptography to make security of the network communication more and stronger so that the intruder can hot hamper the network security.

We have also these random to show how these random numbers can be used in secure message transmission preventing packet reply which is an attack in which an intruder captures a message packet and reply it again to gain access to information or system application. We have used random numbers which are unique for a session of message transmission. At the receiver side these random numbers are checked and if the number has been received earlier also it means that the packet is replayed and discarded.

Chapter 5

RSA CRYPTOSYSTEM

5.1 The Original RSA Cryptography

The RSA cryptosystem is commonly used to authenticate digital signatures and protect data such as customer information and transaction data. It can be used to create key exchanges that establish a secure, encrypted communication channel. It has numerous applications in banking, telecommunications and ecommerce.

The RSA cryptosystem was created in 1977 by MIT professors, Ron Rivest, Adi Shamir, and Leonard Adleman. Their last names form the acronym RSA. It is one of the first public-key or asymmetric cryptosystems.

The RSA method is secure because it is difficult to factor the products of two large primes. “RSA encryption works under the premise that the algorithm is easy to compute in one direction, but almost impossible to reverse”. There are three steps in the RSA algorithm: key generation, encryption and decryption. The Chinese Remainder Theorem plays an integral role in the final phase, the decryption or private key process.

Lemma 5.1.0.1. ***Euler's Totient Theorem:***

If m is natural number and a is an integer such that $gcd(a,m) = 1$, then
$a^{\phi(m)} \equiv 1 \pmod{m}$.

Proof. Let $A = \{n_1, n_2, \cdots n_{\phi(m)}\}$ such that the elements of the set are the numbers relatively prime to m.
It will now be proved that this set is the same as the set
$B = \{an_1, an_2, ...an_{\phi(m)}\}$ where $\gcd(a,m) = 1$.
All elements of B are relatively prime to m so if all elements of B are distinct, then B has the same elements as A.
In other words, each element of B is congruent to one of A.
This means that $n_1 n_2 ... n_{\phi(m)} \equiv an_1 \cdot an_2 ... an_{\phi(m)} \pmod{m}$
$\implies a^{\phi(m)} \cdot (n_1 n_2 ... n_{\phi(m)}) \equiv n_1 n_2 ... n_{\phi(m)} \pmod{m}$
$\implies a^{\phi(m)} \equiv 1 \pmod{m}$ as desired.
Note that dividing by $n_1 n_2 ... n_{\phi(m)}$ is allowed since it is relatively prime to m.

$\square$

Lemma 5.1.0.2. *For any prime p,we have* $(x+y)^p \equiv x^p + y^p \pmod{p}$

Proof. Let p be a prime, and consider the binomial coefficient

$$\binom{p}{i} = \frac{p!}{(p-1)!i!}$$

where $0 < i < p$,. Then p occurs in the numerator and is greater than each number in the denominator. Since p is prime, it follows that $\binom{p}{i}$ is divisible by p.
So $\binom{p}{i}$ is congruent to zero modulo p, $\forall\ 0 < i < p$.
The binomial theorem states that

$$
\begin{aligned}
(x+y)^p &= \sum_{i=0}^{p} \binom{p}{i} x^{p-i} y^i \\
&= \binom{p}{0} x^p + \binom{p}{1} x^{p-1} y + \binom{p}{2} x^{p-2} y^2 + \cdots + \binom{p}{p}-1) x y^{p-1} + \binom{p}{p} y^p \\
&= \binom{p}{0} x^p + 0 + 0 + \cdots + 0 + \binom{p}{0} y^p \\
&= 1x^p + 0 + 0 + \cdots + 0 + 1y^p \\
&= x^p + y^p
\end{aligned}
$$

Since the middle term except the first and last are all congruent to zero modulo p and $\binom{p}{0} = \binom{p}{p} = 1$

$\implies (x+y)^p \equiv x^p + y^p \pmod{p}$ □

Theorem 5.1.0.3. ***Fermat's Little Theorem:***

Let p be a prime. Then,for every positive integer a, we have $a^p \equiv a \pmod{p}$

Proof. The proof is by induction on a. If $a = 1$, then $a^p = 1^p = 1a$. So the result holds for $a = 1$. Inductive step.

Assume that $k^p \equiv k \pmod{p}$

and consider $(k+1)^p$. By previous lemma. we have

$(k+1)^p \equiv k^p + 1 \pmod{p}$.

Hence, by the induction assumption, we obtain

$(k+1)^p \equiv k+1 \pmod{p}$.

By the principle of induction, it follows that $a^p \equiv a \pmod{p}$, for every positive integer a. □

5.1.1 RSA Algorithm

Key Generation:

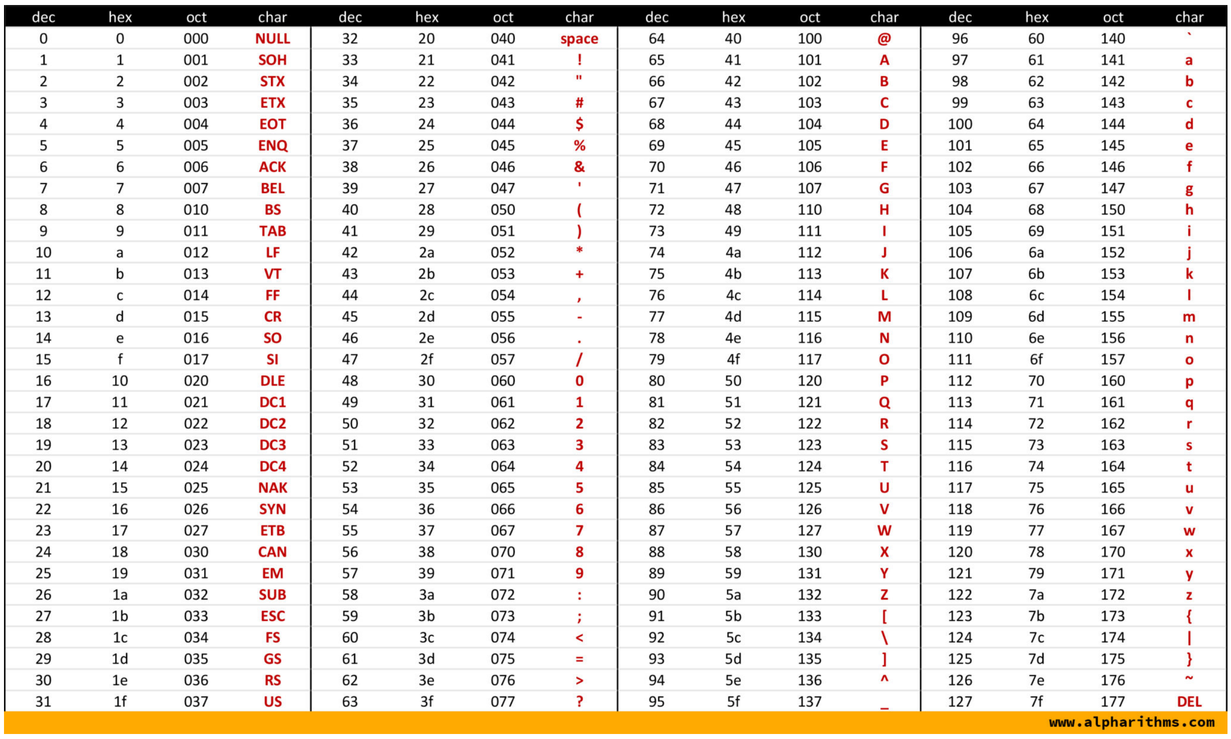
First, we have to prepare keys that will be used by others to encrypt messages, the same information will be used forever. Take two large prime numbers p and q where $p \neq q$.

