Algorithmic Number Theory and Cryptography

Abhay Kasera

A dissertation submitted for the partial fulfillment of BS-MS dual degree in Mathematics

DEPARTMENT OF MATHEMATICS
INDIAN INSTITUTE OF SCIENCE EDUCATION AND RESEARCH MOHALI, PUNJAB−140306

Certificate of Examination

This is to certify that the dissertation titled **"Algorithmic Number Theory and Cryptography"** submitted by **Mr. Abhay Kasera** (Reg. No. MS13087) for the partial fulfillment of BS-MS dual degree programme of the Institute, has been examined by the thesis committee duly appointed by the Institute. The committee finds the work done by the candidate satisfactory and recommends that the report be accepted.

Dr. Amit Kulshrestha Dr. Chandrakant S. Aribam Prof. Kapil H. Paranjape

(Thesis Supervisor)

Dated: 20/04/2018

Declaration

The work presented in this dissertation has been carried out by me under the guidance of Prof. Kapil Hari Paranjape at the Indian Institute of Science Education and Research Mohali.

This work has not been submitted in part or in full for a degree, a diploma, or a fellowship to any other university or institute. Whenever contributions of others are involved, every effort is made to indicate this clearly, with due acknowledgement of collaborative research and discussions. This thesis is a bonafide record of work done by me and all sources listed within have been detailed in the bibliography.

Abhay Kasera
(Candidate)

Dated: 20/04/2018

In my capacity as the supervisor of the candidate's project work, I certify that the above statements by the candidate are true to the best of my knowledge.

Prof. Kapil H. Paranjape
(Thesis Supervisor)

Acknowledgement

I want to thank Prof. Kapil Hari Paranjape for his guidance, support and helping me to explore a wide range of topics in Algorithmic number theory and Cryptography. I deeply believe that it is his guidance that made my MS project very fascinating, under him I had the freedom to work on whatever I feel interesting, and his vast knowledge helped me to get a deeper understanding of the concepts. I want to thank my committee members Dr. Amit Kulshrestha and Dr. Chandrakant S. Aribam. I want to thank my family for believing in me and supporting every decision which I have taken in my life. I also want to thank Department of Science and Technology (DST), Government of India for DST-INSPIRE fellowship.

Abhay Kasera

Abstract

Primality testing and Integer factorization problem are two widely studied problems in Algorithmic Number Theory. Can we factorize integers in polynomial time is still an unsolved question. However, the Primality testing problem can be solved in polynomial time. RSA is the most widely used Public-key cryptosystem whose security is based on Integer factorization problem. Over past years researchers have studied various attacks on RSA cryptosystem, and it has been concluded that these attacks can be avoided if RSA is implemented securely. In this Thesis we have covered Primality testing algorithms, Factoring algorithms and Cryptanalysis of RSA.

List of symbols

$\emptyset$: null set
$\mathbb{N}$: set of natural numbers
$\mathbb{Z}$: set of integers
$\mathbb{Z}^+$: set of positive integers
$\mathbb{Z}_n$: set of integers modulo n
$\mathbb{Z}_n^*$: set of integers modulo n which are relatively prime to n
$a \mid b$: a divides b
$a \nmid b$: a does not divide b
$\mid X \mid$: cardinality of the set X
$x \equiv y \mod n$: x is congruent to y modulo n
$x \not\equiv y \mod n$: x is not congruent to y modulo n
$\Longrightarrow$: implication
$\lfloor a \rfloor$: floor function; the greatest integer less than or equal to x
$\left(\frac{a}{p}\right)$: Legendre symbol
$\left(\frac{a}{n}\right)$: Jacobi symbol
$\phi(n)$: Euler-totient function defined as number of positive integers less than or equal to n which are relatively prime to n

Contents

Chapter 1

Introduction

Algorithmic number theory is a branch of number theory which involves the study of algorithms to solve problems in various branches of number theory. Primality testing problem and integer factorization problem are two widely studied problems in Algorithmic number theory.

Primality testing problem: Given a positive integer $n > 1$, determine whether it is prime or not. In Chapter 2, we have discussed Primality tests. There are two types of primality tests,

- Probabilistic primality tests: These tests are absolutely correct when they output n to be composite, but they are only probably correct when they output n to be prime. for example: Fermat's test, Solovay-Strassen test, and Miller-Rabin test.

- True primality tests: These tests provide the mathematical proof if a number is prime but are generally more computationally intensive. for example: Lucas test and Pocklington's test.

The Fermat's test is based on Fermat's little theorem, but it has a problem that it cannot distinguish between prime numbers and Carmichael numbers. Solovay-Strassen test is based on Euler's criterion and involves computation of Jacobi symbol hence it is more computationally intensive. Miller-Rabin is the most widely used probabilistic primality test; it is never worse than Solovay-Strassen test and Fermat's test.
True primality tests for an integer n requires partial or complete factorization of $n-1$. Lucas test is based on the fact that n is prime if and only if $\mathbb{Z}_n^*$ has an element of order $n-1$. This test requires the knowledge of factorization of $n-1$. Pocklington's test only requires partial factorization of $n-1$, and it is based on the Pocklington's theorem.

Integer factorization problem: Given a positive, composite integer n, find its prime factorization i.e., $n = p_1^{e_1} p_2^{e_2} \cdots p_t^{e_t}$ where the p_i are distinct primes and each $e_i \geq 1$. To solve integer factorization problem it is enough to study algorithms that find a non-trivial factorization of n i.e., $n = ab$ where $1 < a < n$ and $1 < b < n$. In Chapter 3, we have discussed factoring algorithms such as Trial division, Pollard's rho method, Pollard's $p-1$ method, Lenstra's Elliptic-curve factorization and Factorization using group $G(n, N)$. Pollard's rho algorithm uses Floyd's cycle-finding algorithm to find small factors of composite integer n. R. P. Brent has proposed a faster version of Pollard's rho method. Pollard's $p-1$ method finds a non-trivial factor of n, when n has a prime factor p such that $p-1$ has no large prime factors. Lenstra's Elliptic curve factorization method is a generalization of Pollard's $p-1$ method. In this method, the group $\mathbb{Z}_p^*$(p is a prime divisor of n) used in Pollard's $p-1$ method is replaced by Elliptic curve group over $\mathbb{Z}_p$. In Pollard's $p-1$ method, we are focusing our hopes on groups $\mathbb{Z}_p^*$ where p runs over prime divisors of n. for a given n these groups are fixed so we are inconclusive when all of these groups has order divisible by a large prime. The above problem encountered with Pollard's $p-1$ method is solved if we work with elliptic curves over $\mathbb{Z}_p$ because elliptic curves over finite fields provide a large number of finite abelian groups and provide more flexibility in choosing an elliptic curve. So we can expect one group to have order not divisible by a large prime(or prime power). At the end of the chapter, we have proposed a factoring algorithm based on the group $G(n, N)$.

RSA is a public-key cryptosystem invented by Ron Rivest, Adi Shamir and Leonard Adleman. The security of RSA is based on integer factorization problem. Up to now, there does not exist any classical polynomial-time factoring algorithm. Over the past years, RSA has been subjected to various attacks, but none of these attacks are a threat to RSA if RSA is implemented adequately. In Chapter 4, we have examined the security of RSA by studying various attacks. We have discussed elementary attacks such as common modulus attack, forward search attack, adaptive-chosen ciphertext attack. Also, we have studied low public exponent attacks due to Don Coppersmith and low private exponent attack due to M. Wiener.

Chapter 2

Primality Testing

2.1 Introduction

Public-key Cryptosystems require an efficient generation of public-key parameters. For example, In RSA public-key cryptosystem with RSA-modulus $n = pq$, the primes p and q must be of sufficient size, must be chosen randomly and have certain additional properties to avoid attacks on the cryptosystem. One of the most fundamental requirements in Public-key cryptography is to generate large prime numbers. The most natural method is to generate sufficiently large random integer n, and test it for primality. There are two types of primality tests; True primality test, and Probabilistic primality test. Former proves that the candidate is prime whereas, latter is correct when it declares n to be composite but does not provide a mathematical proof for primality of n and establishes a weaker result such as that the candidate is "probable prime". Hence, Probabilistic primality tests are also called *Compositeness tests*.

2.2 Probabilistic primality tests

These tests have following structure. let n be an odd integer and a set $W(n) \subset \mathbb{Z}_n$ defined such that:

1. given $a \in \mathbb{Z}_n$, it can be checked in deterministic polynomial time whether $a \in W(n)$
2. $W(n) = \emptyset$, if n is prime.
3. $\mid W(n) \mid \geq n/2$, if n is composite.

let P, C denote the set of all the primes and composites respectively. Now the probability,

$$P(a \in W(n) \mid \ n \in C) \geq 1/2 \ and,$$

$$P(a \notin W(n) \mid \ n \in C) < 1/2$$

by using Bayes's theorem,

$$P(n \in C \mid a \in W(n)) = \frac{P(a \in W(n) \mid n \in C)P(n \in C)}{P(a \in W(n) \mid n \in C)P(n \in C) + \ P(a \in W(n) \mid n \in P)P(n \in P)}$$

from 2. we have $P(a \in W(n) \mid n \in P) = 0$ thus,

$$P(n \in C \mid a \in W(n)) = 1$$

and,

$$P(n \in P \mid a \notin W(n)) = \frac{P(a \notin W(n) \mid n \in P)P(n \in P)}{P(a \notin W(n) \mid n \in P)P(n \in P) + \ P(a \notin W(n) \mid n \in C)P(n \in C)}$$

from 2. and 3. $P(a \notin W(n) \mid n \in P) = 1$, $P(a \notin W(n) \mid \ n \in C) < 1/2$. thus,

$$P(n \in P \mid a \notin W(n)) < 1$$

Suppose n is an integer whose primality is to be tested. An integer $a \in \mathbb{Z}_n$ is chosen at random and checked if $a \in W(n)$. If $a \in W(n)$ then test outputs 'composite', and it is sure that n is composite. If $a \notin W(n)$ then test outputs 'prime' and n is said to pass primality test for base a, but we can't claim n to be prime with absolute certainty. However, successive independent runs of the test all of which returning answer 'prime' allow the confidence that n is prime and confidence can be increased to any desired level.

Definition 2.2.1. [10] If n is composite, the elements of $W(n)$ are called *witnesses* to the compositeness of n, and elements of the set $\mathbb{Z}_n - W(n)$ are called *liars*.
An integer concluded to be prime on the basis of a probabilistic primality test is called *probable prime*.

2.2.1 Fermat's Test

Theorem 2.2.1. (Fermat's little Theorem) If p is a prime and a be any integer not divisible by p then, $a^{p-1} \equiv 1 \mod p$.

