**Prime Numbers**

**and**

**Cryptography**

**CS101 Project**

By

Ashutosh Singh (2022CSB1071)

Suryadeep Mallick (2022CSB1132)

**Preface**

This research project was taken as part of the project component of the CS101 course for the CSE students of the BTech 22-23 batch. The authors have explored the properties of primes and their various applications in cryptography throughout this paper.

It is highly recommended to strictly view this paper as a reference material to the elaborate video covering all the topics. This will help the reader in getting an eagle’s view of basic and important cryptography techniques and can serve as a good basis in understanding the importance of number theory as a field.

**Sieve of Eratosthenes**

The Sieve of Eratosthenes is an algorithm that helps us find all the prime numbers smaller than a given number.

It consists of the following steps:

1. Create a list of numbers from 2 to the given limit.
2. Start with the first number in the list (2), which is a prime number.
3. Mark all the multiples of 2 as composite numbers by crossing them out from the list.
4. Move to the next unmarked number (3) in the list, which is also a prime number.
5. Mark all the multiples of 3 as composite numbers by crossing them out from the list.
6. Repeat steps 4 and 5 until you reach the square root of the given limit (or the next unmarked number is greater than the square root).
7. The numbers remaining in the list after performing steps 2 to 6 are all prime numbers.

This algorithm roots out all the multiples of prime numbers smaller than the upper limit, thus leaving out only prime numbers themselves behind.

**Euclid theorem**

There are infinitely many prime numbers.

It can be easily proved.

**Prime Number Theory**

The theorem states that the number of primes less than or equal to a given positive integer n, denoted as π(n), is approximately equal to n/ln(n) as n tends to infinity, where ln(n) represents the natural logarithm of n.

lim(n→**∞**) [π(n) / {n/ln(n)}] = 1

This means that as n becomes larger and larger, the ratio of the actual number of primes π(n) to the estimated value of n/ln(n) approaches 1.

This was a very important result that forever shook the world of arithmetic. This result, coupled with several others, together gave us ground breaking techniques that we use for various applications in Mathematics, Statistics, Number Theory, Computer Science and Network Security to name a few.

We have stated an easy yet elaborate proof for the same in our video.

In the scope of our project, this theorem, along with the Fermat’s Little Theorem, have been utilized to generate large prime numbers in the matter of just a few MILLISECONDS!!

**Fermat’s Little Theorem**

The Fermat’s Little Theorem states that for any prime number **p** and any number **a**,

a^p ≡ a (mod p)

Furthermore, if **a** and **p** are coprimes, we see that

a^(p-1) ≡ 1 (mod p)

Once again, we have discussed an easy and elaborate proof for this theorem.

The above given 2 theorems help us in generating huge prime numbers in just a few milliseconds!!!!

The following programme is used to implement the generation of primes -

import random as r

#This function is responsible for running the test on primes

def mod(n):

#n is the integer being tested

b=bin(n-1)

b=str(b)

l=len(str(b))

d=[]

c=0

for i in range(100):

a=r.randint(2,n)

t=a

for j in range(l-1,1,-1):

t=t%n

if b[j]=='1':

d.append(t)

t=t\*\*2

t=1

for k in d:

t=t\*k

t=t%n

if t==1:

c+=1

else:

break

if c==100:

return 1

else:

return 0

#The following is the part of generating a number and checking for prime

z=0

while (z==0):

k=r.randint(10\*\*13,10\*\*14)

z=mod(k)

else:

print(k)

This programme utilizes the probability of obtaining a prime (using Prime Number Theorem) and Fermat’s Little Theorem Method to generate a random number that is mostly prime.

**Diffie Hellman Key Exchange**

It is a special technique that utilizes prime numbers to help secure any communication. It involves a special encryption technique that works similar to a special lock with 2 keys and corresponding key slots that only allows locking and unlocking in a specific manner - if a specific key, say key 1, is used for locking, then only the remaining key can be used to unlock the key, i.e. key 2. This special locking actually involves the use of prime numbers along with strings.

Let us understand how this lock provides security and authentication.

This technique involves 2 keys for every lock - a private key and a public key. The private key is present with the owner only while the public key is available to everyone. Let’s understand with an example how this technique works.

Two people A and B wish to have secure communication. A wishes to send a message to B. He will follow the following steps -

1. Take his message and lock it with his own lock using his private key
2. Lock the whole locked package using B’s lock using B’s public key

It leads to a double locking for the message. Now, in order to open the package, anyone should’ve B’s private key which is not possible. Therefore no one except B can open this package.Furthermore, the second lock can only be opened by A’s public key. Therefore, it will imply that a person must have locked it with A’s private key - this person can only be A. Therefore, the two locks ensure that -

1. Only B can open the message
2. Authenticate that the message is from A

These two factors will ensure that the communication between A and B is always secure.

**RSA (cryptosystem):**

The RSA algorithm is a widely used cryptographic algorithm for secure communication and data encryption. It is an asymmetric encryption scheme, meaning it uses two different keys for encryption and decryption: a public key and a private key. Here's an explanation of how the RSA works:

Key Generation: Select two large prime numbers, p and q. Compute their product, n = p \* q, which serves as the modulus for the keys. Calculate Euler's totient function of n, φ(n) = (p - 1) \* (q - 1). Choose an integer e (the public exponent) that is relatively prime to φ(n) and less than φ(n). Calculate the private exponent d, which is the modular multiplicative inverse of e modulo φ(n).

Encryption: To encrypt a message M, the sender uses the recipient's public key (n, e). Convert the message M into a numerical representation, typically using a specific encoding like ASCII. Compute the ciphertext C as C = M^e mod n, where ^ represents exponentiation and mod is the modulus operator.

Decryption: The recipient uses their private key (n, d) to decrypt the ciphertext C. Compute the plaintext message M as M = C^d mod n. The security of RSA relies on the difficulty of factoring large composite numbers. Breaking RSA encryption involves factoring the modulus n into its prime factors, which becomes computationally infeasible for sufficiently large primes. RSA has several applications in modern cryptography, including secure communication, digital signatures, and key exchange protocols. It provides a way for individuals and organizations to securely transmit sensitive information over public channels, as long as the private keys are kept secret.