Let $n = pq$. Let $\phi(n) = (p-1)(q-1)$. Now choose a prime number $e \in [2, \phi(n))$ that is co-prime to ϕ. If the $gcd(e, \phi(n)) = 1$ only then the value e can be used. After compute $d \in [2, \phi(n))$ such that $e.d \equiv 1 \pmod{\phi(n)}$ and d must be co-prime. Finally announce that (e, n) as public key and (d, n) as private key.

Encrypting:

A message is converted into numbers by using ASCII encoding system. Once the plaintext is converted into a series of numbers where $m \le n-1$, the number m is then transformed into an unreadable ciphertext C.The ciphertext is calculated using the encryption equation:

$$C \equiv m^e \pmod{n}.$$



dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char
0	0	000	NULL	32	20	040	space	64	40	100	@	96	60	140	`
1	1	001	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX	34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT	36	24	044	$	68	44	104	D	100	64	144	d
5	5	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS	40	28	050	(	72	48	110	H	104	68	150	h
9	9	011	TAB	41	29	051	)	73	49	111	I	105	69	151	i
10	a	012	LF	42	2a	052	*	74	4a	112	J	106	6a	152	j
11	b	013	VT	43	2b	053	+	75	4b	113	K	107	6b	153	k
12	c	014	FF	44	2c	054	,	76	4c	114	L	108	6c	154	l
13	d	015	CR	45	2d	055	-	77	4d	115	M	109	6d	155	m
14	e	016	SO	46	2e	056	.	78	4e	116	N	110	6e	156	n
15	f	017	SI	47	2f	057	/	79	4f	117	O	111	6f	157	o
16	10	020	DLE	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM	57	39	071	9	89	59	131	Y	121	79	171	y
26	1a	032	SUB	58	3a	072	:	90	5a	132	Z	122	7a	172	z
27	1b	033	ESC	59	3b	073	;	91	5b	133	[	123	7b	173	{
28	1c	034	FS	60	3c	074	<	92	5c	134	\	124	7c	174	|
29	1d	035	GS	61	3d	075	=	93	5d	135	]	125	7d	175	}
30	1e	036	RS	62	3e	076	>	94	5e	136	^	126	7e	176	~
31	1f	037	US	63	3f	077	?	95	5f	137	_	127	7f	177	DEL

www.alpharithms.com

Figure 5.1: ASCII Table

Decryption:

The receiver is given C and must recover the m. Since the receiver knows the factorization on n and with the use of private key (d,n), m is recovered using the decryption equation:

$$m \equiv C^d \pmod{n}.$$

Example: 5.1.1.1. *For any two primes, $p = 53$ and $q = 61$. Encrypt and decrypt the message $m = 72$.*

Solution Since $p = 53$ and $q = 61$ then $n = 3233$.
Let $\phi(n) = (53-1)(61-1) = 3120$.
Choose $e = 7$ since 3 and 5 are not co-prime.
Compute d.

$$\begin{aligned} e.d &\equiv 1 \pmod{\phi(n)} \\ d &\equiv 7^{-1} \pmod{3120} \\ d &\equiv 1783 \pmod{3120} \end{aligned}$$

Therefore, public key is $(7,3233)$ and private key is $(1783,3233)$.
So, let's take $m = 72$ which will be our message.
To encrypt we use $C \equiv m^e \pmod{n}$, so the ciphertext will be $C \equiv 72^7 \pmod{3233}$
$\implies C \equiv 133640 \pmod{3233}$
$\implies C \equiv 1087 \pmod{3233}$
Now, to recover the message from the ciphertext using private key.
To decrypt, we use $m \equiv C^d \pmod{n}$
$\implies m \equiv 1087^{1783} \pmod{3233}$
$\implies m \equiv 45334 \pmod{3233}$
$\implies m \equiv 72 \pmod{3233}$
Therefore, $m = 72$ which is our secret.

5.2 Double Encryption

Definition 5.2.0.1. Double encryption is where two or more independent layers of encryption are enabled to protect against compromises of any one layer of encryption. Using two layers of encryption mitigates threats that come with encrypting data.

The Process: The double encryption process is similar to RSA algorithm, except we do this process twice since we are encrypting an already encrypted message.

Example: 5.2.0.2. *Encrypt the message using p=53 and q=61 where message is m=72. Then encrypt the ciphertext using p=37 and q=41.*

Solution: We have $p = 53$ and $q = 61$ then $n = pq = 3233$.
$\implies \phi(n) = \phi(pq) = (p-1)(q-1) = 3120$
We choose $e = 7$ and computing d, we get, $d \equiv 1783 \pmod{3120}$.
So we have $PubKey_1 = (7, 3233)$ and $PriKey_1 = (1783, 3233)$

Our message is $m = 72$

Encrypting using, $C_1 \equiv m^e \pmod{n}$

$\implies C_1 \equiv 72^7 \pmod{n}$

$\implies C_1 \equiv 1087 \pmod{3233}$

For second encrypting, we have $p = 37$ and $q = 41$

$\implies n = 1517$ and $\phi(n) = 1440$.
Choose $e = 7$ and compute d.

After computing, we get, $d \equiv 1823 \pmod{1440}$

So the next Keys will be, $PubKey_2 = (7, 1517)$ and $PriKey_2 = (823, 1517)$

Take $C_1 = m_1$. So our m_1 will be 1087.

Encrypting using, $C_2 \equiv m_1^e \pmod{n}$, we get

$C_2 \equiv 1087^7 \pmod{n}$

$\implies C_2 \equiv 689 \pmod{1517}$

Now to decrypt we will use $m_1 \equiv C_2^d \pmod{n}$.

Then using $PriKey_2 = (823, 1517)$

We get, $m_1 \equiv 689^{823} \pmod{1517}$

$\implies m_1 \equiv 1087 \pmod{1517}$.

Now decrypting the above message with $PriKey_1 = (1783, 3233)$ and using

since $m_1 = C_1$ we take $m \equiv C_1^d \pmod{n}$, we get, $m \equiv 1087^{1783} \pmod{3233}$

Now we have $m = 72$ which is our message.

Let's see if we can decrypt the message while using the Key_1 first and then Key_2.

So first use Key_1 in $m \equiv 683^{1783} \pmod{3233}$

$\implies m \equiv 2067 \pmod{3233}$

Then we use Key_2 in $m \equiv 2067^{823} \pmod{1517}$

$\implies m \equiv 827 \pmod{1517}$, which is not the original message.

$\therefore$ While decrypting always use most recent key first then the older key.

5.3 Multiple Encryption

Definition 5.3.0.1. Multiple encryption is similar to double encryption. In this we are encrypting an already encrypted message twice or more times, either using the same algorithm or a different algorithm. It is also known as cascade encryption, cascade ciphering, multiple encryption, and superencipherment. Superencryption refers to the outer-level encryption of a multiple encryption.

The Process: Multiple Encryption process is similar to double encryption, except we encrypt the message multiple times to make it more secure.

Example: 5.3.0.2. *Encrypt the message using p=53 and q=61 where message is m=72. Then encrypt the ciphertext using p=37 and q=41. Again encrypt the obtained ciphertext using* $p = 29$ *and* $q = 31$.