Let n be any integer whose primality is to be determined. By contrapositive of above theorem, if the above condition fails for any integer a, $1 \leq a \leq n-1$ then n is composite. otherwise, finding an integer a, $1 \leq a \leq n-1$ such that $a^{n-1} \equiv 1 \mod n$ then n appears prime for the base a.

Definition 2.2.2. Let n be an odd composite integer. An integer a, $1 \leq a \leq n-1$ such that $a^{n-1} \not\equiv 1 \mod n$ is called a *Fermat witness* for n, and an integer a, $1 \leq a \leq n-1$ such that $a^{n-1} \equiv 1 \mod n$ then n is said to be *pseudoprime* to the base a, integer a is called *Fermat liar* for n. The set of witnesses is $W(n) = \{1 \leq a \leq n-1 \mid a^{n-1} \not\equiv 1 \mod n\}$.

Algorithm: Fermat's Test

Input: an odd integer $n \geq 3$ and parameter t
Output: answer n is prime or composite.

1. for i from 1 to t do:
   1.1. choose a random integer a, $2 \leq a \leq n-2$.
   1.2. using repeated-square modular exponentiation algorithm, compute $x = a^{n-1} \mod n$.
   1.3. If $x \neq 1$ then return 'composite'
2. return 'prime'.

Definition 2.2.3. A composite integer n for which $a^{n-1} \equiv 1 \mod n$ holds for every positive integer a satisfying gcd(a, n)=1 is called a *Carmichael number*.

example: 561, 1105, 1729, 2465, 2821 are first five Carmichael numbers.

Remark:[13] In 1912, Robert Carmichael conjectured that there are infinitely many Carmichael numbers. W. Alford, G. Granville, and C. Pomerance proved this conjecture in 1992.

If n is a Carmichael number then the integers a, $1 \leq a \leq n-1$ such that gcd(a, n)>1 are the only Fermat witnesses for n. In the case when all prime factors of n are large, Fermat's test declares n to be prime, even at a large number of iterations. Hence, Fermat's test is not a true probabilistic primality test as it cannot distinguish between prime numbers and Carmichael numbers.

2.2.2 Solovay-Strassen Test

At first we will introduce the notion of Legendre and Jacobi symbols.

Definition 2.2.4. let $a \in \mathbb{Z}_n^*$, if there exist an $x \in \mathbb{Z}_n^*$ satisfying $x^2 \equiv a \mod n$ then a is called a *Quadratic residue* modulo n. we denote the set of all quadratic residues modulo n by Q_n and the set of all quadratic non-residues by Q_n'

Definition 2.2.5. Let x be an integer and p be an odd prime then Legendre symbol $\left(\frac{a}{p}\right)$ is defined as,

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & ,if\ a \in Q_p \\ -1 & ,if\ a \in Q'_p \\ 0 & ,if\ p \mid a \end{cases}$$

Definition 2.2.6. let $n = p_1^{e_1} p_2^{e_2} \cdots p_t^{e_t}$ then the Jacobi symbol $\left(\frac{a}{n}\right)$ is defined as,

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1} \left(\frac{a}{p_2}\right)^{e_2} \cdots \left(\frac{a}{p_t}\right)^{e_t}$$

where each of the $\left(\frac{a}{p_i}\right)$ are Legendre symbols.

If n is prime then the Jacobi symbol is Legendre symbol.

Some properties of Jacobi symbol

let $n, m \in \mathbb{Z}^+$ and $a, b \in \mathbb{Z}$

1. $a \equiv b \mod n \implies \left(\frac{a}{n}\right) = \left(\frac{b}{n}\right)$
2. $\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right)$
3. $\left(\frac{-1}{n}\right) = (-1)^{(n-1)/2}$
4. $\left(\frac{2}{n}\right) = (-1)^{(n^2-1)/8}$
5. $\left(\frac{n}{m}\right) = (-1)^{(m-1)(n-1)/4}\left(\frac{m}{n}\right)$

Now we will describe a primality test based on following theorems.

Theorem 2.2.2. (Euler's Criterion) Let n be an odd prime, then for any integer a satisfying gcd(a, n)=1,

$$a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \mod n$$

The proof of Theorem 2.2.3 and Theorem 2.2.4 follows the approach given in [7].

Theorem 2.2.3. (Solovay-Strassen) Let n be an odd composite integer then there exist an integer a, such that gcd(a, n)=1 and

$$a^{(n-1)/2} \not\equiv \left(\frac{a}{n}\right) \mod n$$

Proof. suppose n is an odd and composite integer,
case 1: suppose n is squarefree, $n = p_1p_2\cdots p_t$ where $t \geq 2$ and p_i are distinct odd primes. we know that the proportion of non-square numbers modulo p_1 is exactly half, so there exist an integer x such that $\left(\frac{x}{p_1}\right) = -1$. By chinese remainder theorem there exist an integer a such that,

$$a \equiv x \mod p_1,\ a \equiv 1 \mod p_2\cdots p_t$$

then $\gcd(a, p_1) = 1$, $\gcd(a, p_1p_2\cdots p_t) = 1 \implies \gcd(a, n) = 1$
and, $\left(\frac{a}{p_1}\right) = \left(\frac{x}{p_1}\right) = -1$, for $i > 1$ $\left(\frac{a}{p_i}\right) = \left(\frac{1}{p_i}\right) = 1$ which implies,

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)\cdots\left(\frac{1}{p_t}\right) = -1$$

To contrary assume $a^{n-1/2} \equiv \left(\frac{a}{n}\right) \mod n$ then $a^{n-1/2} \equiv -1 \mod n$. reducing this congruence modulo p_2 we get $1 \equiv -1 \mod p_2$, which is a contradiction.

case 2: let n has repeated prime factor p, $n = p^km$ where $\gcd(p, m) = 1$ and $k \geq 2$. By Chinese remainder theorem there exist an integer a such that,

$$a \equiv 1 \mod m \textit{ and } , a \equiv 1 + p \mod p^2$$

therefore $p \nmid a$, $\gcd(a, m) = 1 \implies \gcd(a, n) = 1$.
To contrary assume $a^{n-1/2} \equiv \left(\frac{a}{n}\right) \mod n$ *then* $a^{(n-1)} \equiv 1 \mod n$. Now, reducing the congruence relation modulo p^2 we get $a^{(n-1)} \equiv 1 \mod p^2$. Since $a \equiv 1 + p \mod p^2$ we obtain $(1 + p)^{n-1} \equiv 1 \mod p^2$.
By binomial theorem, $(1 + p)^{n-1} \equiv 1 + (n - 1)p \mod p^2 \implies$ 1+(n-1)p $\equiv 1 \mod p^2 \implies (n - 1)p \equiv 0 \mod p^2 \implies n - 1 \equiv 0 \mod p$. which leads to a contradiction since n is a multiple of p.

□

Definition 2.2.7. let n be an odd composite integer and a be an integer, $1 \leq a \leq n - 1$. If either $a^{n-1/2} \not\equiv \left(\frac{a}{n}\right) \mod n$ or $\gcd(a, n) > 1$, then a is called an Euler witness for n. otherwise, if $a^{n-1/2} \equiv \left(\frac{a}{n}\right) \mod n$ and $\gcd(a, n) = 1$, then n is said to be Euler pseudoprime to the base a and integer a is called an Euler liar for n. The set of witnesses $W(n) = \{1 \leq a \leq n - 1 \mid a^{n-1/2} \not\equiv \left(\frac{a}{n}\right) \mod n \textit{ or } \gcd(a, n)>1\}$.

Theorem 2.2.4. Let $n > 1$ be an odd integer and,

$$\text{X} = \{\ 1 \leq a \leq n - 1 :\ \gcd(a, n) = 1,\ a^{n-1/2} \equiv \left(\frac{a}{n}\right) \mod n\},$$
$$\text{Y} = \{\ 1 \leq a \leq n - 1 :\ \gcd(a, n) = 1,\ a^{n-1/2} \not\equiv \left(\frac{a}{n}\right) \mod n\},$$
$$\text{Z} = \{\ 1 \leq a \leq n - 1 :\ \gcd(a, n) > 1\}$$

(i) If n is prime then $\mid X \mid = n-1$,
(ii) If n is composite then $\mid X \mid < (n-1)/2$.

Proof. proof of (i) follows from Theorem 2.2.2 Euler's criterion.
Now, we will prove (ii), we know that if $\gcd(a,n) = 1$ then $\left(\frac{a}{n}\right) = \pm 1$ and if $\gcd(a,n) > 1$ then $\left(\frac{a}{n}\right) = 0$. since $1 \in X$, $X \neq \emptyset$, from Theorem 2.2.3 $Y \neq \emptyset$ and since n is composite $Z \neq \emptyset$. Also, cardinality of each of them is between 1 to $n-1$.

choose any integer $y \in Y$,
claim: $Xy = \{xy \mod n : x \in X\} \subset Y$.
for any $x \in X$, $\gcd(xy,n) = 1$ and,

$$(xy)^{(n-1)/2} \equiv x^{(n-1)/2} y^{(n-1)/2} \equiv \left(\frac{x}{n}\right) y^{(n-1)/2} \mod n$$

Now, $xy \mod n$ is either in X or Y. let $xy \mod n \in X$ then $(xy)^{(n-1)/2} \equiv \left(\frac{xy}{n}\right) = \left(\frac{x}{n}\right)\left(\frac{y}{n}\right) \mod n$, so $\left(\frac{x}{n}\right)\left(\frac{y}{n}\right) = \left(\frac{x}{n}\right) y^{(n-1)/2} \mod n$. since $\gcd(x,n) = 1$, $\left(\frac{x}{n}\right) = \pm 1$ hence, $\left(\frac{y}{n}\right) = y^{(n-1)/2} \mod n$. This contradicts that $y \in Y$, hence for any $x \in X$, $xy \mod n \in Y$, i.e, $Xy \subset Y$.
let x_1 and $x_2 \in X$, if $x_1 y \equiv x_2 y \mod n$ then $x_1 \equiv x_2 \mod n \implies x_1 = x_2$. hence, $\mid Xy \mid = \mid X \mid$. we have, $Xy \subset Y$, $\mid X \mid = \mid Xy \mid \leq \mid Y \mid$ so,

$$2 \mid X \mid < \mid X \mid + \mid X \mid + 1 \leq \mid X \mid + \mid Y \mid + \mid Z \mid = n-1$$

i.e., $\mid X \mid < (n-1)/2$ □

Theorem 2.2.5. Let $n > 1$ be an odd integer. If n is prime then the fraction of integers between 1 and $n-1$ which are Euler witnesses is 0 and If n is composite then more than $1/2$.

Proof. If n is prime then the result follows from Theorem 2.2.2. If n is composite then result follows from Theorem 2.2.4(ii).

□

Algorithm: Solovay-Strassen test

Input: an odd integer $n \geq 3$ and parameter t
Output: answer n is prime or composite.

