Homework 4 CS 290G Cryptographic Engineering

Magnus Settemsli Mogstad

May 2015

1 Fermats primality testing

Result from code

Carmichael number	Witness	Liar
41041	7	2
62745	3	2
63973	7	2
75361	11	2
101101	7	2
126217	7	2
172081	7	2
188461	7	2
278545	5	2
340561	13	2
449065	5	2
552721	13	2
656601	3	2
658801	11	2
670033	7	2
748657	7	2
838201	7	2
852841	11	2
997633	7	2
103369	7	2
1082809	7	2
1569457	17	2
1773289	7	2
2100901	11	2
2113921	19	2
2433601	17	2
2455921	13	2

Table 1: Fermat's primality test

Start Fermat's test with a=[2,n-1] since if you start with a=1 all the liars will return back 1(it said on wikipedia that it can have the value 1).
The property of the witness is that it is the smallest prime number you need to create that specific Carmichael number. Since a Carmichael number is created by multiplying 3 or more prime numbers. The first number 41041 is built up using the prime numbers 7*11*13*41, and the smallest witness we found was 7 which is the smallest prime number.

2 Miller-Rabin primality testing

It says in the foil from class that a=[1,n-1] but on wikipedia it says that a=[2,n-1], if you run with a=[1,n-1] you end up with 1 as witnesses and 2 as liars on everyone of the numbers. But if you choose to run with a=[2,n-1] you get these numbers.

Carmichael number	Witness	Liar
41041	2	16
62745	2	16
63973	2	9
75361	2	256
101101	2	16
126217	2	16
172081	2	9
188461	2	9
278545	2	98
340561	2	35
449065	2	16
552721	2	21
656601	2	16
658801	2	101
670033	2	9
748657	2	9
838201	2	9
852841	2	16
997633	2	898
103369	2	9
1082809	2	16
1569457	2	256
1773289	2	3
2100901	2	16
2113921	2	195
2433601	2	98
2455921	2	9

Table 2: Miller-Rabin primality test

3 Python code

```
import random
import math

def fermat(n,a):
    x = fastermod(a,(n-1),n)
    if(x!=(1%n)):
        return True
    return False

def millerRabin(n,a):
    k,m =  decompose(n-1)
    x = fastermod(a,m,n)
    if((x%n)==1):
        return False
    for j in range(0,k-1):
        if(x==-1%n):
            return False
        else:
            x=(x**2)%n
    return True

def decompose(n):
    exponentOfTwo = 0
    while n % 2 == 0:
        n = n/2
        exponentOfTwo += 1
    return exponentOfTwo, n

def fastermod(factor,power,modulus):
    result = 1;
    while(power > 0):
        if (power % 2 == 1):
            result = (result*factor) % modulus
            power = power-1

        power = power/2;
        factor = (factor*factor)%modulus
    return result


numbers = []
numbers.append(41041)
numbers.append(62745)
numbers.append(63973)
```

```
numbers.append(75361)
numbers.append(101101)
numbers.append(126217)
numbers.append(172081)
numbers.append(188461)
numbers.append(278545)
numbers.append(340561)
numbers.append(449065)
numbers.append(552721)
numbers.append(656601)
numbers.append(658801)
numbers.append(670033)
numbers.append(748657)
numbers.append(838201)
numbers.append(852841)
numbers.append(997633)
numbers.append(1033669)
numbers.append(1082809)
numbers.append(1569457)
numbers.append(1773289)
numbers.append(2100901)
numbers.append(2113921)
numbers.append(2433601)
numbers.append(2455921)

def runFermat():
    witnesses = []
    liars = []
    for i in range(0,len(numbers)):
        tempWitness=0
        tempLiar=0
        for a in range(2,int(numbers[i])):
            bol = fermat(numbers[i],a)
            if(bol and tempWitness==0):
                tempWitness=a
            elif(not(bol) and tempLiar==0):
                tempLiar=a
            if(tempLiar!=0 and tempWitness!=0):
                break
        witnesses.append(int(tempWitness))
        liars.append(int(tempLiar))
    print "Fermat"
    printNice(witnesses,liars)


def runMillerRabin():
```

```
    witnesses = []
    liars = []
    for i in range(0,len(numbers)):
        tempLiar=0
        tempWitness=0
        for a in range(2,int(numbers[i])):
            bol = millerRabin(numbers[i],a)
            if(bol and tempWitness==0):
                tempWitness=a
            elif(not(bol) and tempLiar==0):
                tempLiar=a
            if(tempLiar!=0 and tempWitness!=0):
                break
        witnesses.append(int(tempWitness))
        liars.append(int(tempLiar))
    print "Miller-Rabin"
    printNice(witnesses,liars)

def printNice(witnesses,liars):
    for i in range(0,len(numbers)):
        print "NUMBER: ",numbers[i]
        print "Witness: ", witnesses[i]
        print "Liar: ",liars[i]

def main():
    runFermat()
    runMillerRabin()
```