Solution: We have $p = 53$ and $q = 61$ then $n = pq = 3233$.
$\implies \phi(n) = \phi(pq) = (p-1)(q-1) = 3120$
We choose $e = 7$ and computing d, we get, $d \equiv 1783 \pmod{3120}$.
So we have $PubKey_1 = (7, 3233)$ and $PriKey_1 = (1783, 3233)$

Our message is $m = 72$

Encrypting using, $C_1 \equiv m^e \pmod{n}$

$\implies C_1 \equiv 72^7 \pmod{n}$

$\implies C_1 \equiv 1087 \pmod{3233}$

For second encrypting, we have $p = 37$ and $q = 41$

$\implies n = 1517$ and $\phi(n) = 1440$.
Choose $e = 7$ and compute d.

After computing, we get, $d \equiv 1823 \pmod{1440}$

So the next Keys will be, $PubKey_2 = (7, 1517)$ and $PriKey_2 = (823, 1517)$

Take $C_1 = m_1$. So our m_1 will be 1087.

Encrypting using, $C_2 \equiv m_1^e \pmod{n}$, we get

$C_2 \equiv 1087^7 \pmod{n}$

$\implies C_2 \equiv 689 \pmod{1517}$

Now to encrypt the message for third time we use $p = 29$ and $q = 31 \implies n = 899$

$\implies \phi(n) = \phi(pq) = (p-1)(q-1) = 840$

We choose $e = 11$ and compute d, we get

$d \equiv 611 \pmod{840}$

So we have $PubKey_3 = (11, 899)$ and $PriKey_3 = (611, 899)$

Take $C_2 = m_2$. So our m_2 will be 689.

Encrypting using, $C_3 \equiv m_2^e \pmod{n}$, we get

$C_3 \equiv 689^{11} \pmod{899} \implies C_3 \equiv 702 \pmod{899}$

Now to decrypt we will use $m_2 \equiv C_3^d \pmod{n}$.

Then using $PriKey_3 = (611, 899)$

We get, $m_2 \equiv 702^{611} \pmod{899} \implies m_2 \equiv 689 \pmod{899}$.

Now to decrypt the above message we will use $m_1 \equiv C_2^d \pmod{n}$.

Then using $PriKey_2 = (823, 1517)$

We get, $m_1 \equiv 689^{823} \pmod{1517} \implies m_1 \equiv 1087 \pmod{1517}$.

Now decrypting the above message with $PriKey_1 = (1783, 3233)$ and using

$m \equiv C_1^d \pmod{n}$, we get, $m \equiv 1087^{1783} \pmod{3233}$

Now we have $m = 72$ which is our message.

Let's see if we can decrypt the message while using the $PrivKey_1$ first and then $PriKey_2$

and last $PriKey_3$.

So first use $PrivKey_1$ in $m \equiv 683^{1783} \pmod{3233}$

$\implies m \equiv 2067 \pmod{3233}$

Then we use $PrivKey_2$ in $m \equiv 2067^{823} \pmod{1517}$

$\implies m \equiv 827 \pmod{1517}$,

Then we use $PrivKey_3$ in $m \equiv 827^{611} \pmod{1517}$

$\implies m \equiv 322 \pmod{899}$, which is not the original message.

Remark: 5.3.0.3. *In the RSA cryptosystem, the security is based on the difficulty of factoring large composite numbers into their prime factors. If a composite number has small prime factors, it becomes vulnerable to factorization attacks, compromising the security of the RSA encryption.*

Remark: 5.3.0.4. *If the underlying mathematical problem that RSA relies on, factoring large composite numbers, is efficiently solved, it could compromise the confidentiality of encrypted messages. This could lead to the unauthorized access to sensitive information and potentially undermine the security of various online systems and communications.*

Chapter 6

DIGITAL SIGNATURE STANDARD

The Digital Signature Standard (DSS) is a cryptographic algorithm used for creating and verifying digital signatures. It was developed by the National Institute of Standards and Technology (NIST) in the United States and was first published in 1993. The purpose of the DSS is to provide a secure method for authenticating the origin and integrity of digital messages or documents.

At its core, the DSS relies on the principles of public key cryptography. It utilizes a pair of mathematically related keys: a private key, which is kept secret by the signer, and a public key, which is freely available for anyone to use. The private key is used to generate the digital signature, while the corresponding public key is used to verify the signature.

The following figure shows how the Digital Signature Standard works.

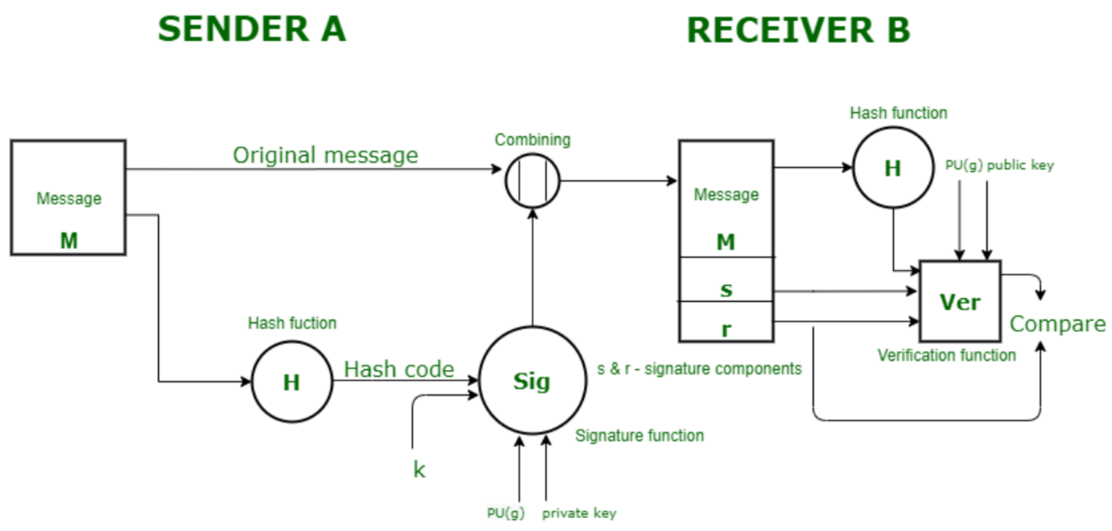


Figure 6.1: Digital Signature Standard

The DSS algorithm involves several steps:

1. **Key Generation:** The signer generates a pair of cryptographic keys - a private key and a corresponding public key.
2. **Signature Generation:** To sign a document or message, the signer uses their private key to perform a mathematical operation on a cryptographic hash of the message. This produces the digital signature.
3. **Signature Verification:** To verify the signature, anyone with access to the public key can perform a similar mathematical operation on the received message and the digital signature. If the result matches the original hash of the message, the signature is considered valid, confirming the authenticity and integrity of the message.

The security of the DSS relies on the computational difficulty of certain mathematical problems, such as factoring large integers or computing discrete logarithms. As long as these problems remain difficult to solve, the digital signatures produced by the DSS are considered secure.

Over the years, the DSS has undergone revisions and updates to address security concerns and improve efficiency. However, it remains a widely used standard for digital signatures in various applications, including electronic transactions, secure communication, and document authentication.