1. for i from 1 to t do:

    1.1. At random, choose an integer a, $1 \leq a \leq n-2$

1.2. Using repeated-square algorithm for modular exponentiation compute $x = a^{(n-1)/2} \mod n$.

1.3. return '*composite*', if $x \neq 1$ and $x \neq n-1$.

1.4. compute $s = \left(\frac{a}{n}\right)$.

1.5. return '*composite*', If $x \equiv s$.

2. return '*prime*'.

The problem with Solovay-Strassen test is that it requires computation of Jacobi symbol. Now, we will discuss Miller-Rabin probabilistic primality test. It is easy to implement, it has less error probability and never worse than Solovay-Strassen test.

2.2.3 Miller-Rabin test

Miller-Rabin test is the most used probabilistic primality test in practice. It is also known as the strong pseudoprime test.

Theorem 2.2.6. [13] Let p be a prime. Then $x^2 \equiv 1 \mod p$ if and only if $x \equiv \pm 1 \mod p$.

Proof. we have,

$$\begin{aligned} \mathrm{x}^2 \equiv 1 \mod p &\iff (x+1)(x-1) \equiv 0 \mod p \\ &\iff \mathrm{p}|(\mathrm{x}+1)(\mathrm{x}\text{-}1) \\ &\iff p \mid (x+1) \text{ or } p \mid (x-1) \\ &\iff x+1 \equiv 0 \mod p \text{ or } x-1 \equiv 0 \mod p \\ &\iff x \equiv -1 \mod p \text{ or } x \equiv 1 \mod p. \end{aligned}$$

Conversely, if either $x \equiv -1 \mod p$ or $x \equiv 1 \mod p$ then, $x^2 \equiv 1 \mod p$.

□

Definition 2.2.8. The integer x such that $x^2 \equiv 1 \mod n$ but $x \not\equiv \pm 1 \mod n$, is called a *non-trivial square root* of 1 modulo n.

example: $x = 6$ is a nontrivial square root of 1 modulo 35. since, $x^2 = 6^2 \equiv 1 (\mathrm{mod}\ 35)$, $6 \not\equiv \pm 1 (\mathrm{mod}\ 35)$

Corollary 2.2.6.1. If there exist a non-trivial square root of 1 modulo n, then n is composite.

Theorem 2.2.7. Let n be an odd prime, and let $n-1 = 2^e k$ where k is odd. let a be any integer such that gcd(a, n)=1. then, either $a^k \equiv 1 \mod n$ or $a^{2^j k} \equiv -1 \mod n$ for some j, $0 \leq j \leq e-1$.

Proof. let n be an odd prime and an integer a relatively prime to n, $1 < a < n$. Consider the sequence

$$\{a^k, a^{2k}, a^{4k}, \cdots, a^{2^{e-1}k}, a^{2^e k}\} \mod n$$

due to Fermat's Little theorem the last entry should be 1 and due to Theorem 2.2.6, above sequence has one of the two following forms,

$$\{1, 1, \cdots, 1, 1, 1, \cdots, 1\} \mod n$$
$$\text{or } \{*, *, \cdots, *, -1, 1, \cdots, 1\} \mod n$$

where $*$ denotes a number different from ± 1. This concludes the proof.

□

Algorithm: Miller-Rabin Test

Input: an odd integer $n \geq 3$ and parameter t
Output: answer n is prime or composite.

1. write $n-1 = 2^e k$ where k is odd.
2. for i from 1 to t do:
   2.1. choose a random integer a, $2 \leq a \leq n-2$.
   2.2. using repeated-square modular exponentiation algorithm, compute $x = a^k \mod n$.
   2.3. If $x \neq 1$ and $x \neq n-1$ do:
      2.3.1. j=1

2.3.2. while $j \leq e-1$ and $x \neq n-1$ do:
compute $x = x^2 \bmod n$
If $x = 1$ then, return 'composite'.
$j = j + 1$
If $y \neq n-1$ then return 'composite'.

3. return 'prime'.

Definition 2.2.9. Let n be an odd composite integer and let $n-1 = 2^e k$ where k is odd. let a be an integer, $1 \leq a \leq n-1$. If $a^k \not\equiv 1 \bmod n$ and if $a^{2^j k} \not\equiv -1 \bmod n$ for all j, $0 \leq j \leq e-1$, then a is called a *strong witness* for n. Otherwise, if either $a^k \equiv 1 \bmod n$ or $a^{2^j k} \equiv -1 \bmod n$ for some $0 \leq j \leq e-1$ then, the integer a is called a strong liar for n and n is said to be a *strong pseudoprime* to the base a. The set of witnesses,
$W(n) = \{1 \leq a \leq n-1 \mid a^k \not\equiv 1 \bmod n \text{ and } a^{2^j k} \not\equiv -1 \bmod n \text{ for all } j, 0 \leq j \leq e-1\}$.

The proof of Theorem 2.2.11 uses Theorem 2.2.8, Theorem 2.2.9 and Theorem 2.2.10. To prove these theorems we follow approach given in [6].

Theorem 2.2.8. Let $n = p^\alpha$ be a prime power for prime p and $\alpha \geq 1$ then the Miller-Rabin liars for n are the solutions to $a^p - 1 \equiv 1 \mod p^\alpha$ which forms a group under the group operation multiplication modulo n.

Proof. let a, $1 \leq a \leq n-1$ be a Miller-Rabin liar then by Euler's theorem $a^{\phi(n)} \equiv 1 \mod n$. since $a^k \equiv 1 \bmod n$ or $a^{2^j k} \equiv -1 \bmod n$ for some $0 \leq j \leq e-1$ we have $a^{2^e k} = a^{n-1} \equiv 1 \mod n$. hence the ord$(a)$ divides $\gcd(\phi(n), n-1) = \gcd(p^{\alpha-1}(p-1), p^\alpha - 1)$. since $p-1 \mid p^\alpha - 1$ and $\gcd(p, p^\alpha - 1) = 1$ we have $\gcd(p^{\alpha-1}(p-1), p^\alpha - 1) = p-1$. hence $a^{p-1} \equiv 1 \mod p^\alpha$.
conversely, let $a^{p-1} \equiv 1 \mod p^\alpha$, $p-1 = 2^r s$ where s is odd and $r \geq 1$. since $p-1 \mid p^\alpha - 1 = 2^e k$, we have $s \mid k$, $r \leq e$. since,

$$a^{p-1} = a^{2^r s} = (a^s)^{2^r} \equiv 1 \mod p^\alpha,$$

order of a^s is 2^j where $0 \leq j \leq r$.
the case when $j = 0$, $a^s \equiv 1 \mod p^\alpha \implies a^k \equiv 1 \mod p^\alpha$. lets consider the case when $j \geq 1$ then $b := (a^s)^{2^{j-1}}$ satisfies $b^2 \equiv 1 \mod p^s$, $b \not\equiv 1 \mod p^s$. hence $p^s \mid (b+1)(b-1) \implies p^s \mid (b+1)$ $or (b-1)$ so $b \equiv \pm 1 \mod p^s$. since $b \not\equiv 1 \mod p^s$ we have $b \equiv -1 \mod p^s$. now $b = a^{2^{j-1} s} \equiv -1 \mod p^\alpha$, $s \mid k$ and k is odd, raise power both side by k/s we get $a^{2^i k} \equiv -1 \mod p^\alpha$ where $i = j-1 \in \{0, \cdots, r-1\}$.

the proof that Miler-Rabin liars for $n = p^\alpha$ forms a group is trivial.

□

Theorem 2.2.9. The equation $a^{p-1} \equiv 1 \bmod p^\alpha$ has $p-1$ solutions modulo p^α for each α.

Proof. we will prove by induction on α. when $\alpha = 1$, this holds due to fermat's little theorem. If $a^{p-1} \equiv 1 \bmod p^\alpha$, then there is a unique a' mod $p^{\alpha+1}$ such that $a'^{p-1} \equiv 1 \bmod p^{\alpha+1}$ and $a' \equiv 1 \bmod p^\alpha$. Now, $a' \equiv 1 \bmod p^\alpha \implies a' \equiv a + cp^\alpha \bmod p^{\alpha+1}$ where c is well-defined mod p.
By binomial theorem,

$$(a + cp^\alpha)^{p-1} \equiv a^{p-1} + (p-1)a^{p-2}cp^\alpha \bmod p^{\alpha+1}.$$

since, $a^{p-1} \equiv 1 \bmod p^\alpha$, $a^{p-1} = 1 + p^\alpha N$ for some integer N , we need to find c for which,

$$(1 + p^\alpha N) + (p-1)a^{p-2}cp^\alpha \equiv 1 \bmod p^{\alpha+1} \iff \text{N - } a^{p-2}c \equiv 0 \bmod p$$

since, a and p are invertible this has a unique solution for c mod p. □

Theorem 2.2.10. G_n is a group under multiplication modulo n and it contains every Miller-Rabin liar for n and is a proper subgroup of the group of invertible numbers modulo n.

Proof. If $a^k \equiv 1 \bmod n$, then $a^{2^{i_0}k} \equiv 1 \bmod n$. and, If for some i, $0 \le i \le e-1$, then by the maximality of i_0, $i \le i_0$ and $a^{2^{i_0}k} \equiv -1 \bmod n$ if $i = i_0$ and if $i < i_0$ $a^{2^{i_0}k} \equiv 1 \bmod n$. thus, every Miller-Rabin liar for n is in G_n.
Now, let p be a prime factor of n and $n = p^\alpha m$, where $\alpha \ge 1$ and $p \nmid m$. both p^α and m are odd integers (> 1).
By Chinese remainder theorem, there exist an integer a, $1 \le a \le n-1$ satisfying,

$$a \equiv a_0 \ mod \ p^\alpha, \quad a \equiv 1 \ mod \ m$$

$\gcd(a, n) = 1$, since $\gcd(a_0, n) = 1$. Now,

$$a^{2^{i_0}k} \equiv a_0^{2^{i_0}k} \equiv (-1)^k \equiv -1 \ mod \ p^\alpha \implies a^{2^{i_0}k} \not\equiv 1 \ mod \ n$$

and,

$$a^{2^{i_0}k} \equiv 1 \mod m \implies a^{2^{i_0}k} \not\equiv 1 \ mod \ n$$

. Hence, $a^{2^{i_0}k} \equiv \pm 1 \mod n$ so, $\gcd(a, n) = 1$ and $a \notin G_n$. □

Theorem 2.2.11. let $n>1$ be an odd and composite integer. The fraction of integers from 1 to n that are Miller-Rabin witnesses for n is greater than $3/4$ except at $n = 9$, where the proportion is $3/4$. Equivalently, the fraction of integers from 1 to n that are Miller-Rabin strong liar for n is less than $1/4$ except at $n = 9$, where the proportion is $1/4$.

