Selected Topics of Embedded Software Development 2

WS-2021/22

Prof. Dr. Martin Schramm

Testing and generating Prime Numbers and Safe Primes using CryptoCore

Group 2 – Team 4
Rashed Al-Lahaseh – 00821573
Vikas Gunti - 12100861

Supriya Kajla – 12100592

Srijith Krishnan – 22107597

Wannakuwa Nadeesh – 22109097

Miller Rabin Test Algorithm

Flow Chart

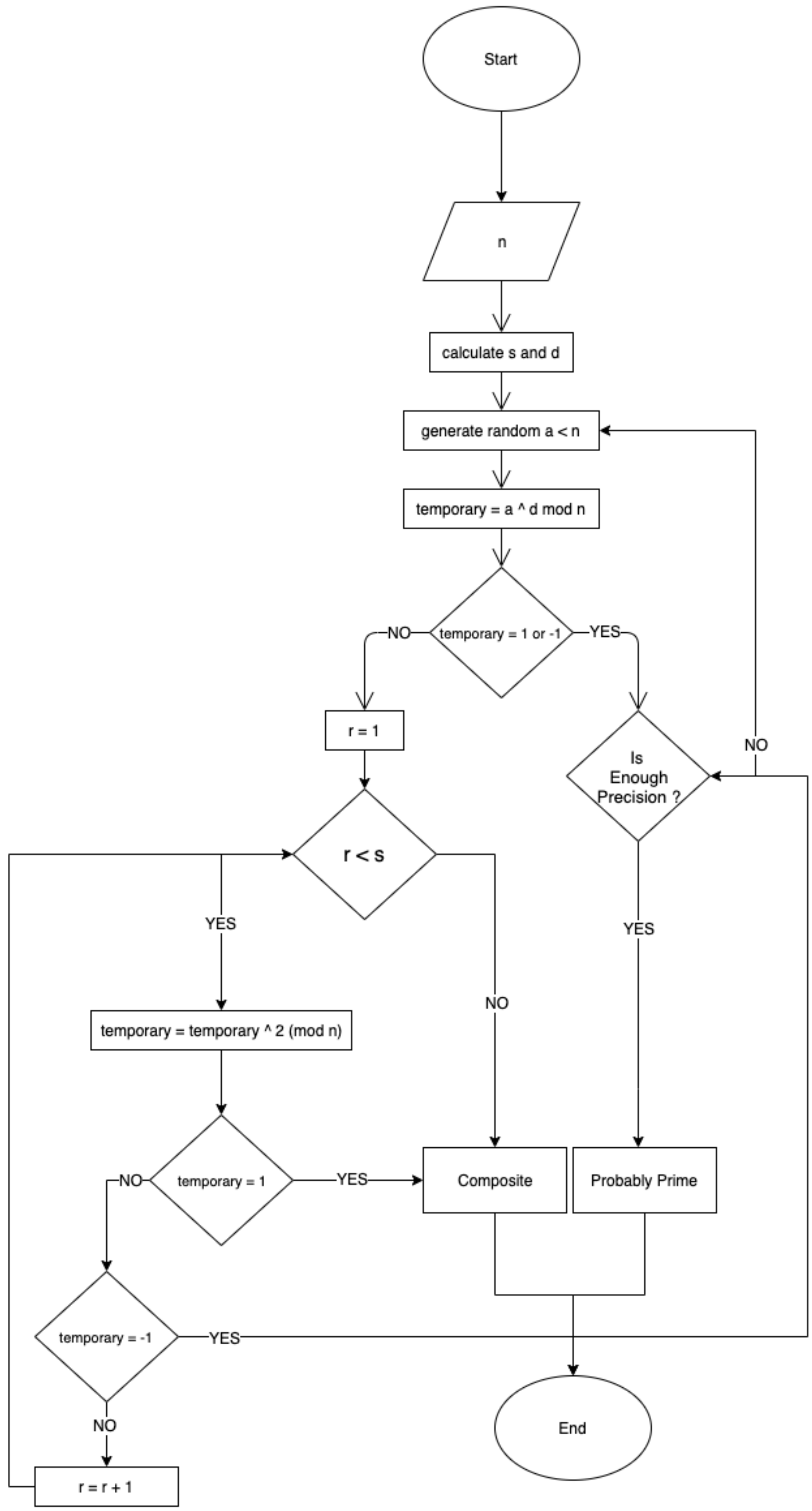

Figure 1: MRT Functionality Flow Chart

Example

1- Enter input n

N = 139

N – 1 = 10_138 | 2_10001010

2- Calculate s and d

S = Number of shifts to make n-1 odd

D = is the no: after shifting.

Here is s = 1 and d = 10_69 | 2_ 1000101

3- Generate a random number a

Let's say we got a = 30

4- Calculate the value of

Temporary= a ^ d mod n = 30 ^ 69 mod 139 = 1

5- Check if the result equals 1 or -1

If yes, then maybe Prime else continue process

6- Check if r is less than s

7- If yes, then calculate temporary = temporary ^ 2 mod n

8- And Check temporary result if its 1 then its composite if its -1 then maybe prime else loop continue until r = s-1

Implementation

```
def miller_rabin(n, k):
    if n == 2:
        return True
    if n % 2 == 0:
        return False

    s = 0
    d = n - 1

    while d % 2 == 0:
        s += 1
        d //= 2

    for _ in range(k):
        a = random.randrange(2, n - 1)
        temporary = pow(a, d, n)
        if temporary == 1 or temporary == n-1:
            continue
        for _ in range(s - 1):
            temporary = pow(temporary, 2, n)
            if temporary == 1:
                return False
            elif temporary == n - 1:
                break
        else:
            return False

return True
```

Sensitivity Parameter

Input Number

Calculating the Value of s and d

Generate a random base for a

Validating Result

Figure 2: MRT Algorithm Implementation

Example

```
##Example:

input numbe is 17

d = n - 1 = 17 - 1 = 16

10_16 | 10000_2

d = 1
s = 4

generate random value for base a, a = 8

a ^ d == 1 mod n
    8 ^ 1 == 8 mod 17 -> ? continue

is there a 'r' in the set {0, ..., s - 1} = {0, 1, 2, 3}
    for which ((a ^ d)^{2^r}) = -1 mod n holds true?

r = 0
    8 ^ 1 = 8 mod 17 /= -1 mod 17
r = 1
    ((8 ^ 1)^{2^1}) = 8 ^ 2 = 64 mod 17 = 13 mod 17 /= -1 mod 17
r = 2
    ((8 ^ 1)^{2^1})^2 = 169 mod 17 = -1 mod 17 => Maybe prime
```

Figure 3: MRT Example

Generate Numbers

```
def gen_prime(bits):
    while True:
        a = (random.SystemRandom().randrange(1 << bits - 1, 1 << bits) << 1) + 1
        if miller_rabin(a,20):
            return a
```

Figure 4: Primality Random Number Generator