6.1 Digital Signature Algorithm

DSA stands for the Digital Signature Algorithm. It's a widely-used cryptographic algorithm for creating digital signatures. Digital signatures are electronic equivalents of handwritten signatures or stamped seals, but they're based on cryptographic techniques, providing integrity, authenticity, and non-repudiation of digital messages or documents.

DSA is primarily used for creating and verifying digital signatures. Digital signatures generated using DSA allow the recipient to verify the authenticity and integrity of a message or document and confirm that it was indeed signed by the purported sender.

6.1.1 The Set Up

Generation of Keys: To generate a key first we need to find a prime number p where $2^{L-1} < p < 2^L$ and L is an integer and L represents the length of the hash output produced by the chosen hash function. Then find q where q is a prime divisor of $p-1$.
Compute $g \equiv h^{\frac{p-1}{q}} \pmod{p}$ where g is a generator, g and p are public parameters.
We choose x as any random number where $0 < x < q$ and $y \equiv g^x \pmod{p}$ which will be our private key and public key respectively..

Creation of Digital signature: We generate the signature, $r \equiv (g^k \pmod{p}) \pmod{q}$ where k is a random number and $s \equiv (k^{-1}(H(M)+xr)) \pmod{q}$ where $H(M)$ is the hash value and M is the message.
The resulting signature is the pair (r,s)

Verification: Ensure that $0 < r < q$ and $0 < s < q$. If not, the signature is invalid. Now compute $w \equiv s^{-1} \pmod{q}$. Calculate $u_1 \equiv (H(M)w) \pmod{q}$ and
$u_2 \equiv rw \pmod{q}$.
Compute $v \equiv (g^{u_1}y^{u_2} \pmod{p}) \pmod{q}$ If $v = r$ then the signature is not altered.

Definition 6.1.1.1. Generator g: The generator g in the DSA is a key parameter that defines a subgroup within the multiplicative group modulo p, and it plays a crucial role in the generation of public and private keys as well as in the signature generation and verification processes.

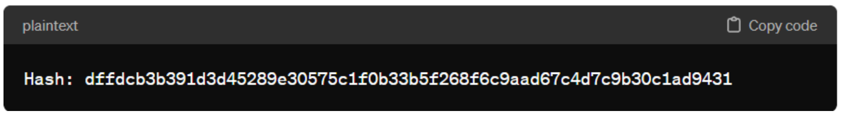
Definition 6.1.1.2. Hash Function: A hash function is a mathematical algorithm that takes an input (or ’message’) and produces a fixed-size string of bytes, typically a hexadecimal number, regardless of the size or complexity of the input. It is denoted by $H(M)$. Hash functions like SHA-256 are commonly used in cryptography for various purposes, including data integrity verification, password hashing, and digital signatures.

Here is an example of a hash function using the SHA-256 algorithm:
Let’s say we have a message "Hello, World!" that we want to hash. This is the $H(M)$ before digest.

```
Message: Hello, World!
```

Figure 6.2: Hash Input

We apply the SHA-256 hash function to this message, which will produce a fixed-size



```
Hash: dffdcb3b391d3d45289e30575c1f0b33b5f268f6c9aad67c4d7c9b30c1ad9431
```

Figure 6.3: The Hash Output

output, typically represented in hexadecimal format.
This hexadecimal string represents the SHA-256 hash of the message "Hello, world!".

Definition 6.1.1.3. Hash value: The output of hash function is often called a hash value or hash code, is typically a hexadecimal number and is of a fixed length, regardless of the size of the input. When computing the generator we work with individual digits after we add the digits to obtain a single number. This only happens when generating g. The hash value is also called hash code or hash digest and it is denoted by h.

Remark: 6.1.1.4. *Thee security of DSA relies on the difficulty of solving the discrete logarithm problem, particularly in the subgroup generated by g.*

Remark: 6.1.1.5. *The use of a per-message random integer k is crucial to prevent private key exposure through a phenomenon called "reusing k".*

Remark: 6.1.1.6. *Care must be taken in choosing parameters and in the generation of random values to avoid vulnerabilities and ensure the security of the DSA implementation.*

Remark: 6.1.1.7. *DSA is generally used with a secure hash function to sign and verify the messages securely.*

Example: 6.1.1.8. *Solve for* $p = 283$ *and* $q = 47$ *with* $H(M) = 41$ *where* $x = 24$ *and* $k = 15$.

Solution: Since $p = 283$ then $q = 47$.
So, $g \equiv h^{(p-1)/q} \pmod{p}$
$\implies (4+1)^{282/47} \pmod{283}$

$\implies 5^6 \pmod{283}$

$\implies g \equiv 60 \pmod{283}$

Given that x is our private key, to find the public key,

we compute $y \equiv g^x \pmod{p}$. $\implies y \equiv 158 \pmod{283}$

To generate the signature, $r \equiv (g^k \pmod{p}) \pmod{q}$

$\implies r \equiv 207 \pmod{47}$

$\implies r \equiv 19 \pmod{47}$

And $s \equiv [k^{-1}(H(M) + xr)] \pmod{q}$

$\implies s \equiv 10934 \pmod{47}$

$\implies r \equiv 30 \pmod{47}$

Then the $Signature = (r,s) = (19,30)$

Let M', r' and s' be the message and signature components respectively received corresponding to M, r and s.

We verify that $0 < r' = 19 < 47$ and $0 < s' = 30 < 47$

$\implies$ we can proceed

So we compute,$w \equiv s'^{-1} \pmod{q}$

$\implies w \equiv 11 \pmod{47}$.

Then $u_1 \equiv (H(M)w) \pmod{q}$

$\implies u_1 \equiv 28 \pmod{47}$

Now we compute, $u_2 \equiv r'w \pmod{q}$

$\implies u_2 \equiv 21 \pmod{47}$

Now to verify if $v = r'$, using $v \equiv (g^{u_1} y^{u_2} \pmod{p}) \pmod{q}$

$\implies v \equiv 207 \pmod{47}$

$\implies v \equiv 19 \pmod{47}$

$\therefore\ v = r'$.

After we check that $v = 19 = r$, then we can accept the signature since it is not altered.

Chapter 7

THRESHOLD CRYPTOSYSTEM

7.1 Threshold Secret Sharing

Secret sharing is a cryptographic technique used to distribute a secret among a group of participants in such a way that only authorized subsets of participants can reconstruct the original secret. The main idea behind secret sharing is to divide the secret into multiple shares or pieces, each distributed to different participants, ensuring that cooperation from a minimum number of participants (known as the threshold) is required to reconstruct the secret. Each piece of the secret is called a share and the person creating the shares is called the dealer.

There are various methods for secret sharing, with one of the most common being Shamir's Secret Sharing scheme, developed by Adi Shamir in 1979. In Shamir's scheme, the secret is represented as a polynomial equation, and each participant is given a point on the polynomial curve. The secret can only be recovered when a predetermined number of shares come together.

Secret sharing has applications in various fields, including cryptography, cybersecurity, and distributed systems, where it is used to enhance security by distributing sensitive information across multiple parties in a way that protects against single points of failure or compromise.

Definition 7.1.0.1. Fault Tolerant: Fault-tolerant computers are systems designed to continue operating without interruption in the event of hardware or software failures. They typically incorporate redundancy in critical components, such as processors, memory, and storage, along with mechanisms for detecting and correcting errors automatically. This ensures that even if a component fails, the system can still function properly, minimizing downtime and ensuring reliability. These systems are commonly used in mission-critical applications where uninterrupted operation is essential, such as aerospace, financial services, and telecommunications.