Proof. we will show that the fraction of strong liar has an upper bound of $1/4$ which is achieved only at $n = 9$. lets consider the case when $n = p^\alpha$, where p is an odd prime and $\alpha \geq 2$. By Theorem 2.2.8, Miller-Rabin strong liar for $n = p^\alpha$ are the solutions to $a^{p-1} \equiv 1 \bmod p^\alpha$, such a are closed under multiplication modulo p^α.
From Theorem 2.2.9 there are $p - 1$ Miller-Rabin strong liar mod p^α, their density is given by,

$$p - 1/(p^\alpha - 1) = 1\ /(1 + p + \cdots + p^{\alpha-1})$$

since $\alpha \geq 2$, this ratio is at most $1/(1 + p)$, which is atmost $1/4$(equal to $1/4$ when $\alpha = 2$ and $p = 3$ i.e., $n = 9$). for any other p^α the value is less than $1/4$.

Now, let n has atleast two different prime factors. write $n - 1 = 2^e k$, k is odd and $e \geq 1$.
let i_0 be the largest index in $\{0, 1, \cdots, e - 1\}$ such that some integer a_0 satisfies $\gcd(a_0, n) = 1$ and $a_0^{2^{i_0}} \equiv -1 \bmod n$.
consider the set $G_n = \{\ 1 \leq a \leq n - 1 : a^{2^{i_0}k} \equiv \pm 1 \bmod n\ \}$ for $i \geq 0$,

By Theorem 2.2.10 G_n is a group under multiplication modulo n and it contains every Miller-Rabin liar for n and is a proper subgroup of the group of invertible numbers modulo n.
since n is not a prime, $\phi(n) < n - 1$. we will show that the proportion:

$$\{MRliar(n)\}/(n - 1) < |G_n|/(\phi(n)) \leq 1/4$$

Claim: for every $a \in G_n$, $a^{n-1} \equiv 1 \bmod n$.

$2^{i_0+1}k$ divides $2^e k = n - 1$, since $i_0 \leq e - 1$. for any $a \in G_n$, $a^{2^{i_0}}k \equiv \pm 1 \bmod n \implies \mathrm{a}^{2^{i_0+1}}k \equiv 1 \bmod n$. hence, $a^{n-1} \equiv 1 \bmod n$.

Since, a carmichael number has atleast three different prime factors, we will consider two cases, first when n is not a carmichael number and second when it has atleast three different prime factors.

case 1: n is not a carmichael number.

consider,

$$F_n = \{1 \leq a \leq n-1 : a^{n-1} \equiv 1 \ mod \ n\}$$

then,

$$G_n \subset F_n \subset \{1 \leq a \leq n-1 : \gcd(a,n) = 1\}$$

all three are group under multiplication modulo n.

Claim: both the above containments are strict.

Since, n is not a Carmichael number, there exist an integer x, $\gcd(x,n) = 1$, and $x \notin F_n$. This implies second containment is strict.
we have already shown that G_n is a proper subgroup of the group of invetible numbers modulo n. In that proof, we constructed an integer a, $1 \leq a \leq n-1$ such that $a \notin G_n$ and $a \in F_n$.
Since a proper subgroup of a group is atmost half the size of group $\phi(n)/ \mid F_n \mid \geq 2$, $\mid F_n \mid / \mid G_n \mid \geq 2$ so,

$$\phi(n)/ \mid G_n \mid = \phi(n)/ \mid F_n \mid \times \mid F(n) \mid / \mid G_n \mid \geq 2.2 \geq 4$$

case 2: n has atleast three different prime factors.

write $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_t^{\alpha_t}$ for distict prime p_i, $\alpha_i \geq 1$ and $t \geq 3$.
Consider,

$$H_n = \{1 \leq a \leq n-1 : a^{2^{i_0}k} \equiv \pm 1 \mod p_i^{\alpha_i} \ for \ i = 1, \cdots, t\}$$

Then,

$$G_n \subset H_n \subset 1 \leq a \leq n-1 : gcd(a,n) = 1\}$$

for integers j and k,

$$j \equiv k \ mod \ n \iff j \equiv k \ mod \ p_i^{\alpha_i} \ for \ i = 1, \cdots, t$$

Now, consider the group homomorphism f between groups H_n and $P = \{\pm 1 \ mod \ p_1^{\alpha_1}\} \times \{\pm 1 \ mod \ p_2^{\alpha_2}\} \times \cdots \times \{\pm 1 \ mod \ p_t^{\alpha_t}\}$ given by,

$$f(a \ mod \ n) = (a^{2^{i_0}k} \ mod \ p_1^{\alpha_1} \cdots, a^{2^{i_0}k} \ mod \ p_i^{\alpha_i}, \cdots, a^{2^{i_0}k} \ mod \ p_t^{\alpha_t})$$

Let $K_n = \ker f$ then, $K_n \subset G_n \subset H_n$. we have $\mid P \mid = 2^t$. Now, we will show that f is surjective. It is sufficient to show $(-1, 1, 1, \cdots, 1)$ is in image the image of f.

By definition of i_0, there exist an integer a_0 such that $a^{2^i_0} \equiv -1 \ mod \ n$. By Chinese remainder theorem there exist an a, $1 \leq a \leq n-1$ such that,

$$a \equiv a_0 \ mod \ p_1^{\alpha_1},$$

$$a \equiv 1 \ mod \ p_t^{\alpha_t}, \ for \ \alpha \geq 2$$

implies,

$$a^{2^{i_0}k} \equiv a_0^{2^{i_0}k} \equiv (-1)^k \equiv -1 \ mod \ p_1^{\alpha_1}$$

and,

$$a^{2^{i_0}k} \equiv 1 \ mod \ p_t^{\alpha_t}, \ for \ t \geq 2$$

then, $f(a \ mod \ n) = (-1, 1, 1, \cdots, 1)$.
Now, $\mid f(H_n) \mid = 2^t$ and $\mid f(G_n) \mid = 2$ which implies $\mid H_n \mid / \mid K_n \mid = 2^t$ and $\mid G_n \mid / \mid K_n \mid = 2$ hence $\mid H_n \mid / \mid G_n \mid = 2^{t-1}$ which is atleast 4(since, $t \geq 3$).
so,

$$\phi(n)/ \mid G_n \mid = \phi(n)/ \mid H_n \mid \times \mid H(n) \mid / \mid G_n \mid \geq \mid H(n) \mid / \mid G_n \mid \geq \ 4$$

this completes the proof.

□

Remark: Let n be an odd composite integer then the probability that Miller-Rabin test for n with $'t'$ number of iterations declares it to be prime is less than $(1/4)^t$ whereas the probability that Solovay-Strassen test for n with $'t'$ number of iterations declares it to be prime is less than $(1/2)^t$. It can be shown that if a is a Miller-Rabin liar, then it is also Euler liar and also Solovay-Strassen test involves computation of Jacobi symbol hence we can conclude Miller-Rabin test is never worse than Solovay-Strassen test.

2.3 True primality tests

These tests are also known as primality proving algorithms and used to prove if the given integer is a prime. True primality tests are more computationally intensive than the probabilistic primality tests. Hence, before applying True primality test to integer n we should apply less computational intensive probabilistic primality test such as Miller Rabin test for primality.

Definition 2.3.1. If a True primality test outputs an integer n to be prime, then n is called *provable prime*.

We know that, If $n > 1$ has no prime factor $\leq \lfloor\sqrt{n}\rfloor$, then n is prime. We can use the sieve of Eratosthenes to get a table containing primes up to $\sqrt{n}$ or use trial division method. These methods are easy to implement as a test for primality but not useful for large integers.
Now, we will discuss True primality tests for an integer n which requires partial or complete factorization of $n-1$. These tests are only useful in the case when the factorization of $n-1$ is easy to compute, i.e., when it is of a special form or constructed via specific methods.

2.3.1 Lucas Test

Theorem 2.3.1. Let $n > 1$ be an integer, then n is prime $\iff \phi(n) = n-1$.

Proof. if n is prime then all the integers from 1 to $n-1$ are relatively prime to n, hence $\phi(n) = n-1$. suppose n is composite integer and let d be divisor of n, $1 < d < n$ then $d \in \{2, \cdots, n-1\}$ with $\gcd(d, n) > 1$. hence $\phi(n) \leq n-2$.

□

The following theorem which is a type of converse of Fermat's little theorem was discovered by Lucas in 1876.

Theorem 2.3.2. (Lucas) Let $n > 1$, if there exist an integer a such that,
(i) $a^{n-1} \equiv 1 \bmod n$,
(ii)$a^{n-1/p_i} \not\equiv 1 \bmod n$ for every prime divisor p_i of $n-1$.
then n is prime.

Proof. from last theorem it is enough to prove $\phi(n) = n-1$. let a be an integer which satisfies both the conditions. from (i) $\text{ord}(a) \mid n-1$. To contrary suppose $\text{ord}(a) \neq n-1$ then $n-1 = k.\,\text{ord}(a)$, for some $k > 1$. Now, let p_i be a divisor of k then,

$$a^{n-1/p_i} = a^{k\text{ord}(a)/p_i} = (a^{\text{ord}(a)})^{k/p_i} \equiv 1 \mod n$$

which contradicts (ii), hence $\text{ord}(a) = n-1$. we know that $\text{ord}(a) \leq \phi(n)$ and $\phi(n) \leq n-1$. since $\text{ord}(a) = n-1$, $\phi(n) = n-1$ implies n is prime.

□

Let n be an integer candidate for primality test. At first we will apply probabilistic primality test such as Miller-Rabin. If after some predefined $'t'$ number of iterations,

each output 'probable prime' then we will use Lucas theorem to test n for primality. If we randomly select an integer $a \in \mathbb{Z}_n$ and check if it has order $n-1$ then expected number of such iterations before we get an integer a of order $n-1$ is $O(\log\log n)$(since, $n/\phi(n) < 6\log\log n$ for $n \geq 5$).

example: Let $n = 2011$, 2011 - 1 = 2.3.5.67,
now test for $a = 7$,

$$7^{2011-1} \equiv 1 \mod 2011,$$
$$7^{(2011-1)/2} \equiv -1 \not\equiv 1 \mod 2011,$$
$$7^{(2011-1)/3} \equiv 205 \not\equiv 1 \mod 2011,$$
$$7^{(2011-1)/5} \equiv 1948 \not\equiv 1 \mod 2011,$$
$$7^{(2011-1)/67} \equiv 948 \not\equiv 1 \mod 2011,$$

hence by Lucas test 2011 must be prime.

2.3.2 Pocklington's Test

The issue with Lucas test is the requirement of the factorization of $n-1$, which is a problem even harder than testing n for primality. Henry C. Pocklington discovered a test which requires only partial factorization of $n-1$.