7.2 Function Sharing Schemes

Function sharing schemes were first introduced by Desmedt et al. in 1989. Key-dependent function is distributed among n people such that any coalition of size t or more can evaluate the function but smaller coalitions cannot. When a coalition S is to evaluate the function, the i^{th} user in S computes his own partial result by using his share y_i and sends it to a platform which combines these partial results. Unlike in a secret sharing scheme, the platform here need not be trusted since the user shares are not disclosed to the platform.

FSSs are typically used to distribute the private key operations in a public key cryptosystem (i.e., the decryption and signature operations) among several parties. Sharing a private key operation in a threshold fashion requires first choosing a suitable SSS to share the private key. Then the subject function must be arranged according to this SSS such that combining the partial results from any t parties will yield the operation's result correctly. This is usually a challanging task and requires some ingenious techniques.

Several solutions for sharing the RSA and ElGamal private key operations have been proposed in the literature. Almost all of these schemes are based on the Shamir SSS, with the only exception of one scheme based on Blakley. Lagrangian interpolation used in the secret reconstruction phase of Shamir's scheme makes it a suitable choice for function sharing, but it also provides several challenges. One of the most significant challenges is the computation of inverses in $\mathbb{Z}_{\phi(N)}$ for sharing the RSA function where $\phi(N)$ should not be known by the users. The first solution to this problem, albeit a relatively less efficient one, was proposed by Desmedt and Frankel, which solved the problem by making the dealer compute all potentially needed inverses at the setup time and distribute them to users mixed with the shares. A more elegant solution was found a few years later by De Santis et al. They carried the arithmetic into a cyclotomic extension of $\mathbb{Z}$, which enabled computing the inverses without knowing $\phi(N)$. Finally, a very practical and ingenious solution was given by Shoup where he removed the need of taking inverses in Lagrangian interpolation altogether by a slight modification in the RSA signature function.

To the best of our knowledge, so far no function sharing schemes based on the Asmuth-Bloom SSS have been proposed in the literature. We show in this paper that the Asmuth-Bloom scheme in fact can be a more suitable choice for function sharing than its alternatives, and the fundamental challanges of function sharing with other SSSs do not exist for the Asmuth-Bloom scheme.

7.3 Asmuth-Bloom Secret Sharing Scheme

One of the threshold SSS, which utilizes the Chinese Remainder Theorem is the Asmuth-Bloom secret sharing scheme. This scheme was presented in 1983 by Charles Asmuth and Charles Bloom. The Asmuth-Bloom SSS uses a weighted threshold access structure. The weighted threshold allows for a more perfect privacy, meaning no unauthorized group of participants can obtain information about the secret. The weighted threshold schemes are essentially a way of modifying the qualified sets of the access structure.

7.3.1 The Original Asmuth-Bloom Secret Sharing Scheme

Dealer Phase: The secret d is shared among the group of n users and t is our recovery threshold.

A set of pairwise relatively prime numbers $m_0 < m_1 < \cdots < m_n$ where $m_0 > d$ are choosen such that

$$\prod_{i=1}^{t} m_i > m_0 \prod_{i=1}^{t-1} m_{n-i+1}$$

Combiner Phase: Let $M = \prod_{i=1}^{t} m_i$ and our $y = d + Am_0$ where A is any random integer. To find shares we take $y_i \equiv y \pmod{m_i}$

Let φ be the coalition of t users such that the combiner phase will be

$$M_\varphi = \prod_{i \in \varphi} m_i$$

We find the possible sets by using ${}^{n}C_t$

To recover d, find y_{it}

So first we find $N = \prod_{j=1}^{t} m_{ij}$ and $n_i = \frac{N}{m_i}$

Then using modular multiplication inverse $n_i y_{it} \equiv 1 \pmod{m_i}$ we find y_{it}

Then computing
$$y \equiv \sum_{i=1}^{t} n_i y_{it} a_i \pmod{N}$$

We finally compute $d \equiv y \pmod{m_0}$

The Asmuth-Bloom SSS is close to perfect in the sense that $t-1$ or fewer shares do not narrow down the key space: Assume a coalition φ' size $t-1$ has gathered and let y be the unique solution for y in $\mathbb{Z}_M$. According to original scheme, $M/M'_\varphi > m_0$, hence $y + jM'_\varphi$ is smaller than M for $j < m_0$. Since $gcd(m-0, M_\varphi) = 1$, all $(y + jM_\varphi) \pmod{m_0}$ are distinct for $0j < m_0$, and there are m_0 of them. That is, d can be any integer from $\mathbb{Z}_{m_0}$. However, this scheme is not exactly perfect since when $t-1$ shares are known, the key candidates are not equally likely.

Example: 7.3.1.1. *For shares $n = 6$ with recovery threshold $r = 4$, given $p = 83, m_1 = 127, m_2 = 131, m_3 = 137, m_4 = 139, m_5 = 149$ and $m_6 = 151$ where the secret is $x = 75$.*

Solution: Take $A = 3000$ where A is any integer.

Given $x = 75$ and $p = 83$ we have, $y = x + Ap = 249075$

So the shares will be $y_i \equiv y \pmod{m_i}$

$y_1 \equiv 28 \pmod{127}$

$y_2 \equiv 44 \pmod{131}$

$y_3 \equiv 9 \pmod{137}$

$y_4 \equiv 126 \pmod{139}$

$y_5 \equiv 96 \pmod{149}$

$y_6 \equiv 76 \pmod{151}$

To recover the secret, using $r = 4$ and $n = 6$, we can find the sets ${}^6C_4 = 15$

We will take y_1, y_2, y_3 and y_4.

By using $n_i = \frac{N}{m_i}$, where $N = 316818391$ we find the rest of n_i's

So we have $n_1 = 2494633, n_2 = 2418461, n_3 = 2312543$, and $n_4 = 2279269$

Find the shares using $n_i s_i \equiv 1 \pmod{m_i}$ where $i = 1,2,3,4$

$s_1 \equiv 68 \pmod{127}$

$s_2 \equiv 73 \pmod{131}$

$s_3 \equiv 8 \pmod{137}$

$s_4 \equiv 118 \pmod{139}$

$\therefore s \equiv 249075 \pmod{316818391}$

Then $x \equiv 249075 \pmod{83}$

$\implies x \equiv 75 \pmod{83}$, so we have recovered the secret.

But we cannot recover the secret when the recovery threshold is $r - 1$

7.3.2 Modified Asmuth-Bloom SSS

Several changes were needed on the basic Asmuth-Bloom scheme to make it more suitable for function sharing. In this section we describe these modi cations: In the original Asmuth-Bloom SSS, the authors proposed an iterative process to solve the system $y \equiv y_i \pmod{m_i}$. Instead, we use a non-iterative and direct solution as described in , which turns out to be more suitable for function sharing in the sense that it does not require interaction between parties and has an additive structure which is convenient for exponentiations. Suppose φ is a coalition of t users gathered to construct the secret d.

1. Let $M_{\varphi\setminus\{i\}}$ denotes $\prod_{j\in S} m_j$ then $M'_{\varphi,i} M_{\varphi\setminus\{i\}} \equiv 1 \pmod{m_i}$

2. The i^{th} user, $u_i \equiv y_i M'_{\varphi,i} M_{\varphi\setminus\{i\}} \pmod{M_\varphi}$ where $y = d + Am_0$ and $y_i \equiv y \pmod{m_i}$
3. y is computed as
$$y \equiv \sum_{i\in\varphi} u_i \pmod{M_\varphi}$$
4. The secret d is computed as $d \equiv y \pmod{m_0}$

We also modified

$$\prod_{i=1}^{t} m_i > m_0 \prod_{i=1}^{t-1} m_{n-i+1} \quad \text{to} \quad \prod_{i=1}^{t} m_i > m_0^2 \prod_{i=1}^{t-1} m_{n-i+1}$$

Remark: 7.3.2.1. *m_0 need not be a prime and the scheme works correctly for a composite number as long as m_0 is relatively prime to m_i, where $1 \le i \le n$*

Remark: 7.3.2.2. *m_0 need not be known during the secret reconstruction process until the 4^{th} step*

7.4 Function Sharing Based On Asmuth-Bloom SSS

In this section, we present three novel FSSs based on the Asmuth-Bloom SSS for sharing the RSA signature, ElGamal decryption functions and Paillier function. In the original Asmuth-Bloom SSS, the authors proposed a recursive process to solve the system $y \equiv y_i \pmod{m_i}$. Here, we give a direct solution which is more suitable for function sharing. Suppose S is a coalition of t users gathered to construct the secret d.

7.4.1 RSA Function

This RSA function is similar to the original RSA. Except when signing the message we use private key. And to verify the message we use public key.

RSA Set Up

Key Generation: Let $N = pq$ be the product of two large primes p and q.
Choose $e \in \mathbb{Z}^*{}_{\phi(N)}$ and compute $de \equiv 1 \pmod{\phi(N)}$.
The public and private key are (e, N) and d respectively.

Signing: Given a message $w \in \mathbb{Z}_N$, the signature s is computed as

$$s \equiv w^d \pmod N$$

Verification: Given a signature $s \in \mathbb{Z}_N$, the verification is done by checking

$$w \equiv s^e \pmod N$$

Example: 7.4.1.1. *Sign for $p = 59$ and $q = 73$ for the message $w = 87$*

Solution: For $p = 59$ and $q = 73$ we have $N = 4307$ and $\phi(N) = 4176$
We have $e = 19$, compute d
$\implies ed \equiv 1 \pmod{\phi(N)}$
$\implies d \equiv 1099 \pmod{4176}$
To sign the message we use $s \equiv w^d \pmod N \implies s \equiv 87^{1099} \pmod{4307}$
$\implies s \equiv 1254 \pmod{4307}$
Now to verify the authenticity of the message we use $w \equiv s^e \pmod N$
$\implies w \equiv 1254^{19} \pmod{4307}$
$\implies w \equiv 87 \pmod{4307}$ which is our message. So the message has not been altered.

7.4.2 Threshold RSA

Threshold RSA extends the basic RSA scheme to distribute the private key among multiple parties in such a way that a threshold of those parties must collaborate to decrypt a message. This approach enhances security by preventing any single entity from having complete access to the private key.

Threshold RSA enhances security by mitigating the risk of single points of compromise and ensuring that decryption requires collusion among multiple entities. It's useful in scenarios where key management and distribution are challenging or where a higher level of security is desired, similar to Threshold ElGamal.

Threshold RSA Set Up

Key Generation: Choose $p = 2p' + 1$ and $q = 2q' + 1$ where p' and q' are both primes. Let $N = pq$ and the public key e and the private key d are choosen from $\mathbb{Z}^*{}_N$ where $ed \equiv 1 \pmod{\phi(N)}$ Use Asmuth-Bloom SSS for sharing d with $m_0 = 4p'q' = \phi(N)$.

Signing: Let w be the messsage to be signed. Assume a coalition φ of size t wants to obtain the signature $s \equiv w^d \pmod{N}$

Generating Partial Results: Each user $i \in \varphi$. Computes $u_i \equiv y_i M'_{\varphi,i} M_{\varphi\setminus\{i\}} \pmod{M_\varphi}$ where $y = d + Am_0$ and $y_i \equiv y \pmod{m_i}$ such that $s_i \equiv w^{u_i} \pmod{N}$

Combining Partial Results: The incomplete decryptor $\bar{s}$ is obtained by combining the s_i values $\bar{s} \equiv \prod_{i\in\varphi} s_i \pmod{N}$

Correction: Let $K \equiv w^{-M_\varphi} \pmod{N}$ be the corrector. The incomplete signature can be corrected by trying

$$(\bar{s}K^j)^e \equiv \bar{s}^e (K^e)^j \equiv w \pmod{N}, \text{ for } 0 \le j < t \tag{7.1}$$

Then the signature t is computed by $s \equiv \bar{s}K^\delta \pmod{N}$ where δ denotes the value for j that satisfies (7.1)

Verification: Verification is the same as the standard RSA verification i.e. $w \equiv s^e \pmod{N}$

Definition 7.4.2.1. Generating partial results: Generating partial results refers to the process of computing or deriving intermediate outcomes or data points within a larger calculation or operation. These partial results are often generated sequentially or concurrently as part of a multi-step process and may be used to facilitate further computation, analysis, or decision-making.

Definition 7.4.2.2. Combining Partial Results: Combining partial results refers to the process of merging intermediate outcomes or data points generated from different sources or computations into a cohesive final result. This consolidation step is typically necessary in various computational tasks where the final outcome depends on the integration of multiple partial results.

Definition 7.4.2.3. Correction: The use of correction mechanisms after combining partial results is essential for ensuring the accuracy, integrity, and reliability of the combined outcome, particularly in scenarios where data may be subject to errors, inconsistencies, or faults. By detecting and rectifying deviations from the expected outcome, correction mechanisms help produce trustworthy and dependable results that reflect the true nature of the underlying data.

7.4.3 ElGamal Function

ElGamal Function is named after its inventor Taher Elgamal in 1985. The ElGamal Algorithm provides an alternative to the RSA for public key encryption. Security of the RSA depends on the (presumed) difficulty of factoring large integers. While the security of the ElGamal algorithm depends on the (presumed) difficulty of computing discrete logs in a large prime modulus. In other words, it's hard to determine the private key x given the public key y and the generator g. Thus, an adversary who intercepts the ciphertext (C_1, C_2) without knowing the private key should find it computationally infeasible to recover the plaintext message m.

ElGamal encryption is often used in practice, especially in scenarios where secure communication over public channels is needed, such as secure messaging protocols or cryptographic applications. It is a powerful public key encryption scheme that provides perfect forward secrecy. It is slower than others but it is easy to implement and is widely used.

ElGamal Set Up

The Following is the set up for ElGamal Function:

Key generation: Let p be a large prime number and g be a generator of $\mathbb{Z}_p$.
We find $g \equiv h^{(p-1)/q} \pmod{p}$ where h is the hash digest.
Choose a random $\alpha \in \{1, 2, \ldots, p-1\}$ and compute $\beta \equiv g^{\alpha} \pmod{p}$ where (β, g, p) and α are the public and private key respectively.

Encryption: Given a message $w \in \mathbb{Z}_p$, the ciphertext $C = (C_1, C_2)$ is computed as $C_1 \equiv g^r \pmod{p}$ and $C_2 \equiv \beta^r w \pmod{p}$, where r is a random integer from $\mathbb{Z}_p$

Decryption: Given a ciphertext C, the message w is computed as
$w \equiv (C_1^{\alpha})^{-1} C_2 \pmod{p}$.