Theorem 2.3.3. (Pocklington) Let $n \geq 3$ be a positive integer and $n-1 = rf$, where $f = q_1^{e_1} q_2^{e_2} \cdots q_t^{e_t}$ and $\gcd(r, f) = 1$. If there exist an integer a such that:
(i) $a^{n-1} \equiv 1 \bmod n$,
(ii) $\gcd(a^{n-1/q_j} - 1, n) = 1$ for every j, $1 \leq j \leq t$,
If $f > \sqrt{n} - 1$, then n is prime.

Proof. suppose p is a prime factor of n, from (i) we have $\mathrm{ord}(a^r)$ in $\mathbb{Z}_p^*$ a divisor of $(n-1)/r = f$. from (ii) it is not a proper divisor of f. hence it must be equal to f and f divides order of $\mathbb{Z}_p^*$, i.e, $f \mid p-1$ which implies every prime factor of n is congruent to 1 modulo f. so, each prime factor of n is larger than f. but $f \geq \sqrt{n}$, so each prime factor of n is larger than $\sqrt{n}$, which implies n is prime.

□

Let n be an integer such than factorization of one of its divisor $f \geq \sqrt{n}$ is known. we will randomly choose an integer a, $1 < a < n-1$ and test condition (i) and (ii) of Pocklington's theorem, If a satisfy both of the conditions then n is prime. Otherwise, another a is chosen randomly and tested similarly. The probability that a randomly selected integer a, $1 \leq a \leq n-1$ satisfy the conditions (i) and (ii) of Theorem 2.3.3 is given by,

$$\frac{1}{\log q_t} \approx \prod_{i=1}^{t}(1 - \frac{1}{q_i}) \geq 1 - \sum_{i=1}^{t} \frac{1}{q_j}$$

If we are unable to find such an a after a large number of iterations, then n is probably composite. Hence, before applying Pocklington's test, we will apply Miller-Rabin probabilistic primality test.

example: Let $n = 997$, $n - 1 = 12 \times 83$, $\sqrt{997} < 83$. take $a = 2$ and $f = 83$, then

$$2^{997-1} \equiv 1 \mod 997$$
$$\gcd(2^{(997-1)/83} - 1, 997) = 1,$$

from Pocklington's theorem n is prime.

Chapter 3

Integer Factorization Problem

3.1 Introduction

Definition 3.1.1. *(Integer factorization problem)* Let n be the given positive integer then find its prime factorization, i.e, $n = p_1^{e_1} p_2^{e_2} \cdots p_t^{e_t}$ where p_i are distinct primes and $e_i \geq 1$.

In the last chapter, we have discussed Primality testing algorithms. Given as input an integer n, Primality testing algorithms output that n is composite, probable prime or provable prime but does not provide a factor of n. If we provide a non-trivial factor of n, then n is composite, and If the only factors of n are 1 and n, then n is prime. The Integer factorization problem may be harder than the problem of testing integers for primality. So for a given integer n, we should first test it for primality, and if it turns out to be composite then we should try to factor it.

Let $n > 1$ be a positive integer then, $n = ab$ where, $1 < a < n$ and $1 < b < n$ is called *split or non-trivial factorization* of n.
Given an integer n we will find the split of n and then test a and b for primality. If they turn out to be composite, then we split them and apply this process recursively. Hence, to find the prime factorization of any given integer n, it is enough to study algorithms to split n.
the factoring algorithms are of two types,
(i) *Special purpose factoring algorithms*: The running time of these algorithms depends on properties and size of the factor p of composite integer n. for example:- Trial division, Pollard's rho, Pollard's $p-1$, Lenstra's elliptic curve factorization.
(ii) *General purpose factoring algorithms*: The running time of these algorithms depends merely on the size of n. for example:- quadratic sieve and general number field sieve.

3.2 Factoring algorithms

3.2.1 Trial division

Trial division refers to divide given composite integer n by 'small' primes. given composite integer n if we attempt to factor n by trial division then in the worst case when n is product of two primes of same size, it takes $\sqrt{n}$ divisions.

3.2.2 Pollard's rho method

Let n be an integer and $S = \{0, 1, \cdots, n-1\}$. consider the iteration,

$$x_{i+1} = f(x_i)$$

where $f : S \to S$ be some easily-computable random function and $x_0 \in S$ is given. since S is a finite set, the sequence x_0, x_1, $x_2, \cdots$ must eventually cycle. hence there exist $m \geq 0$, $q > 0$ such that,

$$x_{m+q} = x_m \; and \; x_{i+q} = x_i$$

for all $i \geq m$.

Definition 3.2.1. The minimal such m and q are called the *non-periodic part* and *period* of the sequence $\{x_i\}$. If there exist indices i and j such that $x_i = x_j$ then it is called a *collision*.

The most natural method to find a collision in sequence $\{x_i\}$ is to store x_i and then look for duplicates. but, this method requires roughly $O(\sqrt{n})$ memory and $O(\sqrt{n})$ time(memory and time estimates are due to Birthday-problem).
Now we will discuss Floyd's algorithm which reduces the storage requirement to find a collision. his idea is to find $j \leq m + q$ such that,

$$x_j = x_{2j}$$

Algorithm: Floyd's Algorithm

Input: f, x_0.

1. set $x \leftarrow x_0$, $y \leftarrow x_0$
2. repeat $j \leftarrow j + 1, x \leftarrow f(x), y \leftarrow f(f(y))$ until $x = y$.

Let $n = ab$ where $\gcd(a, b) = 1$ and $S = \mathbb{Z}_n \simeq \mathbb{Z}_a \times \mathbb{Z}_b$. Let $f : S \to S$, $f = (f_1, f_2)$,

$$x_{i+1} = f(x_i)$$

Since, period q of $\{x_i\}$ in $\mathbb{Z}_n$ is an upper-bound to the period q_a of $\{x_i\}$ in $\mathbb{Z}_a$ and the period q_b of $\{x_i\}$ in $\mathbb{Z}_b$ we should find q_a or q_b[11]. but a and b are unknown so compute $g = \gcd(x_{m+q} - x_m, n)$. If $1 < g < n$ then we have found a factor of n.
Pollard gave a factorization method using Floyd's cycle finding algorithm which is as follows:
Let n be a composite integer and p be a prime factor of n. let $x_0 = 2$ and $f(x) = x^2+1$ mod p. Now generate the iterative sequence $x_0, x_1, \cdots$ where $x_{i+1} = f(x_i)$ for $i \geq 0$. using Floyd's cycle finding algorithm we can find j such that $x_j \equiv x_{2j} \mod p$. since p is unknown hence to obtain a factor of n, we will compute $g = \gcd(x_m - x_{2m}, n)$. If $n > g > 1$, then Pollard's rho method provides a non-trivial factor of n. since n is very large, the case of obtaining $g = n$ has negligible probability.

Algorithm: Pollard's rho algorithm

Input: a composite integer n.
Output: a non-trivial factor g of n.

1. set $x \leftarrow 2$, $y \leftarrow 2$
2. for $i = 1, 2, \cdots$ do:
   - 2.1. compute $x \leftarrow x^2 + 1 \mod n$, $y \leftarrow y^2 + 1 \mod n$, $y \leftarrow y^2 + 1 \mod n$.
   - 2.2. compute $g = gcd(x - y, n)$.
   - 2.3. If $1 < g < n$ then return g,
   - 2.4. If $g = n$ then the algorithm fails.

Remark: If the Pollard's rho algorithm terminates with failure then we can start with different x_0 or use different polynomial $f(x) = x^2 + k$, $k \neq -2, 0$.
Let p be the smallest prime divisor of n, then the expected time for Pollard's rho method to find p is $O(\sqrt{p})$ modular multiplications or the expected time for algorithm to find a non-trivial factor of n is $O(n^{\frac{1}{4}})$ modular multiplications.

R. P. Brent[4] has proposed a modification to Pollard's rho method. The following cycle finding algorithm is due to Brent.

Algorithm: Brent's cycle finding algorithm

Input: f, x_0
Output: period q of the sequence $\{x_i\}$.

1. set $y \leftarrow x_0$, $r \leftarrow 1$, $done \leftarrow false$.
2. repeat until *done*
   2.1. $x \leftarrow y$, $j \leftarrow k$, $r \leftarrow 2 \times r$
   2.2. repeat until $k \geq r$ or *done*
      2.2.1. $k \leftarrow k + 1$
      $y \leftarrow f(y)$
      $done \leftarrow (x = y)$
3. return $q = k - j$

Now, we will explain above algorithm. The algorithm outputs period q of the sequence $\{x_i\}$, and we use variable j to keep track of the period. We start with $y \leftarrow x_0$ and $x \leftarrow y$. for each value of k we update y and check if $x = y$. If $x = y$ we are done but, if $x \neq y$ we increase k by 1 and repeat procedure until k is smaller than r. when k exceeds r and still $done \leftarrow false$ we update $x \leftarrow y$ and increase r by a factor of 2 and repeat the procedure. since we are working with finite sets, the algorithm will eventually terminate with period $q = k - j$.
By measuring work in units of f evaluations, Brent[4] has shown that on average his cycle finding algorithm is 36% faster than Floyd's cycle-finding algorithm and when applying Brent's cycle-finding algorithm to Pollard's rho method in place of Floyd's cycle-finding algorithm the Pollard's rho algorithm gets about 24% faster.

3.2.3 Pollard's $p - 1$ method

Let $n = pq$, $p < q$ be the integer which we want to factor. choose $a \in \mathbb{Z}_n^*$ at random and compute $a^Q \mod n$(we will provide suitable candidate for Q later). Now, by chinese remainder theorem there exist $a_p \in \mathbb{Z}_p^*$ and $a_q \in \mathbb{Z}_q^*$ such that,

$$a^Q \mod n \leftrightarrow (a_p^Q \mod p, a_q^Q \mod q)$$

Lets assume $p - 1 \mid Q$ and $q - 1 \nmid Q$ then by Fermat's little theorem we have,

$$a^Q \mod n \leftrightarrow (1 \mod p, a_q^Q \mod q)$$

If $a_q^Q \mod q \not\equiv 1 \mod q$ then $p \mid ((a^Q - 1) \mod n)$ and $q \nmid ((a^Q - 1) \mod n)$. hence $gcd(a^Q - 1, n) = p$, gives a non-trivial factor of n.
Now we will provide a suitable candidate for Q.

Definition 3.2.2. let n, B be a positive integers then n is said to be B-*smooth* if all prime factors of n are less than or equal to B.