Example: 7.4.3.1. *For $p = 101, q = 5$ and $h = 41$ for the message $w = 72$.*

Solution: Given $p = 101,\ \ q = 5$ and $h = 41$ we first find the generator
$g \equiv h^{(p-1)/q} \pmod{p}$

$\implies g \equiv h^{100/5} \pmod{101}$

$\implies g \equiv 84 \pmod{101}$

Choose α as any random number as private key. So we have $\alpha = 53$

Then our public key will be $\beta \equiv g^{\alpha} \pmod{p}$

$\implies \beta \equiv 84^{53} \pmod{101}$

$\implies \beta \equiv 36 \pmod{101}$

Now to encrypt the message $w = 72$

We use $C_1 \equiv g^r \pmod{p}$ where r is random number.

$\implies C_1 \equiv 84^{17} \pmod{101}$ where $r = 17$

$\implies C_1 \equiv 87 \pmod{101}$

Now to find C_2 we use $C_2 \equiv \beta^r w \pmod{p}$

$\implies C_2 \equiv 36^{17} \times 72 \pmod{101}$

$\implies C_2 \equiv 89 \pmod{101}$

Now to decrypt the message we compute $w \equiv (C_1^{\alpha})^{-1} C_2 \pmod{p}$

$\implies (87^{53})^{-1} \times 89 \pmod{101}$ Since we cannot solve the equation once we break it down.

First we find the value of $z \equiv (C_1^{\alpha} \pmod{p}$

$\implies z \equiv 87^{53} \pmod{101}$

$\implies z \equiv 84 \pmod{101}$

Now we find the z^{-1}

So $k \equiv z^{-1} \pmod{p}$

$\implies k \equiv 84^{-1} \pmod{101}$

$\implies k \equiv 95 \pmod{1}01$

After mutiplying $95 \times 89 = 8455$

$\implies w \equiv 8455 \pmod{101}$

$\implies w \equiv 72 \pmod{101}$ which is our secret.

7.4.4 Threshold ElGamal

Threshold ElGamal is an extension of the ElGamal encryption scheme that introduces a mechanism for distributing the private key among multiple parties in such a way that a predetermined threshold of those parties must collaborate to decrypt a message. This threshold mechanism enhances security and ensures that no single entity has full access to the decryption key. Similar to the Threshold RSA Function, Threshold ElGamal is also a combination of ElGamal and Asmuth-Bloom SSS. The security of Threshold ElGamal relies on the assumption that no fewer than t parties collude to reconstruct the private key. If fewer than t parties collude, they should gain no information about the private key, ensuring the confidentiality of the encrypted messages.

Threshold ElGamal is particularly useful in scenarios where key management and distribution are challenging or where a higher level of security is desired. It finds applications in secure multi-party computation, distributed systems, and cryptographic protocols where access control and confidentiality are paramount.

Threshold ElGamal Set Up

Key Generation: Choose $p = 2q+1$ where q is a large prime number and let $g \in \mathbb{Z}^*{}_p$ with order q. Choose a random $\alpha \in \{1,\dots,p-1\}$ and compute $\beta \equiv g^\alpha \pmod{p}$.
Let α and (β, g, p) be the private and public key respectively.
Use Asmuth-Bloom SSS for sharing the private key α with $m_0 = 2q$.

Encryption: Given a message $w \in \mathbb{Z}_p$, the ciphertext $C = (C_1, C_2)$ is computed as $C_1 \equiv g^r \pmod{p}$ and $C_2 \equiv \beta^r w \pmod{p}$ where r is a random integer from $\mathbb{Z}_p$

Decryption: Let (C_1, C_2) be ciphertext to be decrypted where $C_1 \equiv g^k \pmod{p}$ for some $k \in \{1,\dots,p-1\}$ and $C_2 = \beta^k w$, where w is the message. The coalition φ of t

users wants to obtain the message $w \equiv sC_2 \pmod{p}$ for the decryptors
$s \equiv (C_1^{\alpha})^{-1} \pmod{p}$.

Generating Partial Results: Each user $i \in \varphi$ computes
$u_i \equiv y_i M'_{\varphi,i} M_{\varphi \setminus \{i\}} \pmod{M_\varphi}$ where $y = d + Am_0$ and $y_i \equiv y \pmod{m_i}$.
$s_i \equiv C_1^{-u_i} \pmod{p}$ and $\beta \equiv g^{u_i} \pmod{p}$

Combining Partial Results: The incomplete decryptor $\bar{s}$ is obtained by combining the s_i values $\bar{s} \equiv \prod_{i \in \varphi} s_i \pmod{p}$

Correction: The β_i values will be used to find the exponent which will be used to correct the incomplete decryptor.
Compute the incomplete public key $\overline{\beta}$ as $\overline{\beta} \equiv \prod_{i \in \varphi} \beta_i \pmod{p}$.
Let $K_s \equiv C_1^{M_\varphi} \pmod{p}$ and $K_\beta \equiv g^{-M_\varphi} \pmod{p}$ be the correctors for s and β respectively. The corrector exponent δ can be obtained by trying

$$\overline{\beta} K_\beta^j \equiv \beta \pmod{p} \text{ for } 0 \leq j \leq t \tag{7.2}$$

Extracting the Message: Compute the message w as $s \equiv \bar{s} K_s^{\delta} \pmod{p}$ and
$w \equiv sC_2 \pmod{p}$ where δ denotes the value for j that satisfies (7.2)

7.4.5 Paillier Function

The Paillier cryptosystem is a public-key encryption scheme with homomorphic properties, meaning it supports operations on encrypted data without needing to decrypt it first. It was proposed by Pascal Paillier in 1999 and is based on the computational difficulty of the decisional composite residuosity assumption.

The security of the Paillier cryptosystem relies on the difficulty of the decisional composite residuosity assumption, which states that given n and n^2 it's computationally hard to distinguish between g^x mod n^2 and a random value in the set $\mathbb{Z}^*{}_{n^2}$.

It allows two types of computation: Addition of two ciphertext and Multiplication of two ciphertext by a plaintext number.

Paillier Set Up

Key Generation: Choose two large prime numbers p and q randomly and independently of each other such that $gcd(N,(p-1)(q-1)) = 1$ This property is assured if both primes are of equal length. Compute $N = pq$ and $\lambda = lcm(p-1, q-1)$.
Choose a random $g \in \mathbb{Z}_{N^2}$ such that the order of g is a multiple of N. Ensure n divides the order of g by checking the existence of the following modular multiplicative inverse

$$\mu \equiv L(g^\lambda \pmod{N^2})^{-1} \pmod{N}$$

where L is a function s.t. $L(x) = \frac{x-1}{N}$ and the result of function L is always an integer. (N,g) and (λ,μ) are the public and private key respectively.

Encryption: Let w be the message to be encrypted where $0 \leq w \leq N$. Select a random r where $gcd(r,N) = 1$, the ciphertext C is computed as $C \equiv g^w r^N \pmod{N^2}$.

Decryption: Given a ciphertext $C \in \mathbb{Z}_{N^2}$, the message w is computed as $w \equiv \frac{L(C^\lambda \pmod{N^2})}{L(g^\lambda \pmod{N^2})} \pmod{N}$ which can also be written as $w \equiv L(C^\lambda \pmod{N^2})\mu \pmod{N}$.