We will choose B according to the time which we are suppose to spend on algorithm. let Q be the l.c.m. of all powers of primes which are less than or equal to B that are less than or equal to n. If $p^\alpha \leq n$ then $\alpha \leq \lfloor \log n / \log p \rfloor$.so,

$$Q = \prod_{p \leq B} p^{\lfloor \log n / \log p \rfloor}$$

where p runs over all the primes less than or equal to n.
Let p be a factor of n such that $p-1$ is B-smooth then $p-1 \mid Q$. On the other hand if $q-1$ has any factor greater than B then $q-1$ is not B-smooth. Now, for any integer a relatively prime to n we have $a^Q \equiv 1 \mod p$. since p is unknown, we compute $g = gcd(a^Q - 1, n)$. If $n > g > 1$ then g is an non-trivial factor of n and if g is 1 or n then algorithm fails.

Algorithm: Pollard's $p-1$ algorithm

Input: a composite integer n.
Output:a non-trivial factor g of n.

1. select a bound B.
2. select an integer a, $1 < a < n$ at random and compute $g = gcd(a, n)$. If $g > 1$ then return g.
3. for every prime $p \leq B$ do:
   3.1. compute $\alpha \leq \lfloor \log n / \log p \rfloor$
   3.2. using repeated square and multiply algorithm for modular exponentiation to compute $a \leftarrow a^{p^\alpha} \mod n$
4. compute $g = gcd(a-1, n)$
5. If $1 < g < n$ then return g. If $g = 1$ *or* n, then the algorithm has failed to give a non-trivial factor so terminate the algorithm.

If the algorithm terminates with the failure, then one option is to increase B while keeping a fixed. The smoothness bound B is chosen according to the time we want to spend on Pollard's $p-1$ algorithm before moving to other algorithms Pollard's $p-1$ method is not useful when every prime divisor p of n, $p-1$ is not B-smooth i.e, it is divisible by a large prime than B.

3.2.4 Lenstra's Elliptic-curve factorization

This method was given by H. W. Lenstra in 1987. At first we will introduce minimal background to understand this method.

Definition 3.2.3. [9]Let K be a field of characteristic other than 2,3 and let $x^3 + ax + b$ $(a, b \in K)$ be a cubic polynomial with no multiple roots. An *elliptic curve over* K is the set,

$$E = \{(x, y) \in K \times K \mid y^2 = x^3 + ax + b\} \cup O$$

where O is called the point at infinity.

It is known that points on elliptic curve forms an abelian group with the operation '+'. let $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2) \in E$ then $P_1 + P_2 = P_3$ where $P_3 = (x_3, y_3)$,

$$(x_3, y_3) = (\alpha^2 - x_1 - x_2, \alpha(x_1 - x_3) - y_1)$$

where,

$$\alpha = \frac{m_1}{m_2} = \begin{cases} (y_1 - y_2)/(x_1 - x_2) & , if\ P_1 \neq P_2 \\ (3x_1^2 + a)/2y_1 & , if\ P_1 = P_2 \end{cases}$$

Let E be an elliptic curve over $\mathbb{F}_q$, where $q = p^r$ then we have Hasse's theorem which provides a bound on number of $\mathbb{F}_q$ points on E.

Theorem 3.2.1. (Hasse) Let E be an elliptic curve defined over $\mathbb{F}_q$ and let n be the number of $\mathbb{F}_q$-point on E then,

$$\mid N - (q + 1) \mid \leq 2\sqrt{q}$$

Algorithm: Lenstra's algorithm [13]

Input: a composite integer n.
Output: a non-trivial factor of n.

[1] choose a random pair (E, P) where E is elliptic curve over $\mathbb{Z}_n$ and $P = (x, y)$ is a point on the elliptic curve. for this we will randomly choose $a, x, y \in \mathbb{Z}_n$ and compute $b \leftarrow y^2 - x^3 - ax$, $d = gcd(4a^3 + 27b^2, n)$. If $1 < d < n$ then return d. If $d = n$ then we will discard (E, P) and choose another pair (E, P).

[2] select a bound B. compute $Q = \prod_{p \leq B} p^{\lfloor \log n / \log p \rfloor}$, where p runs over primes less than or equal to B.

[3] compute $QP \mod n$ by doubling and addition using the following formula, let $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2) \in E$ then $P_3 = P_1 + P_2 \mod n$ where $P_3 = (x_3, y_3)$,

$$(x_3, y_3) = (\alpha^2 - x_1 - x_2 \mod n, \alpha(x_1 - x_3) - y_1 \mod n)$$

where,

$$\alpha = \frac{m_1}{m_2} = \begin{cases} (y_1 - y_2)/(x_1 - x_2) \mod n & , if\ P_1 \neq P_2 \\ (3x_1^2 + a)/2y_1 \mod n & , if\ P_1 = P_2 \end{cases}$$

[4] if $QP \equiv O \mod n$ then compute $g = gcd(m_2, n)$, otherwise go to step [1] and choose another pair (E, P).

[5] If $1 < g < n$ then return g. otherwise go to otherwise go to step [1] and choose another pair (E, P).

In Pollard's $p-1$ method, we are focusing our hopes on groups $\mathbb{Z}_p^*$ where p runs over prime divisors of n. for a given n these groups are fixed so we are inconclusive when all of these groups has order divisible by a large prime. The above problem encountered with Pollard's $p-1$ method is solved if we work with elliptic curves over $\mathbb{Z}_p$ because elliptic curves over finite fields provide a large number of finite abelian groups and provide more flexibility in choosing an elliptic curve. So we can expect one group to have order not divisible by a large prime(or prime power).

Remark:[9] Let p be the smallest prime factor of n where n is not a prime power and is not divisible by 2 or 3. Assuming some plausible conjecture, Lenstra proved that estimated number of bit operations required to find a factor of n is,

$$exp(\sqrt{(2+\epsilon)\log p \log\log p})$$

where ϵ tends to zero for large p. Lenstra's algorithm is useful to find small factors of n.

3.2.5 Factorization using group $G(n, N)$

Consider the equation,

$$x^2 + Ny^2 \equiv 1 \mod n \tag{3.1}$$

where $N \in \mathbb{Z}_n^*$. The set of solutions of this equation $G(n, N)$ over $(\mathbb{Z}_n^*)^2$ forms group under the group operation[12] defined as,
let (x_1, y_1) and (x_2, y_2) be two elements in $G(n, N)$ then,

$$(x_1, y_1).(x_2, y_2) = (x_1x_2 + Ny_1y_2 \mod n, x_1y_2 - x_2y_1 \mod n)$$

let $n = pq$ where p and q are primes then(from [1]),

$$|G(n,N)| = \begin{cases} (p-1)(q-1) & ,if \ -1/N \in Q_n \\ (p-1)(q+1) & ,if \ -1/N \in Q_p \ and -1/N \in Q'_q \\ (p+1)(q-1) & ,if \ -1/N \in Q'_p \ and -1/N \in Q_q \\ (p+1)(q+1) & ,if \ -1/N \in Q'_p \ and -1/N \in Q'_q \end{cases}$$

$e = (1,0)$ is identity of the group and for any $g \in G(n,N)$ we have $g^2 = e$, hence $G(n,N)$ is a finite-abelian group.

(1,0) is a solution to equation (3.1). The line $l : y = b(x-1)$ passes through (1,0). Now look at the intersection of l and $x^2 + Ny^2 = 1$,

$$x^2 + N(b(x-1))^2 = 1$$

by solving, we get point of intersection,

$$(x,y) = (\frac{kb^2-1}{kb^2+1}, \frac{-2b}{kb^2+1}) \tag{3.2}$$

let f be defined as,

$$f(z=(x,y)) = \begin{cases} \beta.z & ,if \ x \equiv 1 \mod 3 \\ z^2 & ,if \ x \equiv 0 \mod 3 \\ \alpha.z & ,if \ x \equiv 2 \mod 3 \end{cases}$$

where α and β be two fixed element in $G(n,N)$. α and β are computed using (3.2) and reducing (x,y) modulo n.
Define a sequence of group elements z_0,z_1,z_2,$\cdots$ by $z_{i+1} = f(z_i)$ where $z_0 = (0,1)$. Using Floyd's cycle-finding algorithm we can find index i for which $z_i = z_{2i}$. Now compute $g = \gcd(x_i - x_{2i}, n)$(also we can compute $\gcd(y_i - y_{2i}, n)$). If $1 < g < n$ then g is a non-trivial factor of n. If $g = 1$ or $g = n$ we can try this method with different α and β or with different N.

Chapter 4

Cryptanalysis of RSA

4.1 Introduction

In Public-key(or Asymmetric) cryptosystems each entity has a public key and a private key, and given the public key, it is not feasible to compute the respective private key. RSA is a public-key cryptosystem invented by Ron Rivest, Adi Shamir and Leonard Adleman. The security of RSA is based on integer factorization problem. Over the past years, RSA has been subjected to various attacks, but none of these attacks are a threat to RSA if RSA is implemented adequately. In this chapter, we will discuss RSA cryptosystem, its security and various attacks on RSA. In the end, we will conclude that these attacks can be avoided if we implement RSA properly.

4.2 RSA cryptosystem

In this section, we will discuss how keys are generated in RSA and, how encryption and decryption in RSA works. Throughout this chapter, we will assume that Alice(A) and Bob(B) are two entities who are trying to communicate.
At first, we will state Euler's theorem, which is used several places in this chapter.

Theorem 4.2.1. (Euler) Let $n > 1$ be an integer, If $a \in \mathbb{Z}_n^*$ then $a^{\phi(n)} \equiv 1 \mod n$.

example:- take $n = 15 = 3 \times 5$ and $a = 2$. then, $\phi(15) = 8$. since, $15 \mid 255 = 15 \times 17$,

$$2^{\phi(15)} = 2^8 \equiv 256 \equiv 1 \mod 15$$

RSA key generation: Each communicating entity A(or B) generate keys in the following manner,

1. randomly generate two large, distinct primes p and q of roughly same size, each having $n/2$ bits.

2. compute $n = pq$ and $\phi = (p-1)(q-1)$.

3. select an integer e at random such that $1 < e < \phi$ and it is relatively prime to ϕ.

4. compute inverse modulo ϕ of e using Extended Euclidean algorithm call it d, $1 < d < \phi$ and $ed \equiv 1 \mod \phi$.

The integer n, e *and* d are called RSA modulus, public exponent, and decryption exponent respectively. The pair (n, e) is the public-key, which is used for message-encryption and it is publicly available. The integer d is the private key which is used for message-decryption, it is kept secret and known to the message recipient only. Typically the size of RSA modus n is 1024 bits and p, q is of size 512 bits each.

RSA cryptosystem

Let two entities A and B are communicating and A wants to send a message $m \in \mathbb{Z}_n^*$ to the entity B. Then, at first A obtains the public key pair (n, e) of entity B and compute,

$$c = m^e \mod n$$

and send the encrypted message c to B. Now, In order to decrypt c the entity B computes $c^d \mod n$. By Euler's theorem,

$$c^d = m^{ed} = m \mod n$$

4.3 Security of RSA cryptosystem

Fact: Let $n = pq$ be the RSA modulus and $(N,\ e),\ d$ be public and private keys respectively then, given the factorization of n the private key d can be recovered efficiently. conversely given the private key d the RSA modulus n can be factored efficiently.

Proof. Suppose factorization of n is given then we will compute ϕ and using extended Euclidean algorithm we will recover d. Now, suppose d is given. since $ed \equiv 1 \mod \phi$, there exist an integer x such that $ed - 1 = x\phi$. Now by Euler's theorem for any $a \in \mathbb{Z}_n^*$, $a^{ed-1} \equiv 1 \mod n$. $\phi = (p-1)(q-1)$ is even number since, p and q are odd. So, let $ed - 1 = 2^r s$ where s is an odd integer. we know that for atleast half of elements $a \in \mathbb{Z}_n^*$, there exist an integer $j \in [1, r]$ such that $a^{2^{j-1}s} \not\equiv \pm 1$ and $a^{2^j s} \equiv 1 \mod n$. Now, $g = gcd(\mathrm{a}^{2^{j-1}s} - 1, n)$ gives a non-trivial factor of n. The probability that a randomly chosen $a \in \mathbb{Z}_n^*$ will lead to the factorization of n is atleast half. we will choose $a \in \mathbb{Z}_n^*$ at random and check if it satisfies above condition. if it fails then we choose another a and check above condition.

□

If we can efficiently factor n, then we can recover private key d from public key pair (N, e). One of the widely studied problems in Algorithmic number theory is Integer factorization problem. Over the years the factoring algorithms have improved, but yet we are unable to discover any classical polynomial-time factoring algorithm. Currently, the fastest factoring algorithm is General Number Field Sieve(GNFS) whose running time n-bit integers is $\exp((64/9)^{1/3} + o(1)n^{1/3}log^{2/3}n)$.

4.3.1 Elementary attacks

Common modulus attack:
In order to avoid prime generation again and again for each user, one can fix n. let each user uses same modulus n. each user i has unique pair e_i, d_i.
Now let $c = m^{e_i} \mod n$ is intended for user i. any user j with the help of his e_i, d_i pair can factor n. once n is factored user j can recover private key d_i of user i from his public key e_i. hence, the same RSA-modulus should not be used by more than one user.

Forward search attack:
Let the message space is small. In order to decrypt m from the ciphertext c attacker can try to encrypt all the messages m until c is obtained. The process of appending a pseudorandomly generated bitstring to plaintext message before encryption is called *salting* the message. The forward search attack can be prevented by salting the messages.

Adaptive-chosen ciphertext attack:
Let c_1 and c_2 be ciphertexts corresponding to plaintext messages m_1 and m_2. Now,

$$(m_1 m_2)^e \equiv m_1^e m_2^e \equiv c_1 c_2 \mod n$$

let the attacker wants to decrypt $c = m^e \mod n$. suppose the attacker by some means can get the plaintext message corresponding to any arbitrary ciphertext other than c. Then attacker will choose ciphertext $c_0 = cr^e \mod n$ for some random $r \in \mathbb{Z}_n^*$ and obtains $m_0 = c_0^d \mod n$. Since,

$$m_0 \equiv c_0^d \equiv c^d (r^e)^d \equiv mr \mod n$$

the attacker will recover the desired plaintext message by computing $m = m_0 r^{-1} \mod n$. In order to prevent this attack, we should impose some structural constraint on the plaintext messages, and the ciphertext which decrypts to the plaintext message not having this structure should be rejected.

4.3.2 Low public exponent attacks

In order to reduce encryption time, we may prefer to use small public exponent e. But working with small e can make RSA susceptible to attacks. In this section, we will discuss attacks based on Coppersmith's Theorems which use the notion of Lattice and Lenstra-Lenstra-Lovász(LLL) lattice basis reduction algorithm. So, first, we will introduce a minimal background on the lattice(the background is from [10]).

Definition 4.3.1. Let $x = (x_1, x_2, \cdots, x_n)$, $y = (y_1, y_2, \cdots, y_n) \in \mathbb{R}^n$ then the *inner product* of x and y is a real number denoted by $< x, y >$ and given by,

$$< x, y >= x_1y_1 + x_2y_2 + \cdots + x_ny_n$$

Definition 4.3.2. Let $x = (x_1, x_2, \cdots, x_n) \in \mathbb{R}^n$ then the *length* of x is a real number given by,

$$\| x\| = \sqrt{< x, x >} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

Definition 4.3.3. Let $B = \{v_1, v_2, \cdots, v_m\} \subset \mathbb{R}^n (m \geq n)$ be a set of linearly independent vectors. The set,

$$L = \mathbb{Z}v_1 + \mathbb{Z}v_2 + \cdots + \mathbb{Z}v_m$$

is called a *Lattice of dimension* m and B is called a *basis* for the lattice L.

If $n = m$ the *determinant* of L is defined as the determinant of the matrix of order $m \times m$ whose rows are the vectors $v_1, v_2, \cdots, v_m$.

Now, we will introduce the notion of *reduced-basis* which is based on the Gram-Schmidt orthogonalization process. It consists of vectors of relatively small lengths.

Definition 4.3.4. Let $B = \{v_1, v_2, \cdots, v_n\}$ be a basis for the Lattice L in $\mathbb{R}^n$. Define the real numbers $\mu_{i,j} (1 \leq j < i \leq n)$ and the vectors $v_i^* (1 \leq i \leq n)$,

$$\mu_{i,j} = \frac{< v_i, v_j^* >}{< v_j^*, v_j^* >}, \quad 1 \leq j < i \leq n \tag{4.1}$$

$$v_i^* = v_i - \sum_{j=1}^{i-1} \mu_{i,j} v_j^*, \quad 1 \leq i \leq n \tag{4.2}$$

the basis B is called a $reduced - basis$ if for $1 \leq j < i \leq n$,

$$| \mu_{i,j} | \leq 1/2 \tag{4.3}$$

and for $1 < i \leq n$,

$$\left\| v_j^* \right\|^2 \geq (\frac{3}{4} - \mu_{i,i-1}^2) \left\| b_{i-1}^* \right\|^2 \tag{4.4}$$

Given a lattice L and its basis B the Lenstra-Lenstra-Lovász(LLL) lattice based reduction algorithm finds a reduced basis for L in polynomial-time.

Theorem 4.3.1. Let L be a lattice spanned by $B = \{v_1, v_2, \cdots, v_m\}$. If B is given as input in LLL algorithm then it outputs $v \in L$ satisfying,

$$\| v\| \leq 2^{m/4} \det(L)^{\frac{1}{m}}$$

Let $n = pq$ be k-bit RSA modulus where p, q are large primes satisfying,

$$\sqrt{n}/2 < q < p < 2\sqrt{n}$$

The following theorem is due to Don Coppersmith.

Theorem 4.3.2. (Coppersmith[3]) Let $f(x, y)$ be a bivariate polynomial over $\mathbb{Z}$ having maximum degree d in each variables separately. assume the coefficients of f are relatively prime as a set. Let X, Y be bounds on the desired solutions x_0, y_0. Define $f^*(x, y) := f(Xx, Yy)$ and let D be the absolute value of the largest coefficient of f^*. If $XY < D^{2/(3d)}$, then in time polynomial in $(logD,\ 2^d)$, we can find all integer pairs (x_0, y_0) with $f(x_0, y_0) = 0$, $| x_0 | < X$, and $| y_0 | < Y$.

Given appropriate bounds on x_0, y_0 Theorem 4.3.2 provides small solutions to a bivariate polynomial f efficiently. The proof of Corollary 4.3.2.1 and Theorem 4.3.3 follows the approach given in [3]. We will use Theorem 4.3.2 to get following useful result.

Corollary 4.3.2.1. let $n = pq$ be an k-bit RSA modulus. Given $t \geq 2^{k/4}$ and $p_0 := p \mod t$ then we can factor n in time polynomial in k.

Proof. since p_0 is given, $q_0 := n/p_0 \mod t$. Now consider the polynomial,

$$f(x, y) = (tx + p_0)(ty + q_0) - n$$

we will look for solution (x_0, y_0) of above polynomial with $x_0 < X = 2^{k/2+1}/t$ and $y_0 < Y = 2^{k/2+1}/t$. Since gcd of the coefficients of $f(x, y)$ is t, hence In order to apply Coppersmith theorem consider $g(x, y) = f(x, y)/t$. Now,

$$g^*(x, y) = g(xX, yY) = f(xX, yY)/t = \frac{(txX + p_0)(tyY + q_0) - n}{t}$$

the largest coefficient of $g^*(x, y)$ is atleast $2^{k+2}/t$. The condition,

$$XY = \frac{2^{k+2}}{t^2} < (2^{n+2}/t)^{2/3}$$

holds whenever $t > 2^{(n+2)/4}$ and can be reduced to $t \geq 2^{n/4}$ by exhaustive search on first two bits of x_0 and y_0.

□

The attack based on following Theorem 4.3.3 is called *Partial-key exposure attack* i.e., given $k/4$ least significant bits of d one can recover d completely. we require small public exponent so that it is possible to perform the exhaustive search on the values less than it.

Theorem 4.3.3. [3]Let $n = pq$ be a k-bits RSA modulus, $\phi = (p-1)(q-1)$, (n, e) be the public key and d be the private key. If $k/4$ least significant bits of d are given then there is an algorithm which recovers d completely in polynomial time in k and e.

Proof. We have $k/4$ least significant bits d_0 of d i.e., $d = d_0 \mod 2^{k/4}$. Since, $ed \equiv 1 \mod \phi$ there exist an integer x such that,

$$ed - x\phi = ed - x(n - r + 1) = 1$$

Where $r = p + q$ or,

$$ed = 1 + x\phi = 1 + x(n - r + 1)$$

reducing above equation modulo $2^{k/4}$ we get,

$$ed_0 \equiv 1 + x(N - r + 1) \mod 2^{k/4}$$

Since $\phi > d$ we have $x < e$. At first we will try each value of $x \in [0,\ e]$ and for each x value solve above equation to obtain $r \mod 2^{k/4}$.
Now, consider the equation,

$$p^2 – rp + n = p^2 - (p + q)p + n = 0$$

Reducing above equation modulo $2^{k/4}$ we get,

$$p^2 – rp + n \equiv 0 \mod 2^{k/4}$$

Putting $r \mod 2^{k/4}$ in above equation we solve for $p_0 :\equiv p \mod 2^{k/4}$ Now let $t = 2^{k/4}$ and apply last theorem which factors n in time polynomial in k. since the correct value for p_0 can be found in atmost e attempts, the total running time of algorithm to factor n is linear in e. we know that given factorization of n, we can expose the private d efficiently.

□

Now, we will discuss another low public exponent attack on RSA due to Coppersmith. The proof of the following theorem uses Lenstra-Lenstra-Lovász(LLL) lattice basis reduction algorithm.