Example: 7.4.5.1. *Encrypt the message $w = 42$ using $p = 7$ and $q = 11$*

Solution: Let $p = 7$ and $q = 11$ be two prime integers such that $N = pq = 7 \times 11 = 77$ since $gcd(77,60) = 1$.

Now we select a random $g = 5652$ since it satisfies the property that order of g is $2310 = 30 \times 77$ in $\mathbb{Z}_{N^2}$.

Thus the public key will be $(N, g) = (77, 5652)$

Now we encrypt the message $w = 42$ and we choose a random number r as 23 using $C \equiv g^m r^N \pmod{N^2}$.

$\implies C \equiv 5652^{42} 23^{77} \pmod{5929}$

$\implies C \equiv 4626 \pmod{5929}$

Now we find μ,

$$\begin{aligned}
\mu &\equiv L(g^{\lambda} \pmod{N^2})^{-1} \pmod{N} \\
&\equiv L(5652^{30} \pmod{5929})^{-1} \pmod{77} \\
&\equiv L(3928)^{-1} \pmod{77} \\
&\equiv \left(\frac{3928-1}{77}\right)^{-1} \pmod{77} \\
&\equiv 51^{-1} \pmod{77} \\
&\equiv 74 \pmod{77}
\end{aligned}$$

Now we decrypt the message using $w \equiv L(C^{\lambda} \pmod{N^2})\mu \pmod{N}$

$\implies w \equiv L(4626^{30} \pmod{5929})74 \pmod{77}$

$\implies w \equiv L(4852)74 \pmod{77}$

$\implies w \equiv 63 \times 74 \pmod{77}$

$\implies w \equiv 4662 \pmod{77}$

$\implies w \equiv 42 \pmod{77}$ which is our message.

7.4.6 Threshold Paillier

Threshold Paillier is an extension of the Paillier cryptosystem that introduces a mechanism for distributing the private key among multiple parties in such a way that a predetermined threshold of those parties must collaborate to decrypt a message. This threshold mechanism enhances security and ensures that no single entity has full access to the decryption key.

The security of Threshold Paillier relies on the assumption that no fewer than t parties collude to reconstruct the private key. If fewer than t parties collude, they should gain no information about the private key, ensuring the confidentiality of the encrypted messages.

Threshold Paillier is particularly useful in scenarios where key management and distribution are challenging or where a higher level of security is desired. It finds applications in secure multi-party computation, distributed systems, and cryptographic protocols where access control and confidentiality are paramount.

Threshold Paillier Set Up

Definition 7.4.6.1. Carmichael Number: A composite integer n is a Carmichael number if and only if $an \equiv a \pmod{n}$ for all $a \in \mathbb{Z}$

Key Generation: Choose $p = 2p' + 1$ and $q = 2q' + 1$ where p' and q' are also primes and $gcd(N, \phi(N)) = 1$ for $N = pq$. Let $g \equiv (1+N)^a b^N \pmod{N^2}$ for a random $a, b \in \mathbb{Z}^*{}_N$.
Compute $\theta \equiv a\beta \pmod{N}$ for $\beta \in \mathbb{Z}^*{}_N$ and where $\lambda = lcm(p-1, q-1)$ is the Carmichael number for N. Let (N, g, θ) and λ be the public and private key respectively.

Use Asmuth-Bloom SSS to share $\beta\lambda$ with $m_0 = N\lambda$.

Encryption: Given a message $w \in \mathbb{Z}_N$, the ciphertext C is computed as $C \equiv g^w r^N \pmod{N^2}$.

Decryption: Let $C \equiv g^w r^N \pmod{N^2}$ be the ciphertext to be decrypted for some random $r \in \mathbb{Z}^*{}_N$, where w is the message from $\mathbb{Z}_N$.
Assume a coalition φ of size t wants to obtain the message $w \equiv \frac{L(C^{\beta\lambda} \pmod{N^2})}{\theta} \pmod{N}$. We call $s \equiv C^{\beta\lambda} \pmod{N^2}$ as the decryptor.

Generating Partial Results: Each user $i \in \varphi$ computes$u_i \equiv y_i M'_{\varphi,i} M_{\varphi\setminus\{i\}} \pmod{M_\varphi}$ where $y = d + Am_0$ and $y_i \equiv y \pmod{m_i}$ then $s_i \equiv C_i^{u_i} \pmod{N^2}$ and $\theta_i \equiv g^{u_i} \pmod{N^2}$

Combining Partial Results The incomplete decryptor $\bar{s}$ is obtained by combining the s_i values $\bar{s} \equiv \prod_{i\in\varphi} s_i \pmod{N^2}$

Correction: The θ_i values will be used to find the exponent which corrects the incomplete decryptor. Compute $\theta \equiv \prod_{i\in\varphi} \theta_i \pmod{N^2}$. Let $K_s \equiv C^{-M_\varphi} \pmod{N^2}$ and $K_\theta \equiv g^{-M_\varphi} \pmod{N^2}$ be the correctors for s and θ respectively. The corrector exponent δ can be obtained by trying

$$\theta \equiv L(\overline{\theta} K_\theta^j \pmod{N^2}) \text{ for } 0 \le j < t \tag{7.3}$$

Extracting the Message: Compute the message w as $s \equiv \bar{s} K_s^\delta \pmod{N^2}$ and $w \equiv \frac{L(s)}{\theta} \pmod{N}$ where δ denotes the value for j that satisfies (7.3).

The decryptor $\bar{s}$ is incomplete and to find the corrector exponent we used a similar approach. When the equality holds we know that $\theta \equiv a\beta\lambda \pmod{N^2}$ is the correct value. Also, this equality must hold for one j value, denoted by δ, in the given interval. Actually our purpose is not computing θ since it is already known. We want to find the corrector exponent δ to obtain $\bar{s}$, which is also equal to the one we used to obtain θ.

Bibliography

[1] Dr Michael Evans AMSI. “RSA Encryption”. In: (). URL: https://amsi.org.au/teacher_modules/pdfs/Maths_delivers/Encryption5.pdf.

[2] C. Asmuth and J. Bloom. “A modular approach to key safeguarding”. In: *IEEE Transactions on Information Theory* 29.2 (1983), pp. 208–210.

[3] David M. Burton. “Elementary Number Theory, Seventh Edition”. In: *Sigma Mathematics* (), pp. 79–82.

[4] Carol S. Jackson. “The Chinese Remainder Theorem”. In: *Mathematics Capstone Project* (Dec. 2021).

[5] Baivab Kumar Jena. “Digital Signature Algorithm (DSA) in Cryptography: How It Works More”. In: (). URL: https://www.simplilearn.com/tutorials/cryptography-tutorial/digital-signature-algorithm.

[6] Kamer Kaya. “Threshold Cryptography with Chinese Remainder Theorem”. In: *Sigma Mathematics* (Aug. 2009).

[7] Kamer Kaya and Ali Aydin Selçuk. “Threshold Cryptography based on Asmuth-Bloom Secret Sharing”. In: *Information Sciences* (Apr. 2006-2007).

[8] Saurabh Singh and Gaurav agarwal. “Use of CRT to generate Random numbers for Cryptography”. In: *International Journal of Applied Engineering* 2 (2010).

[9] Yufei Tao. “RSA Cryptosystem”. In: (). URL: https://www.cse.cuhk.edu.hk/taoyf/course/bmeg3120/notes/rsa.pdf.