Theorem 4.3.4. (Coppersmith[8]) Let $f \in \mathbb{Z}[x]$ be a monic polynomial of degree δ and n be an integer. given (n, f) one can efficiently find all the integers $\mid x_0 \mid < X = n^{(1/\delta)-\epsilon}$ for some $\epsilon \geq 0$ satisfying $f(x_0) = 0 \mod n$. The running time is dominated by the time it takes to run the LLL lattice basis reduction algorithm on a lattice of dimension $O(s)$ where s=min$(\log n, 1/\epsilon)$.

So, using above theorem we can find all roots of $f \mod n$ which are less than the bound X. The running time of algorithm decreases as the bound X gets smaller. Now, we will provide a sketch of the proof of Coppersmith's Theorem 4.3.4(we will follow the approach given in [2]). To prove the theorem we require following lemma which is due to Howgrave-Graham.

Lemma 4.3.5. Let $g(x) \in \mathbb{Z}[x]$ be a polynomial of degree δ and X be a positive integer. suppose $\|g(xX)\| < n/\sqrt{\delta}$. If $x_0 < X$ satisfies $g(x_0) = 0 \mod n$, then $g(x_0) = 0$ holds over the integer.

Proof. Using Schwarz inequality we get,

$$\mid g(x_0) \mid = \mid \sum a_i x_0^i \mid = \mid \sum a_i X^i (x_0/X) i \mid \leq \sum \mid a_i X^i (x_0/X)^i \mid$$

$$\leq \sum \mid a_i X^i \mid \leq \sqrt{\delta} \|g(xX)\| < n$$

since $g(x_0) = 0 \mod n$, $\mid g(x_0) \mid < n$ we get $g(x_0) = 0$.

□

We know that it is easy to find roots of polynomial over integers. from the above lemma, if we have a polynomial having small norm then its modulo n roots are also roots over $\mathbb{Z}$. In order to find roots of $f(x) \mod n$ we should look for a polynomial $g \in \mathbb{Z}$[x] which has same roots as $f \mod n$ and small norm as to apply Lemma 4.3.5. for this we need to find $h \in \mathbb{Z}$[x] such that $g = hf$ has small norm.
Now, we know that $f(x_0) = 0 \mod n$ then $f(x_0)^k = 0 \mod n^k$ for any k. for some predefined m define,

$$h_{i,j}(x) = n^{m-j} x^i f(x)^j$$

then for any $i \geq 0$ and $0 \leq j \leq m$, x_0 is a root of $h_{i,j}(x) \mod n^m$. we will find integer linear combination $g(x)$ of $h_{i,j}(x)$ to use Lemma 4.3.5. Now, Let L be lattice spanned by $h_{i,j}(x)$ where $i = 0, \cdots, \delta - 1$ and $j = 0, \cdots, m$ hence dimension of L is $s = (m+1)\delta$. Due to Theorem 4.3.1, LLL-algorithm outputs a polynomial $g \in L$ satisfying,

$$\| g\| \leq 2^{s/4} \det(L)^{\frac{1}{s}}$$

we need,

$$2^{s/4} \det(L)^{\frac{1}{s}} < n^m/\sqrt{s}$$

It can be shown that for large enough m the bound is satisfied and when $X = n^{(1/\delta)-\epsilon}$ it suffices to take $m = O(k/\delta)$ where $k = \min(\log n, 1/\epsilon)$. The running time is dominated by the time it takes to run the LLL lattice basis reduction algorithm on a lattice of dimension $O(k)$. Now we will apply Lemma 4.3.5 to conclude the proof.

□

Let n be the RSA-modulus and e, d be public and private exponent respectively where e is small. suppose c be the ciphertext we want to decrypt then consider,

$$f(x) = x^e - c$$

then by Theorem 4.3.4 we can find all the integer $\mid x_0 \mid < n^{1/e}$ satisfying,

$$f(x_0) = 0 \mod n$$

$$x_0^e = c \mod n$$

as $x_0^e < n$ we will compute integer e-th root $c^{1/e}$ to decrypt c.

4.3.3 Low Private exponent attack

In order to reduce the decryption time, we may prefer to use a small private exponent d. M. Wiener has shown that a small d can lead to the exposure of d. At first, we will introduce a minimal background about continued fractions(the background is from [5])

Definition 4.3.5. A finite *continued fraction* is an expression given by,

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots + \cfrac{1}{a_n}}}}$$

where $a_i \in \mathbb{Z}$, $a_i \geq 1$. we will denote above continued fraction and the rational number which it represent as $[a_0; a_1, a_2, \cdots, a_n]$.
let $x/y (y \neq 0)$ be a rational number then it has exactly two continued fractions,

1. If $x/y = [a_0; a_1, a_2, \cdots, a_n]$ with $a_n = 1$, then $x/y = [a_0; a_1, a_2, \cdots, a_{n-1} + 1]$
2. If $x/y = [a_0; a_1, a_2, \cdots, a_n]$ with $a_n \neq 1$, then $x/y = [a_0; a_1, a_2, \cdots, a_n - 1, 1]$.

hence every rational number has a unique shortest continued fraction. This expansion can be computed using the Euclidean algorithm. Consider $x/y = x_0/x_1$, $\gcd(x_0, x_1) = 1$ and perform Euclidean algorithm,

$$\begin{aligned} x_0 &= x_1a_0 + x_2 \text{ where } 0 \leq x_2 < x_1, \\ x_1 &= x_2a_1 + x_3 \text{ where } 0 \leq x_3 < x_2, \\ &\vdots \\ x_n &= x_{n+1}a_n \text{ where } x_{n+1} = 1 \end{aligned}$$

So, the continued fraction exapnsion is $x_0/x_1 = [a_0; a_1, a_2, \cdots, a_n]$.

Definition 4.3.6. Let $x/y = [a_0; a_1, a_2, \cdots, a_n]$ then the rational numbers,

$$[a_0], [a_0; a_1], \cdots, [a_0; a_1, \cdots, a_n]$$

are said to be *convergents* to x/y.

The low private exponent attack due to M. Wiener is based on the following theorem,

Theorem 4.3.6. (Poincaré) Let $x, y, a, b \in \mathbb{Z}$ where $y \geq 1$, $1 \leq b < y$, $\gcd(x, y) = 1$. If

$$\left|\frac{x}{y} - \frac{a}{b}\right| < \frac{1}{2b^2}$$

then a/b is a convergent of x/y.

Theorem 4.3.7. (M. Wiener) Let $n = pq$ with $q < p < 2q$ and $d < 1/3n^{1/4}$. Given the public-key (n, e) the private key d can be recovered efficiently.

Proof. In this proof, we will use approximations based on continued fraction. Since $ed \equiv 1 \mod \phi$ there exist an integer k such that $ed - k\phi = 1$. Since,

$$\left|\frac{e}{\phi} - \frac{k}{d}\right| = \frac{1}{\phi d}$$

thus k/d is an approximation of e/ϕ.
Since $p + q - 1 < 3\sqrt{n}$ and $\phi = n - (p + q) + 1$, we get

$$| n - \phi | < 3\sqrt{n}$$

Now consider,

$$\begin{aligned} \left|\frac{e}{n} - \frac{k}{d}\right| &= \left|\frac{ed - k\phi - kn + k\phi}{dn}\right| \\ &= \left|\frac{1 - k(n - \phi)}{dn}\right| \leq \left|\frac{3k\sqrt{n}}{dn}\right| = \frac{3k}{d\sqrt{n}} \end{aligned}$$

We have $k\phi < ed$, $e < \phi$ so, $k < d < 1/3n^{1/4}$. hence we get,

$$\left|\frac{e}{n} - \frac{k}{d}\right| \leq \frac{1}{dn^{1/4}} < \frac{1}{2d^2}$$

By Theorem 4.3.6, k/d is a convergent of e/n. Now, we will compute the $\log n$ convergents of the continued fraction for e/n, one of them is equal to k/d. Hence, we will recover the private key d.

□

4.4 Conclusion

The security of RSA cryptosystem is based on the Integer factorization problem. Up to now, there does not exist any classical polynomial-time factoring algorithm. The attacks discussed in this chapter provide issues which should be avoided while implementing RSA.

While choosing RSA modulus $n = pq$ the care must be taken in selecting primes p and q. If the difference between p and q is small then one can factor n by dividing odd integers close to $\sqrt{n}$. Both p and q should be sufficiently large and have the same bitlength to avoid factoring by Lenstra's algorithm. If p is such that $p-1$ and $p+1$ have a large factor then n will avoid factoring due Pollard's $p-1$ algorithm and William's $p+1$ algorithm.

Bibliography

[1] Rana Barua and Mahabir P. Jhanwar. "On the Number of Solutions of the Equation $Rx^2 + Sy^2 = 1 \mod n$". In: *Sankhya: The Indian Journal of Statistics* 72-A.1 (2010), pp. 226–236.

[2] Dan Boneh. "Twenty Years of Attacks on the RSA Cryptosystem". In: *Notices of the American Mathematical Society(AMS)* 46.2 (1999), pp. 203–213.

[3] Dan Boneh, Glenn Durfee, and Yair Frankel. "An Attack on RSA Given a Small Fraction of the Private Key Bits". In: *ASIACRYPT'98* (1998), pp. 25–34.

[4] R. P. Brent. "An improved Monte Carlo factoriation algorithm". In: *BIT* 20 (1980), pp. 176–184.

[5] Levente Buttyan. *Wiener's Attack*. 2015. URL: `http://www.hit.bme.hu/~buttyan/Wiener/WienersAttack.pdf`.

[6] Keith Conrad. *The Miller-Rabin test*. 2017. URL: `http://www.math.uconn.edu/~kconrad/blurbs/ugradnumthy/millerrabin.pdf`.

[7] Keith Conrad. *The Solovay-Strassen test*. 2016. URL: `http://www.math.uconn.edu/~kconrad/blurbs/ugradnumthy/solovaystrassen.pdf`.

[8] D. Coppersmith. "Finding a small root of a univariate modular equation". In: *Advances in Cryptology-EUROCRYPT '96* (1996), pp. 155–165.

[9] Neal Koblitz. *A Course in Number Theory and Cryptography (2nd ed.)* New York: Springer-Verlag, 1994.

[10] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[11] Kapil hari Paranjape. *Factorization and Certificates*. URL: `https://www.imsc.res.in/~kapil/crypto/chap5.pdf`.

[12] John M. Pollard and Claus P. Schnorr. "An Efficient Solution of the Congruence $x^2 + ky^2 = m \mod n$". In: *IEEE Transactions on Information Theory* IT-33.5 (1987), pp. 702–709.

[13] Song Y. Yan. *Primality Testing and Integer Factorization in Public-Key Cryptography*. Springer, 2009.